CREATION OF A TRIMMED POWERED AERODYNAMIC DATABASE FOR A GENERIC HYPERSONIC VEHICLE

By

Cannon James DeBardelaben

Kidambi Sreenivas
Professor, Mechanical Engineering
(Chair)

Charles Margraves
Associate Professor, Mechanical Engineering
(Committee Member)

James Newman
Department Head, Mechanical Engineering
(Committee Member)

CREATION OF A TRIMMED POWERED AERODYNAMIC DATABASE FOR A GENERIC HYPERSONIC VEHICLE

By

Cannon James DeBardelaben

A Thesis Submitted to the Faculty of the University of
Tennessee at Chattanooga in Partial
Fulfillment of the Requirement of the Degree
of Master of Science: Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

December 2021

ABSTRACT

The purpose of this thesis is to propose and evaluate a methodology for creating an inviscid trimmed powered database for a generic airbreathing hypersonic vehicle. The database was created using the cartesian flow solver Cart3D. Propulsion effects were coupled to the airframe and inlet by using conditions at the inlet of the combustor to determine conditions at the exit of the combustor. This was done by solving the quasi-one-dimensional momentum equation with heat addition. Selected trajectory points were also evaluated using the viscous solver, FUN3D. The viscous solutions contain effects such as boundary layer separation which had a pronounced effect on the computed axial force. In all cases, the viscous solutions predicted a lower thrust than the inviscid solutions, mainly due to the presence of viscous shear stresses. Experimental data would be required to further evaluate the validity of the assumptions made in both the inviscid and viscous analysis.

ACKNOWLEDGEMENTS

I would like to thank Dr. Kidambi Sreenivas for answering my endless questions and serving as my advisor and committee chair. In addition, I would like to thank Drs. Newman and Margraves for furthering my understanding of the material contained in this thesis, as well as serving on my committee. I would also like to thank my research partners Jason DeHay, Jameson Snuggs, and Alex Snyder. Funding was provided through the Reusable Hypersonic Vehicle Structures program by Air Force Research Laboratory, Aerospace System Directorate, High Speed Systems Division (AFRL/RQH), Prime Contract Number FA8650-18-C-2253. Computing resources were provided by the SimCenter at the University of Tennessee at Chattanooga. In particular, the ”EPYC” and ”ETEST” clusters, acquired via NSF Award #1925603, were used for the viscous simulations. The ”Firefly” cluster was used for the inviscid simulations. Images were created by using FieldView as provided by FieldView CFD, Inc. through its University Partners Program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AFRL	Air Force Research Lab
CFD	Computational Fluid Dynamics
RANS	Reynolds-Averaged Navier-Stokes

LIST OF SYMBOLS

χ	Ratio of x location to combustor length
$\dot{m}$	Mass flow rate
η_b	Burner efficiency
γ	Specific heat ratio
ϕ	Equivalence ratio
ψ_{st}	Stoichiometric fuel to air ratio
θ	Shape parameter for combustor temperature profile
A	Cross sectional area
C_A	Axial force coefficient
C_N	Normal force coefficient
C_p	Pressure coefficient
c_p	Specific heat at constant pressure
C_x	Force coefficient in the x direction

C_y Force coefficient in the y direction

CFD Computational fluid dynamics

D Hydraulic diameter

f Friction coefficient

F_A Axial force

F_D Drag force

F_L Lift Force

F_N Normal force

h_l Lower heating value

I_{sp} Specific impulse

M Mach number

p Static pressure

P_0 Total pressure

R Radius

T Temperature

T_0 Total Temperature

y x-velocity over velocity magnitude ($V_x/|V|$)

CHAPTER I

INTRODUCTION

The thesis is organized as follows. The first chapter contains the introduction, including a background, problem statement, an objectives. The second chapter consists of a review of the literature. The third chapter contains the methodology of the research performed. First, the geometry of the generic hypersonic vehicle is described. Then the propulsion computation and flow solvers are explained including the usage of the inviscid solver, Cart3D, and the viscous solver, FUN3D. Chapter four presents the results of the inviscid database, as well as the viscous simulations. The final chapter provides a conclusion and a recommendation for future work.

Background

Air-breathing hypersonic vehicles have been of interest to the international community since around the late 1950s and early 1960s. The general appeal of such a vehicle is built around its ability to achieve sustained hypersonic flight at much higher efficiencies than traditional chemical rockets. This is due to the fact that hypersonic vehicles are only required to carry fuel for propulsion, as opposed to both fuel and oxidizer. There are a few other key differences between air-breathing hypersonic vehicles and rockets. First, although hypersonic vehicles do not need to carry oxidizer with them, the incoming air must still be processed in some way to achieve

propulsion; the vehicle must have an inlet. This results in a higher structural weight of engines for airbreathing hypersonic vehicles compared to rocket engines. Another disadvantage of hypersonic vehicles is that their performance is a function of altitude, angle of attack, and Mach number. In order to achieve sufficient performance, hypersonic vehicles are restricted to the atmosphere, whereas rockets have no such restrictions and are effective in a vacuum and in the atmosphere. Finally, sustained hypersonic flight in the atmosphere can lead to significant issues, such as high aerodynamic and heating loads. However, this is traded off with the ability to use the air to maneuver, a significant advantage over an ordinary rocket. What Smart [2] calls the holy grail of hypersonic air-breathing propulsion is a vehicle that could deliver a payload, such as a satellite or an astronaut, to low-earth orbit. With the current state of technology, this would likely not be possible as an ideal "single-stage-to-orbit" vehicle, but as a stage in a larger vehicle. A turbojet could be used to get the vehicle to Mach 3+, where a scramjet could take over to boost the payload through the bulk of the atmosphere, and a rocket would handle the final orbital insertion. A major difficulty with this configuration would be designing a scramjet that could take over from the turbojet around Mach 3 while still being efficient at much higher Mach numbers [2].

To properly explain how scramjets function, it is important to first mention turbojets and ramjets briefly. A turbojet is a traditional jet engine where air first enters a diffuser to slow the flow down, raising the pressure. The flow then enters a (usually multi-stage) compressor where the pressure and temperature are raised further to prepare the air for combustion. After combustion, the air passes through a turbine, which removes just enough energy to power the compressor and other necessary auxiliary devices, before leaving through the nozzle to produce thrust. In general,

turbojets are effective up to around Mach 3 (see Figure 1.3). In contrast, ramjets operate in the range of Mach 3 to 6. The stages of a ramjet are similar to those of a turbojet. Instead of using a compressor, ramjets compress the incoming flow using the vehicle geometry to decelerate the incoming flow, usually through a series of oblique shocks. At the end of the oblique shocks is a normal shock where the flow decelerates to subsonic speeds, accompanied by a further increase in pressure. Fuel is added, which mixes and burns as the flow exits the engine through a converging-diverging nozzle which accelerates the flow back to supersonic speeds and produces thrust. Figure 1.1 shows a schematic of a generic ramjet.

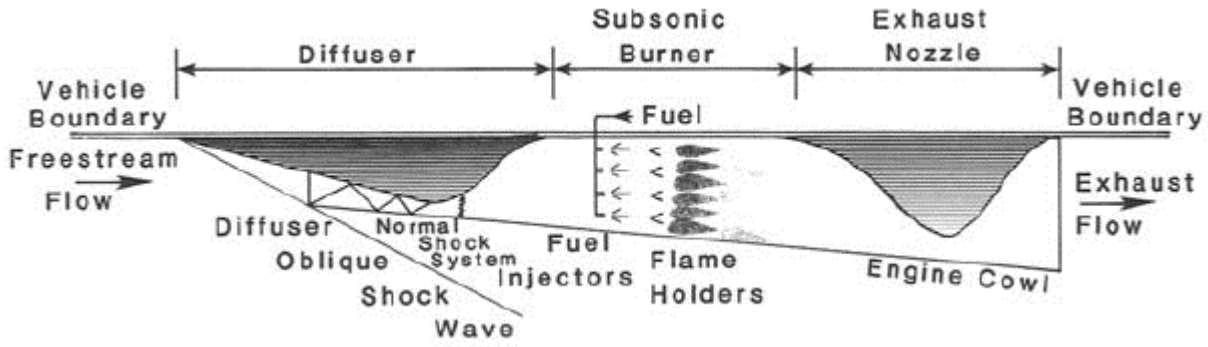


Figure 1.1

Generic Ramjet Geometry[1]

Scramjets are similar to ramjets in that they require no moving parts with the key difference being that in a scramjet, the fuel is burned while the flow is still moving at supersonic speeds. Scramjets are designed to operate in Mach numbers from around 4-6 and up. After a certain point, it is no longer advantageous to bring the incoming flow down to subsonic speeds due to

the extremely high temperatures and pressures that occur after the shocks at high Mach numbers. Since the flow is already supersonic in the combustor, only a diverging nozzle is required to expand the exhaust to produce thrust [1]. Figure 1.2 shows a schematic of a generic scramjet.

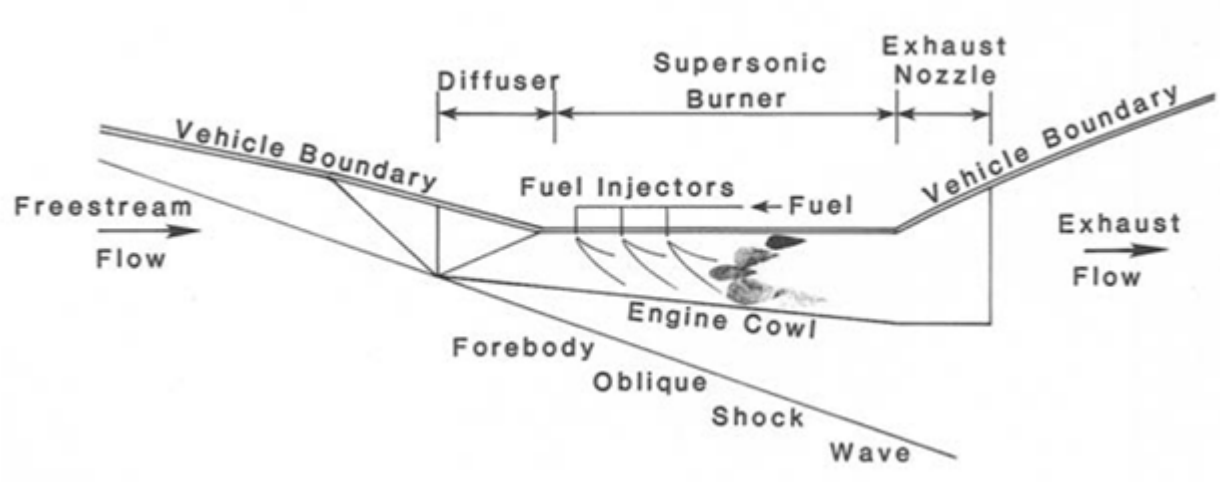


Figure 1.2

Generic Scramjet Geometry[1]

As previously mentioned, scramjets have higher efficiencies than rockets. Efficiencies of engines are often measured using specific impulse (I_{sp}). Specific impulse is the ratio of thrust to the mass flow rate of the propellant. Figure 1.3 shows the specific impulse of the three types of air-breathing propulsion mentioned compared to rockets. Note how the scramjet has a much higher specific impulse than rockets, and can theoretically operate at a much wider range than the other two air-breathing engines.

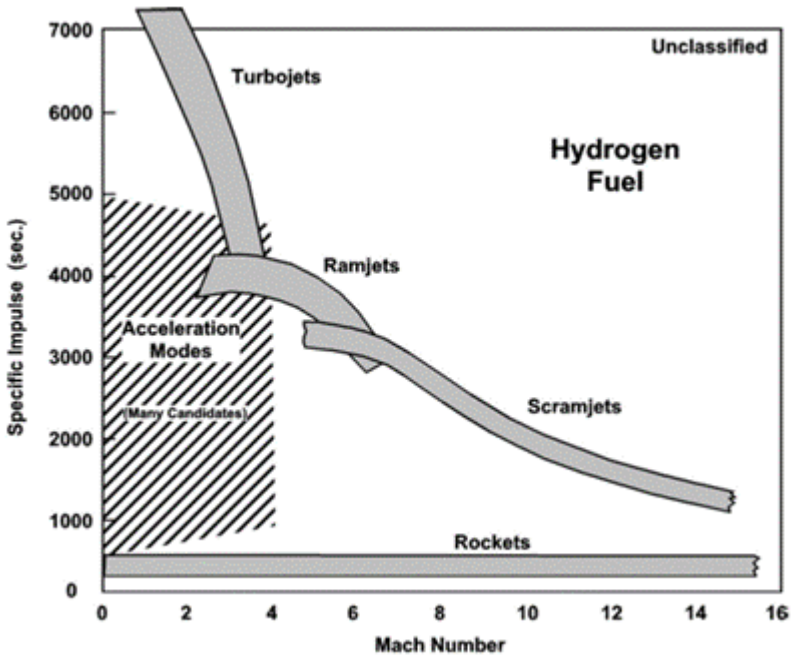


Figure 1.3

Specific Impulse of Various Propulsion Systems[1]

Scramjet engines do not come without their own challenges, however. Combustion at supersonic speeds is difficult due to the extremely short amount of time spent in the combustor. These short residence times can lead to incomplete combustion which results in a loss of thrust. In addition, there is the issue of thermal choking where the engine can unstart or go subsonic if too much fuel is added. As seen from figure 1.3, scramjets generally cannot operate or produce very little thrust below Mach 3.5 or 4. In addition, designing a scramjet engine that can operate over much or all of the very wide theoretical operating range is very difficult and would require a complicated and/or heavy variable geometry [2].

Statement of the Problem

Although hypersonic flight has received significant investment since the 1950s, the technology still has a long way to go and many challenges to overcome. One of the biggest pushes has been in the development of tools, both experimental and computational, in order to generate hypersonic vehicle data at much lower costs than through flight testing, which is often expensive and impractical. Although necessary, experimental data from ground testing can also be extremely expensive or have low fidelity. Often, experimental data includes a few pressure taps on a few points of interest in a proposed flight trajectory. However, through the use of Computational Fluid Dynamics (CFD), hundreds of trajectory points can be run in a relatively short amount of time. Each of these runs contains important flow data at any point on the vehicle. A very common approach is as follows. First, researchers will identify points of interest to run in the experimental facility. Then, CFD Engineers will run those specific points to tie the simulation to the experimental data. Once the engineers are confident in their simulation results, the entire trajectory can be simulated. However, within CFD, there are varying levels of fidelity. In general, there are potential or panel flow methods that are the lowest fidelity. Next are the inviscid Euler Codes, with viscous Navier-Stokes codes having the highest fidelity. However, at hypersonic speeds, more physics begins to become important than that modeled by a generic Navier-Stokes flow. At hypersonic speeds, the chemical properties begin to change. This changes the nature of the governing equations and requires more complicated CFD solvers which can handle such complication. With the increase in complication comes an increase in computational resources (number of computers, time, etc.) required to complete such a simulation. It is for this reason

that it is very important to determine what level of fidelity is required, useful, or feasible for any given project. Often, designers are interested in simulating a vehicle over its entire intended trajectory. Such a trajectory simulation requires a trimmed aerodynamic database to determine the performance of the vehicle at any given point. Creating a trimmed powered hypersonic database requires hundreds of CFD simulations. Historically, many hypersonic scramjet databases were created by running an inviscid or potential flow simulation for the outside of the vehicle with separate simulations for the scramjet engine. Creating a database in this way partially decouples the engine from the aircraft body. However, in current scramjet designs, including the one studied in this research, the engine and the airframe are one and the same. Ideally, an aerodynamic database of a scramjet vehicle would use the most accurate model possible, including viscosity, changing physical properties as the air heats up, and changing chemical properties as the fuel combusts in the engine. However, such a simulation is extremely costly and currently not possible using the resources available at UT Chattanooga. For reference, inviscid solutions using Cart3D take about 3 hours to complete while running on 16 computing cores. In contrast, a standard viscous solution which is not even taking into account varying physical and chemical properties using FUN3D takes approximately 16 hours on 256 cores. This means on a 512 core machine, 32 Cart3D simulations can be run simultaneously compared to only 2 cases using FUN3D. It is for this reason that the aerodynamic database involving over 400 CFD simulations was run using an Euler (inviscid) flow solver. The cost in accuracy associated with an inviscid solution over a viscous one will be discussed later in this thesis.

Objectives

This thesis aims to address the intricacies and complexities of creating an inviscid trimmed powered database of a hypersonic scramjet vehicle. The database includes tables of axial force coefficient (C_A), normal force coefficient (C_N), and elevon deflections required to trim the vehicle for a range of flight conditions representative of a potential trajectory. Conditions were chosen to encompass a similar range as Ruttle et al [3]. The Mach numbers chosen are 5,6, and 7. The angles of attack analyzed are -4, -2, 0, 2, 4, and 6. The equivalence ratios (ϕ) chosen are 0.5, 0.6, 0.7, 0.8, 1.0, and 1.2. Propulsion boundary conditions were determined by solving the one-dimensional compressible flow equation with heat addition and area variation. In all, this resulted in a trimmed aerodynamic database consisting of 162 points. In addition to the inviscid database, selected cases will also be run using a viscous solver to attempt to evaluate the fidelity of the inviscid solutions and examine possible strengths and shortcomings.

CHAPTER II

LITERATURE REVIEW

Ruttle et al. [3] provide the general framework on which this paper is based. The Air Force Research Laboratory (AFRL) saw a gap in the current available hypersonic vehicle geometries; many of them are not available for public use. As a response, AFRL created a series of Generic Hypersonic Vehicles (GHV) called the Road Runner family in order to allow for greater collaboration in the hypersonic community. This family of vehicles has all the main mission parameters in common. They are designed to cruise at Mach 6 at a dynamic range of 1000-2000 psf. The vehicles possess an axisymmetric scramjet engine, which is different from many of the other geometries often present in the literature. The vehicles were designed around flying a basic trajectory consisting of a boost to Mach 4, acceleration to Mach 6, and subsequent powered maneuver followed by an unpowered descent. The conditions at the inlet were determined from CFD codes, but it is not mentioned whether a viscous or inviscid model was used. The engine-specific impulse was computed with a separate code. The aerodynamic database was determined by using the panel code S/HABP [4]. The trajectory was then flown by combining the aerodynamic loads with the aforementioned I_{sp} calculations. The final results include estimations of aerodynamic loads, trimmed elevon deflections, and engine performance metrics for a large range of Mach numbers, angles of attack, and equivalence ratios. One of the

goals of this thesis is to replicate the approach of the creation of this aerodynamic database but from a different perspective. Instead of decoupling the propulsion from the aerodynamic loads and elevon deflection, this thesis will consider all these factors simultaneously, while getting the benefit of higher accuracy of an Euler CFD code compared to the panel code used by Ruttle et al.

In 2001, Cockrell et al. [5] at NASA analyzed, tested, and flew the X-43A research vehicle, also called the Hyper-X. One of the main goals of the project outside of testing a hypersonic vehicle was to help develop the technology required to design future air-breathing hypersonic vehicles. This project was the first time researchers were able to run CFD simulations and compare it directly to a flight-tested scramjet vehicle. The vehicle was boosted to the flight Mach number, either Mach 7 or Mach 10, where the Hyper-X opened the cowl over the inlet and burned its hydrogen fuel for approximately 7 seconds. One important part of the analysis of a hypersonic vehicle that was excluded from this thesis for simplicity is the high-temperature effects on the incoming air. In the Hyper-X CFD computations, curve fits were used to determine the thermodynamic quantities. The exhaust was assumed to have a single constant composition to reduce the computational requirements. The authors correctly point out the high complexity of modeling a scramjet; the external and internal flow fields must be analyzed, with special care to ensure their impacts on each other are not neglected. The authors also state the importance of determining pitching moments and control surface deflections, as trim drag can have significant impacts on the vehicle's performance. Most of the simulations were run using a structured grid Navier-Stokes flow solver. The solver contains frozen, equilibrium, and finite-rate chemistry models, and the Baldwin-Lomax turbulence model was used. The internal flow path and scramjet combustor were analyzed using

two additional codes. One code was used to model the scramjet engine flow path. This was done by solving the equations for mass, momentum, energy, fuel, and turbulence fields over a rectangular duct with variable area. The other code was used to solve the inlet using a 2D axisymmetric Euler flow solver, as well as a one-dimensional chemical equilibrium code used to model the combustion. In addition, the second code has the capability to model the viscous effects and add them to the solution. For the inviscid solutions, the Euler code was used to determine the forces on the airframe, with the combustor code being used to determine the force contributions from the engine itself. Only one viscous case was run where the entire vehicle was considered simultaneously. In this case, the combustor was still modeled using the one-dimensional code. The numerical solutions were in reasonable agreement with the wind tunnel data. The viscous and inviscid solutions both predict the normal force reasonably well. The viscous solution is also able to predict the axial force, whereas the inviscid solutions under predict. Neither set of solutions are fully able to predict the measured pitching moment. The authors conclude with a general satisfaction of the performance of the CFD simulations compared to the experiment.

Keshmiri et. al [6] are analyzing a Generic Hypersonic Vehicle which is not of the Road Runner family developed by AFRL. This particular vehicle was developed and simulated at NASA Langley, Rockwell International, and California State University LA. In addition, NASA Langley conducted experimental tests in a wind tunnel. The results of the CFD and experimental simulations were digitized and organized into lookup tables for the paper. The aerodynamic database was created using the wind tunnel data combined with the CFD results. Variation in Mach number, angle of attack, left and right elevon and rudder deflections were considered. The authors

explain that a simple lookup table is not a good option for control system design, so curve fit analytical expressions were generated for the aerodynamic coefficients. Since the flight trajectory of this vehicle is akin to a single stage to orbit vehicle, the analytical expressions for aerodynamic results are split into subsonic, supersonic, and hypersonic regimes to improve efficacy. In general, the curve fits appear to fit the experimental data reasonably well. The authors also compare the selected CFD cases to the available experimental data. At hypersonic speeds, the CFD codes over predict C_L/C_D drastically. This is most likely due to the CFD simulations under predicting C_D. The engine of this particular GHV is a combined cycle turbojet, ramjet, and rocket motor, with the rocket motor taking over from Mach 4 to Mach 24. The propulsion effects were applied by simply adding on thrust based on available I_{sp} data. The authors then used this flight model to fly a six degree of freedom simulation. Although this paper has similar goals as this thesis, its approach is very different. The database is mostly based on experimental data. In addition, the vehicle does not use a scramjet, as it uses a rocket motor above Mach 4. Furthermore, since it is using a rocket motor in the hypersonic regime, it is much easier to simply add the propulsion effects after the fact, since one does not need to consider the complicated flow physics inside the inlet or isolator, nor the matter of burning supersonic flow in the combustor.

The next paper was written by the same authors as the previous paper and concerns a methodology for computing thrust on a ramjet/scramjet engine [7]. The inlet is modeled as a 2D forebody, compressing the flow through oblique shocks. The engine is modeled as a dual-mode ramjet/scramjet with variable area. While ideal for an engine that would have to handle incoming subsonic or supersonic flow, such a mechanism would need to handle the high heats

in the combustor. In addition, it would significantly increase the weight and complexity of the engine. A 1D flow with liquid hydrogen combustion was used to model the combustor itself. The nozzle flow was computed using the method of characteristics. The 2D inlet model, while simple, does allow for the performance of the engine to be analyzed with varying altitudes, flight Mach numbers, and angles of attack. The combustor will also need to have a varying area ratio to maintain a constant temperature in the combustor. This would require an actuator with a very fast response time, which would add a great deal of complexity to the engine. In all, while the paper presents useful techniques for analyzing a scramjet engine, it makes many assumptions that do not apply to a more feasible vehicle. In addition, the inlet and nozzle flow analysis was 2d only and would not apply to a complex 3D inlet such as the Roadrunner series of hypersonic vehicles. Finally, the model used in the inlet and nozzle is very simple and does not integrate with the actual shape of the vehicle in question.

Mirmirani et al. [8] evaluate modeling and simulation techniques for analyzing the aerodynamics, propulsion system, and structural mechanics of a generic hypersonic vehicle. In this case, the generic vehicle is 2D only. The authors begin by addressing some of the challenges in analyzing an air-breathing hypersonic vehicle, before going into analytical results, followed by CFD studies. One of the biggest challenges to realizable hypersonic flight is in the design of flight control systems. The main source of the difficulty comes from the unpredictability associated with hypersonic flight. High heating loads cause thermal stresses in the airframe which can, in turn, change the aerodynamics to off-design conditions. In addition, since the aircraft carries so much momentum, its trajectory is resistant to changes in attitude, making controlling the vehicle

more difficult. Hypersonic vehicles are also designed to operate over a high Mach number range. Many vehicles are designed to be stable at low Mach numbers, as they are meant to land and/or takeoff from a runway. Designing for low-speed stability means the vehicle becomes inherently unstable in the pitch direction at hypersonic speeds. The authors also highlight the inherent coupling between aerodynamic, thermal, and structural loads, as well as the propulsion system. It is difficult to talk about one of these factors without considering most or all of the others, which is why this thesis attempts to ensure the propulsion system is directly tied to the aerodynamics of the vehicle. Another challenge in hypersonic vehicle design is the serious lack of experimentally based aerodynamic databases and flight test data. After designing a 2D generic scramjet vehicle using analytical results from inviscid compressible flow theory, the design was analyzed using CFD. In order to reduce computational complexity, the vehicle was broken up into three pieces. The inlet of the vehicle was first solved using an inviscid solver. The combustor was simulated using a one-step finite rate model of hydrogen combustion. The data were compared to the X-43 wind tunnel data, as it is of a similar configuration, but the authors do not expect an exact match because they are in fact different geometries. The data do compare well with the general trends of the X-43 results. There is also a significant difference in pitching moment between the power on and power off conditions, further emphasizing the importance of considering the effects of power and aerodynamics together when determining the trimmed elevon position. The authors conclude by highlighting the importance of developing a model which is comprehensive enough, without providing unnecessary complexity.

Kline et. al. [9] examine the effect of shape deformation in the inlet to a scramjet due to heating and aerodynamic loads on the performance of the inlet, a challenge highlighted by the previous paper. Due to the very large range of flight conditions for which an air-breathing hypersonic vehicle is usually designed, there is uncertainty in the exact shape of the inlet because of the changing thermal aerodynamic loads. In these simulations, the deformation is not directly coupled to the CFD solutions; the thermo-mechanical and aerodynamic simulations are separate. The geometry which was studied was a vehicle with a rectangular inlet that transitions to an elliptical combustor. The inlet and isolator were solved using RANS CFD codes. The nozzle was represented simply by adiabatic expansion to freestream pressure. The combustor was modeled using the one-dimensional channel flow equations with variable area and heat addition, the same model used in this thesis. The heat addition was determined by assuming a temperature profile of the form

$$\tau(x) = 1 + (\tau_b - 1)\{\frac{\theta\chi}{1+(\theta-1)\chi}\} \tag{2.1}$$

where

$$\tau_b = \frac{T_{t4}}{T_{t2}} \tag{2.2}$$

and

$$\chi = \frac{x - x_3}{x_4 - x_3} \tag{2.3}$$

with conditions at point 2 representing the beginning of the isolator, and 3 and 4 representing the beginning and end of the combustor respectively. The issue of converting a high-fidelity CFD simulation into conditions to perform 1D analysis poses an interesting problem. Kline et. al.

followed a procedure of flux averaging which ensures the 1D properties possess the same mass, momentum, and energy as the flow at the 2D inlet of the combustor [10]. In this thesis, the 1D properties are determined by merely averaging the quantities over the 2D face. It would be a worthy topic of investigation to compare the results of arithmetic averaging of the primitive variables against the flux averaged results. The authors also point out the presence of nonuniformities in the solution at the inlet to the combustor but do not do anything to address the fact. The authors were forced to switch to inviscid simulations in the larger deformation cases due to viscous mesh quality issues. The inviscid results suggest that the inlet performance sensitivity is on the same order as the sensitivity due to basic manufacturing tolerances. However, the authors are wary of these results, as the RANS simulations suggest that there could be a greater effect once viscosity is taken into account.

Shu et al. [11] analyze another X-43 style 2D vehicle. The authors are specifically analyzing the difference in started and unstarted engine configurations. Although in general, a scramjet engine should always be run in the standard started mode, engine unstart will not be able to be completely avoided, especially since the technology is still in its infancy. The authors use unsteady viscous CFD to analyze the performance of the vehicle in both states. The results compare reasonably well with the pressure data taken from the experimental runs in the started configuration. For the unstarted configuration, the experimental data is treated as steady due to the poor temporal resolution of the probes. Thus, the unsteady CFD is used to attempt to explain what is occurring in the unstarted configuration. The aerodynamic forces fluctuate to an extreme degree in the unstarted condition; lift to drag ratio varies from 0.25 to 2.09. This is due to the

unsteadiness of the shocks on the forebody, which causes a ripple of unsteady behavior through the entire flowpath of the vehicle. In the end, the authors highlight the importance of considering the entire flowpath of the vehicle in the CFD analysis, and that the unstarted configuration is of a highly unsteady nature, and should be treated as such in all analyses.

The next paper concerns specifically the simulation of a scramjet combustor [12]. Due to the complexities of hypersonic flight and combustion, CFD and experimental results both have challenges that make it difficult to use one type of analysis without the other. Supersonic combustion creates a difficult environment for experimental data acquisition and so few points were able to be sampled. On the other hand, most CFD solvers are not able to fully model the complex physics of hypersonic flow, particularly with combustion. Because of this, it is important to have both computational and experimental data to tie the results together. The CFD simulations were completed using a Navier-Stokes code which can assume calorically or thermally perfect gases. The inlet and isolator before the combustor were simulated beforehand to provide realistic profiles at the entrance to the combustor. The supersonic combustion of ethylene was modeled using both a 3-step model and a 10-step model. The ignition was started by raising the temperature in the cavity of the combustor. After ignition, the flame stayed ignited in the cavity for the rest of the simulations. The 3-step model resulted in only a small amount of the mainstream fuel flow combusting, with the 10-step model resulting in almost no mainstream combustion. This is largely due to the flameholder not having a large enough effect on the core flow. To counter this, a shock train was used to ignite the core fuel flow. This caused the flow to go subsonic for a portion of the combustor which greatly increased combustion efficiency in this region. Once the flow went

supersonic again, the efficiency began to reduce. The authors recommend improvements to the combustor design, as it is intended to operate as a scramjet under Mach 6 freestream conditions, rather than a ramjet which was necessary for the CFD simulations to produce sufficient combustion. The purpose of reviewing this paper is to highlight the extreme challenges of simulating a scramjet combustor and are a major motivation for using a much simpler model for the studies completed for this thesis. It would not be feasible to generate an aerodynamic database while fully simulating the combustor geometry, even with a simplified reaction model; the challenges are just too great for current available computational resources.

The final paper to be reviewed follows a very similar procedure as the viscous simulations performed for this thesis [13]. The hypersonic vehicle to be analyzed has a very similarly shaped inlet to the Roadrunner GHV. The vehicle is boosted by a turbojet until around Mach 4 where a dual-mode ramjet/scramjet takes over up to the cruise Mach number of 8. Only two full nose-to-tail simulations were completed, one at Mach 6 and another at Mach 8. The authors reiterate the infeasibility of fully simulating the combustor, including chemistry, along with the inlet and nozzle. Instead, the combustor was removed from the computational domain and a 1D tool was used to estimate the conditions at the exit of the nozzle. The 1D tool is able to take into account heat losses, skin friction, and multiple fuel injection points. Static pressure, temperature, and velocity are averaged using stream-thrust averaging at the inlet to the combustor. Then the results of the 1D code are imposed at the exit of the combustor. The combustor was also simulated fully 3D with combustion. The results of the 3D simulations compare reasonably well, suggesting that the 1D model is very good for a very efficient model of the combustor effects.

In conclusion, in all the papers reviewed, there is a significant trade-off between the feasibility of obtaining a large number of computational results and fidelity. With the current state of computing power, it is nearly impossible to reasonably simulate enough data points for a trajectory of a hypersonic vehicle without resulting in a panel or Euler code. In addition, many of the geometries which were analyzed are very simple and are of the 2D inlet type, similar to the X-43. Furthermore, even the nose-to-tail simulations are forced to loosely couple the scramjet engine to the body in some way. Many simulations are run with the engine computed completely separately from the body, with the effects added together. Langener et al. [13] took this a step further by imposing boundary conditions back onto the vehicle to simulate propulsive effects with the rest of the vehicle. This is particularly important, as all papers emphasized the extent to which all parts of the vehicle's performance (structural, thermal, and aerodynamic) are coupled together. Finally, there is a severe lack of experimental databases and flight test data to which to tie the results of a simulation. Scramjet research still has a ways to go before full confidence is able to be placed in all CFD or experimental data.

CHAPTER III

METHODOLOGY

Geometry

For the purpose of this study, the Road Runner 1X was chosen from the GHV family. Figure 3.1 below shows the entire geometry with defined coordinate axes. The body is designed around the scramjet engine, with two vertical stabilizers and two elevons. Since the trajectory analysis was initially performed with only 3 degrees of freedom and will be only trimmed in the pitch axis, the elevons will be rotated together; however, if desired, they could be rotated differentially. The combustor is not included in the computational domain, as propulsion boundary conditions will be applied at the end of the combustor. The vehicle's internal flowpath is shown in Figure 3.2.

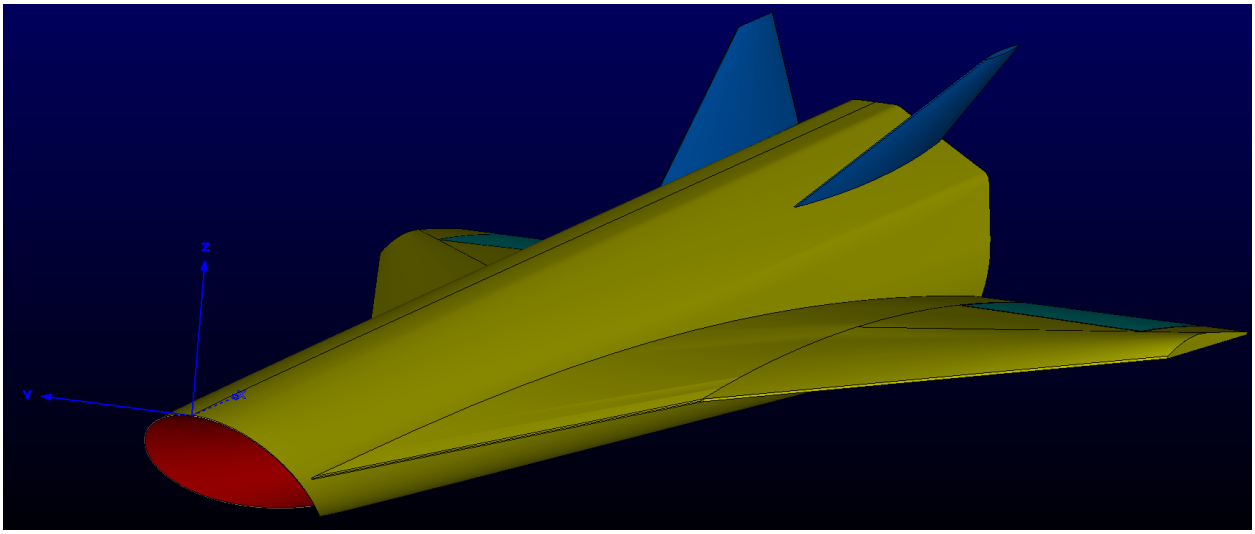


Figure 3.1

Roadrunner GHV Geometry

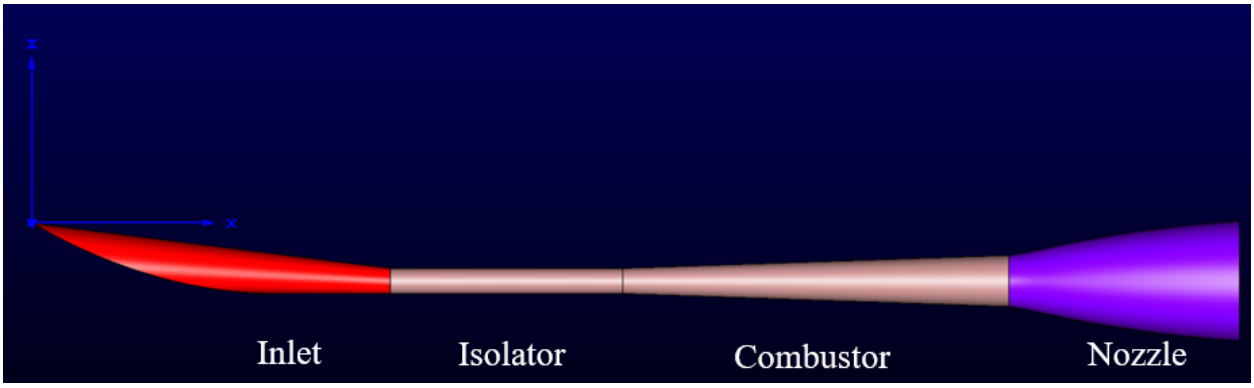
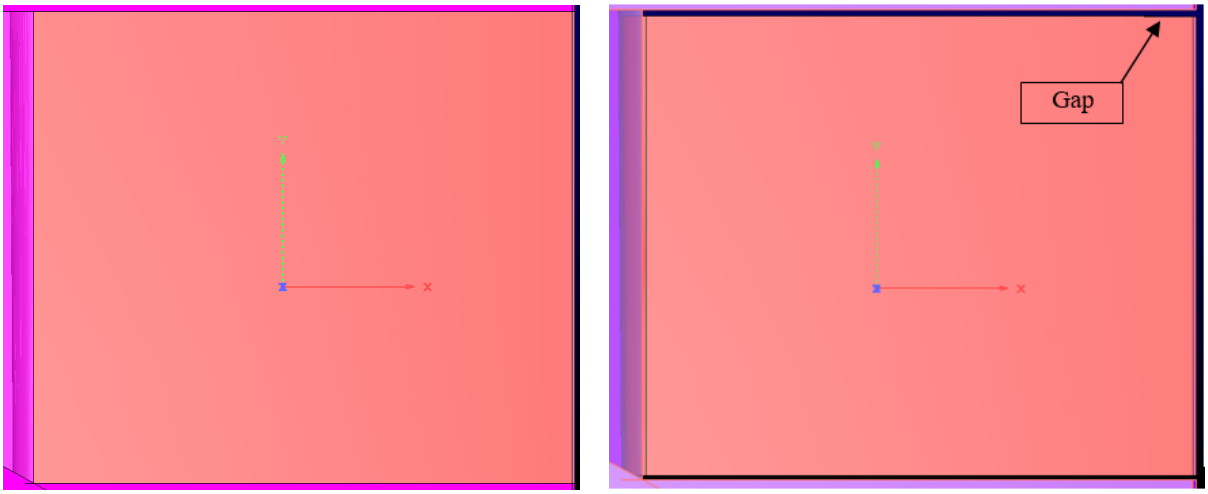


Figure 3.2

GHV Flowpath

The original Road Runner 1X geometry featured an elevon that was completely flush with the wing. However, to facilitate rotation, the geometry has been slightly modified to resemble a more realistic vehicle with gaps in between the elevon and the wing, as well as allowing for the rotation of the elevon without interference from the wing. The elevon was shrunk by a factor of 0.9. It was then moved towards the rear by around 0.1in. and centered in the gap. In Figure 3.3, the gap between the elevon and fuselage can be seen after modification. Figure 3.4 gives a more detailed view of the gap between the elevon and the wing.



(a) (b)

Before Modification After Modification

Figure 3.3

Elevon Before and After Modification

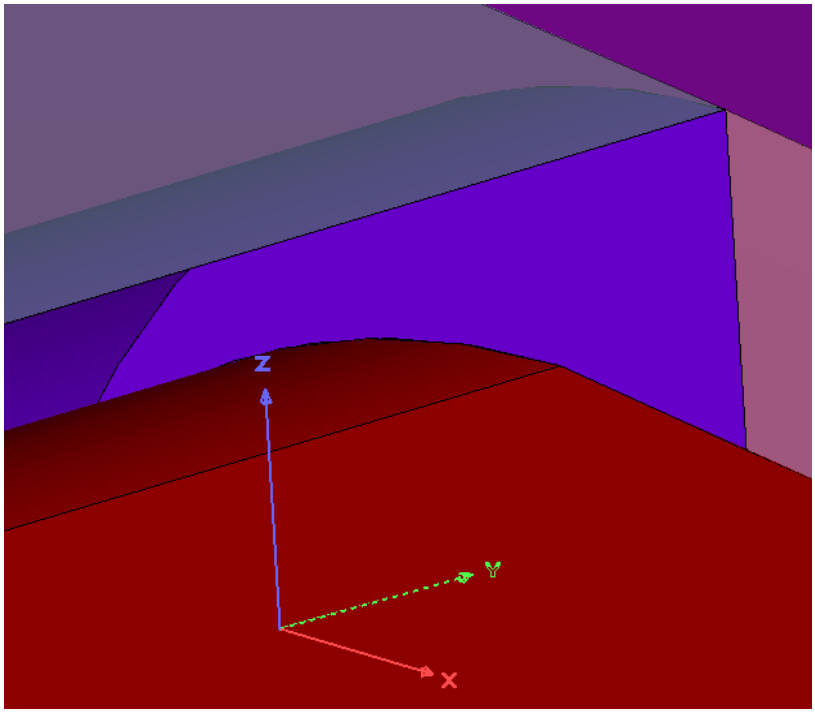


Figure 3.4

Close up of the Region Between the Elevon and the Wing cove

Propulsion

Propulsion boundary conditions were determined by computing the average flow values at the end of the isolator and integrating the quasi-1D equation for supersonic heat addition to find the average flow values exiting the combustor. The full equation to be integrated is shown below in equation 3.1 [14].

$$\frac{dM}{M} = \frac{1+\frac{\gamma-1}{2}M^2}{1-M^2}\left\{\frac{-dA}{A}+\frac{\gamma M^2}{2}\left[(\frac{4fdx}{D})+\frac{2\delta F_D}{\gamma M^2 pA}\right]+\frac{1+\gamma M^2}{2}\frac{dT}{T}+\left[(1+\gamma M^2)-y\gamma M^2\right]\frac{d\dot{m}}{\dot{m}}\right\} \tag{3.1}$$

If all these parameters are important to the problem at hand, then this equation may be integrated as is, as all variables can be written as functions of Mach number and x, the axial distance along the combustor. However, for the purpose of this research, friction has been neglected. For simplicity, it has also been assumed that γ is constant and that the fuel mass is incrementally added throughout the length of the combustor. Then equation 3.1 can be rewritten as

$$\frac{dM}{dx} = M\frac{1+\frac{\gamma-1}{2}M^2}{1-M^2}\left\{\frac{-dA}{dx}\frac{1}{A} + \frac{1+\gamma M^2}{2}\frac{dT}{dx}\frac{1}{T} + \left[(1+\gamma M^2) - y\gamma M^2\right]\frac{d\dot{m}}{dx}\frac{1}{\dot{m}}\right\} \tag{3.2}$$

where

$$\frac{dA}{dx} = 2\pi R\frac{R_B - R_A}{x_B - x_A} \tag{3.3}$$

$$\frac{dT}{dx} = \frac{(\frac{T_{0,B}}{T_{0,A}} - 1)}{1+(\theta-1)\chi}(\theta + \frac{\theta\chi(\theta-1)}{1+(\theta-1)\chi}) \tag{3.4}$$

$$T = 1 + \frac{T_{0,B}}{T_{0,A}}(\frac{\theta\chi}{1+(\theta-1)\chi}) \tag{3.5}$$

$$\frac{d\dot{m}}{dx} = \frac{\dot{m}_f}{x_B - x_A} \tag{3.6}$$

where variables with the A and B subscripts mark conditions at the beginning and end of the combustor respectively. $T_{0,A}$ can be determined directly from the average conditions at the inlet of the combustor. $T_{0,B}$ can be directly determined from equations 3.7 and 3.8 below.

$$q = \eta_b h_l \phi \psi_{st} \tag{3.7}$$

$$T_{0,B} = (\frac{q}{c_p} + T_{0,A}T_{inf})/T_{inf} \tag{3.8}$$

In equation 3.8, $T_{0,A}$ and $T_{0,B}$ are non-dimensional temperatures used by the flow solver. As it can be seen, equation 3.2 can now be written solely as a function of Mach number, x, and constants, and can be numerically integrated to determine the Mach number at the exit. Since Mach number and total temperature are known at the exit of the combustor, the rest of the required flow variables may be computed using the equations of the generalized 1D flow of a perfect gas.

Flow Solvers

Two different flow solvers were used, Cart3D and FUN3D. Cart3D [15] is a program developed by NASA Ames which solves the Euler equations on Cartesian grids. FUN3D [16] is a Navier-Stokes solver developed by NASA Langley. It has a wide range of capabilities but requires a mesh generated in other software, such as Pointwise [17].

Cart3D

Because Cart3D is an inviscid Cartesian solver, the entire simulation process from discretization to post-processing can be completed very quickly. This makes Cart3D the ideal candidate for generating the inviscid powered database, which involves hundreds of simulations. Although the total amount of work to create such a database is high, much of the process can be easily automated to reduce the burden on the user. Since producing the trimmed aerodynamic database involves creating a new geometry and mesh for each individual elevon rotation, an automated procedure was developed to ease the process of setting up the simulation for any given configuration.

For any meshing to occur, Cart3D first requires a watertight surface triangulation. Pointwise was chosen as the mesh generation software due to familiarity and its easy-to-use Cart3D exporter. For the creation of the trimmed database, the GHV has elevons that must be rotated. Since interactions in the gap between the elevon and the wing were not crucial for the inviscid aerodynamic database and it would take a large cell count to resolve, the cove between the elevon and the wing was modified by closing off the cove where the elevon sits. In order to automate the meshing process using Cart3D, special consideration must be given when creating the surface triangulation in Pointwise. First, the solver should be set to Cart3D, which should automatically change the solver dimension to 2D, meaning only a surface mesh may be created. Then, an extremely fine surface mesh, consisting of only triangles, was created all over the surface of the vehicle, including the elevons. In Cart3D's "cubes" command, a volume grid is created by recursively splitting cubes until the required level of volume refinement is reached. The geometry is then "cut out" of the volume mesh creating cut-cells. Because of this, the surface triangulation should be as fine as possible so the volume generation process has a high fidelity geometry to refer to, especially if small cell sizes are required. In other words, creating a fine surface mesh allows for the greatest flexibility over mesh size without affecting computational cost. Figure 3.5 shows how the gap between the elevon and the cove was modified for Cart3D.

After completing the surface triangulation, the surface patches must be oriented so that they all have outward-pointing normals. Then, each component of interest needs to be assigned its own tag. In this case, the body, combustor inlet, combustor outlet, nozzle, left elevon, and right elevon were each assigned as a unique component with its own ID. Finally, three Cart3D geometry (*.tri)

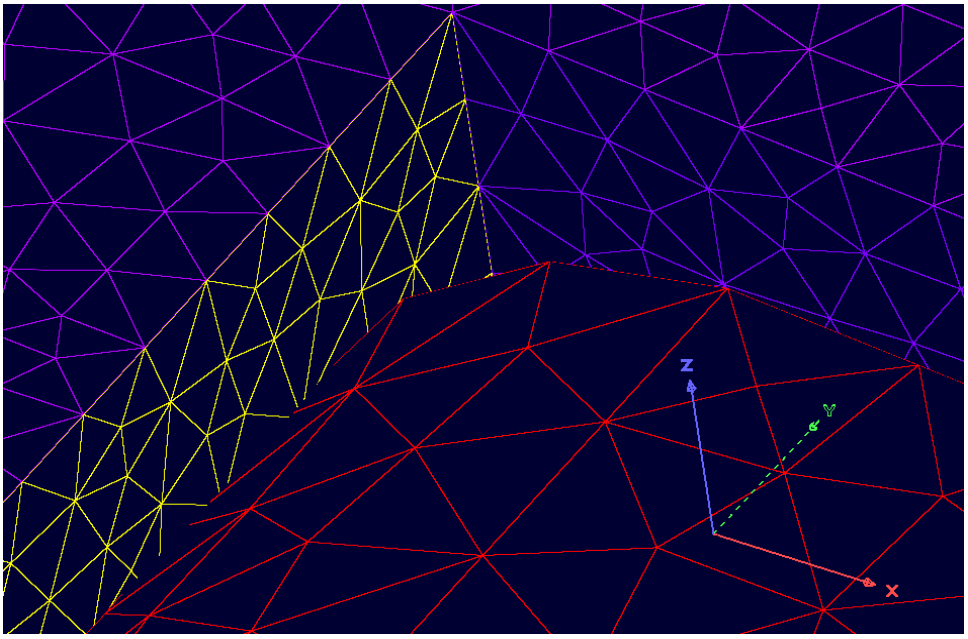


Figure 3.5

Intersection between the Elevon and the Patched cove; The Red Surface is the Elevon, the Blue/Purple is the Wing, and the Yellow is the Cove which was Closed Off

files were generated. The first consisted of all the surface patches besides those corresponding to the two elevons. The second and the third files contained the patches from each of the two elevons respectively. Then the elevons may be rotated to the appropriate deflection using a script or built-in commands for rotating components from the newest version of Cart3D (v1.5.7 or newer). Finally, the elevons and body were intersected into the final watertight geometry using a combination of the Cart3D commands trix, comp2tri, and intersect. The full procedure may be found in the appendix.

Boundary conditions in Cart3D are extremely simple. The simulation was first run once with arbitrary propulsion boundary conditions per combination of Mach number, angle of attack, and equivalence ratio. Then, the solution was read into a post-processing program such as Tecplot or FieldView. From there, primitive variables (density, axial velocity, and pressure) were averaged

over the face of the inlet of the combustor. Then, the appropriate exit boundary conditions were computed using the procedure provided in the propulsion section.

Cart3D is shipped with a very robust adjoint-based adaptation feature. The whole adaptation process is automated and driven by a script provided by the Cart3D distributor which can be modified to use settings appropriate to the user's specific problem. For example, the user is able to specify the number of adaptation cycles as well as how many iterations of the flow solver to run in each cycle. Adaptation was used anytime a Cart3D case was run, as it provides the ability to only use points where they are needed to resolve the important flow features. Cart3D's adaptation script is able to refine around areas of high gradients in density and velocity, in addition to its shock capturing feature. An important requirement of the initial mesh used for the adaptation process is the resolution of the most important flow features; otherwise, the flow will have no features to which to adapt. Since we possess some a priori knowledge of the flow features that will be present in the vehicle's internal flow path, we can build an adequate mesh to ensure those features are captured. In addition, the first step of the adaptation cycle is to move the initial mesh points to where they are needed most, reducing the bias of the original mesh. Figure 3.6 shows the initial mesh compared to the adapted mesh for a Mach 6 case with no angle of attack.

It can clearly be seen that the mesh has been adapted to pick up the expected flow features. This type of refinement would not be possible without adaptation, not only due to the complexity of the flow, but it is also outside of the capability of Cart3D's "cubes". The alternative approach would be to use a single extremely refined mesh to ensure all flow features get resolved. Figure 3.7 shows the adapted solution at the symmetry plane compared to the solution on a mesh which was

created by refining the entire flow path. The uniformly adapted mesh contains 1.7M cells, whereas the adapted mesh is much larger, at 30.8M cells. This large disparity in mesh size indicates that although the mesh with uniform refinement seemed like it would be sufficient, the adapted solution required many more points to reach a converged solution. This is another motivation for adaptation, it removes the need for a user's judgment and instead makes sure points go where they are needed.

From Figure 3.7, it can be seen that the adaptation produces an extremely crisp solution and picks up many flow features which are not present in the un-adapted solution. Table 3.1 compares the mesh size, time to solution, C_x, and C_y. As mentioned in the section on the boundary conditions, each data point was run twice, the first time to determine the correct inlet conditions, and the second to use the appropriate thrust boundary condition to determine C_x and C_y. Both solutions produce roughly the same C_y, but the un-adapted solution was not able to produce the correct thrust. This is due to its insufficient resolution of the flow field in the inlet and isolator, which leads to incorrect propulsion boundary conditions which in turn leads to the incorrect prediction of drag. For this reason, adaptation was used on all Cart3D runs, as it allows the solver to decide where points are needed the most to achieve convergence. The solution was adapted until the force coeffecients were no longer changing.

Table 3.1

Comparison of Adapted and Un-Adapted Results

Solution	Final Mesh Cell Count	CPU Time	C_x	C_y
Un-adapted	1,710,162	17m	-50.02	238.39
Adapted	30,840,330	188m	-36.50	239.48

Figure 3.8 show the variation of non-dimensional pressure on the symmetry plane due to the Mach number. The GHV was designed for the "shock-on-lip" condition at Mach 6. Therefore, at Mach 5 the shock extends below the lip, which produces some spillage drag. At Mach 7, the shock hits the inlet past the lip, resulting in a very different set of reflected shocks from the other two cases. Figure 3.9 shows the variation of non-dimensional pressure at the combustor inlet due to incoming Mach number. In addition, the conditions at the inlet to the combustor are sensitive to the angle of attack, as shown in Figure 3.10. Furthermore, pressure is not the only variable that is a function of the angle of attack. The average primitive flow variables at the inlet used to determine propulsion conditions, also vary strongly with the angle of attack, as shown in Table 3.2. Changing the angle of attack changes the angle of the shock at the inlet, which in turn changes the shock interactions in the inlet and isolator, which produces these large changes in average conditions. These changes in inlet conditions can cause thrust to vary significantly with the angle of attack. This further highlights the need to determine conditions at the inlet to the combustor for all points in the database.

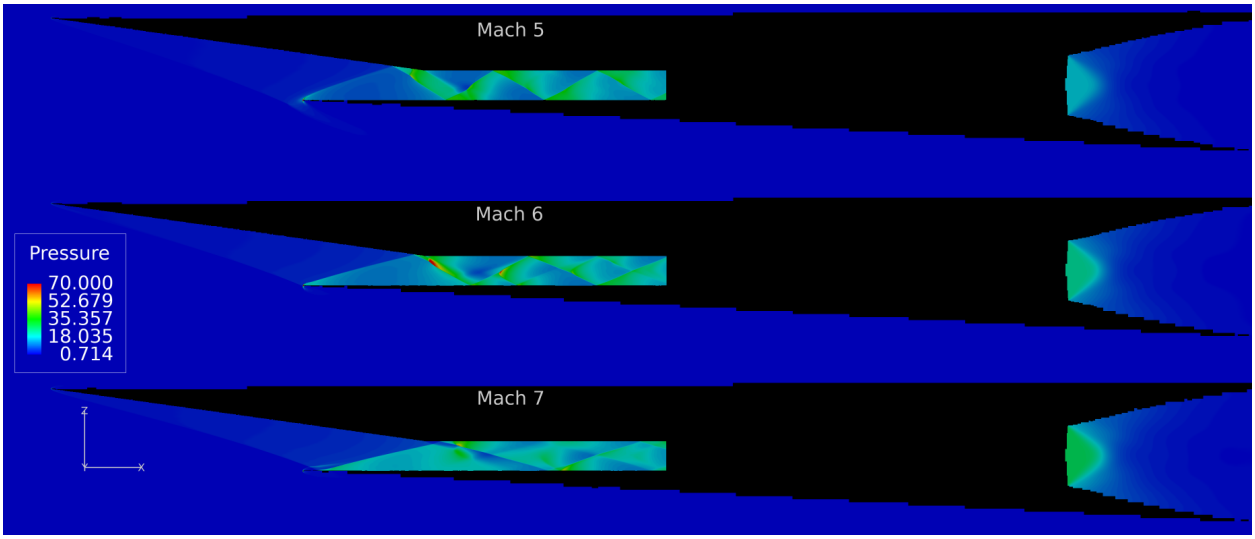


Figure 3.8

Variation of Pressure along Symmetry Plane due to Varying Mach Number

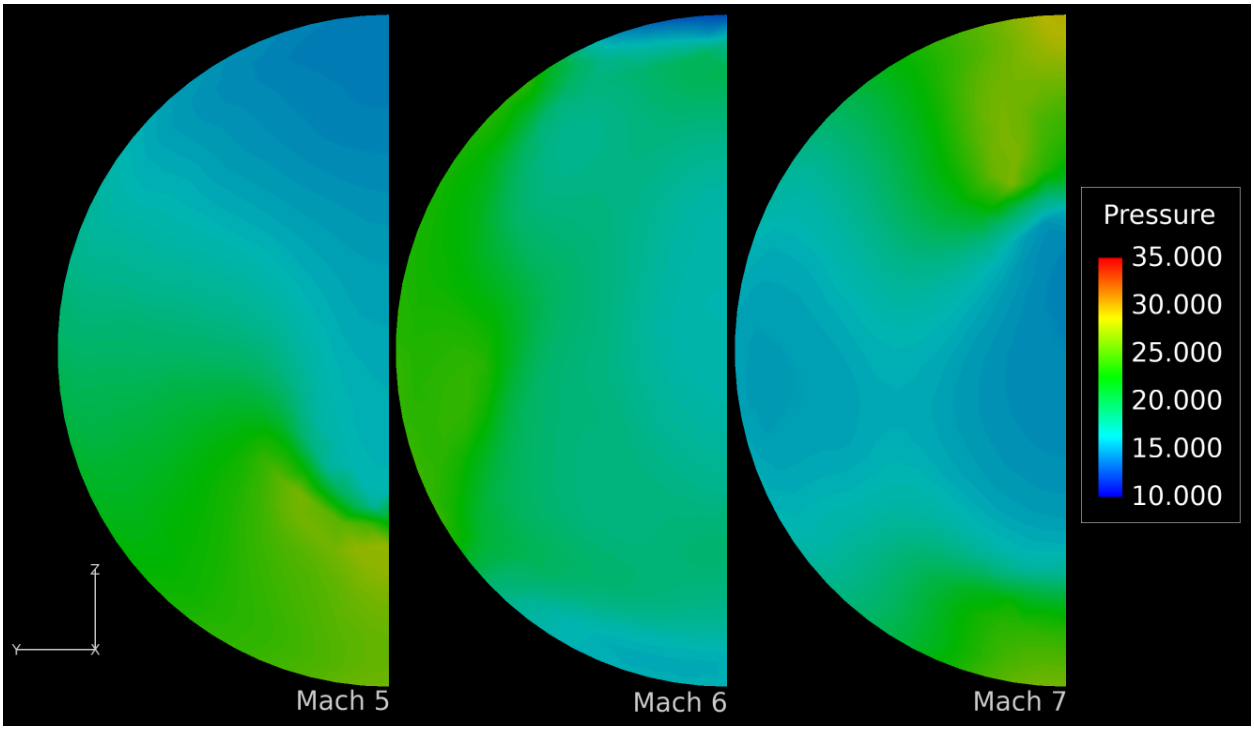


Figure 3.9

Variation at Entrance of Combustor due to Varying Mach Number

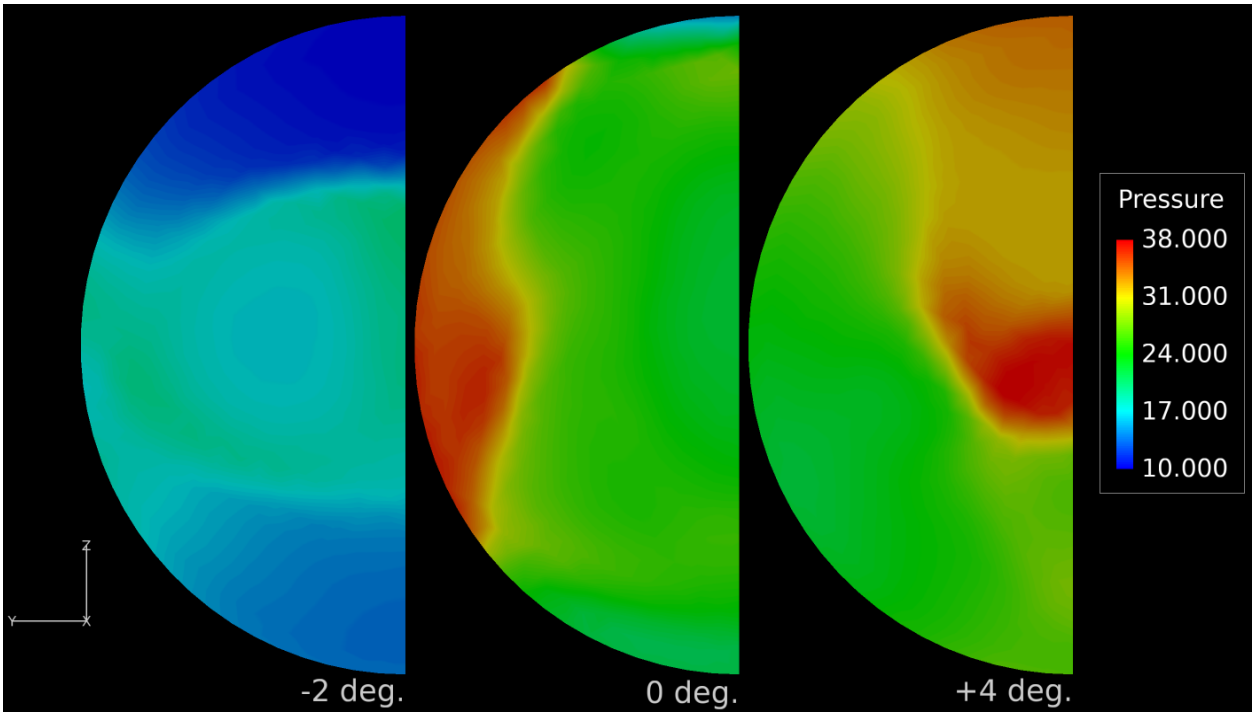


Figure 3.10

Variation of Pressure on the Inlet at Mach 6 due to Angle of Attack

Table 3.2

Primitive Variables as a Function of Angle of Attack

AoA	Density	X-Velocity	Pressure
-4	7.085	5.307	12.35
-2	8.143	5.236	15.77
0	9.203	5.145	19.01
2	10.30	5.030	23.06
4	11.47	4.877	28.20

The powered database was trimmed with respect to the pitch axis. For each combination of Mach number, angle of attack, and equivalence ratio, the case was run once to compute the average inlet conditions. These were then input into the quasi-one-dimensional propulsion code to compute the average exit conditions. These were then imposed at the exit to the combustor. From there, the simulation was repeated with the elevon rotated to various angles until zero pitching moment was bracketed. From there, the trim angle was linearly interpolated. The test matrix of the trimmed conditions may be found in Appendix A. Each solution was run using Cart3D's aero.csh script, which guides mesh adaptation until solution convergence. The solution was started on a coarse grid, with 8 levels of refinement, and was adapted 7 times. In each adaptation cycle, the number of iterations run was equal to the number of iterations run on the previous adaptation cycle plus 200.

The cases were run with three levels of multigrid, which help smooth out low-frequency errors and aid convergence. A CFL of 1 was used for all of the adaptation cycles. In addition ”robust mode” was found to help the stability of the solutions greatly. This was activated by setting Cart3D to reevaluate the gradient and limiter at all steps of the Runge-Kutta scheme. In all, over 400 inviscid simulations were run.

FUN3D

Selected cases from the inviscid analysis were chosen to be run in the viscous solver, NASA's FUN3D [16]. FUN3D is a fully unstructured node-centered finite volume flow 3D solver which is capable of running inviscid, laminar, and fully turbulent solutions. All FUN3D simulations in this analysis were run turbulent. Viscous meshes were generated in Pointwise [17], with viscous layers thin enough to resolve the thermal boundary layer, which has been shown to be smaller than the velocity boundary layer in comparable hypersonic situations. The spacing off the wall was around 1.8E-06in, resulting in a maximum y+ of around 0.1. Since the cove was not closed off for the FUN3D solutions, additional care was put into resolving the region between the elevon and the wing of the vehicle. Figures 3.11 and 3.12 show these portions of the mesh. The viscous mesh had approximately 29 million points due to the resolution of the thermal boundary layer and elevon gap. The mesh statistics are summarized below in Table 3.3

Table 3.3

Typical Viscous Mesh Statistics

Element Type	Count
Nodes	28,981476
Triangles	836,662
Quadrilaterals	332,977
Tetrahedrons	28,665,249
Pyramids	857,260
Prisms	35,378,783
Hexahedrons	5,996,265
Total Cells	72,067,196

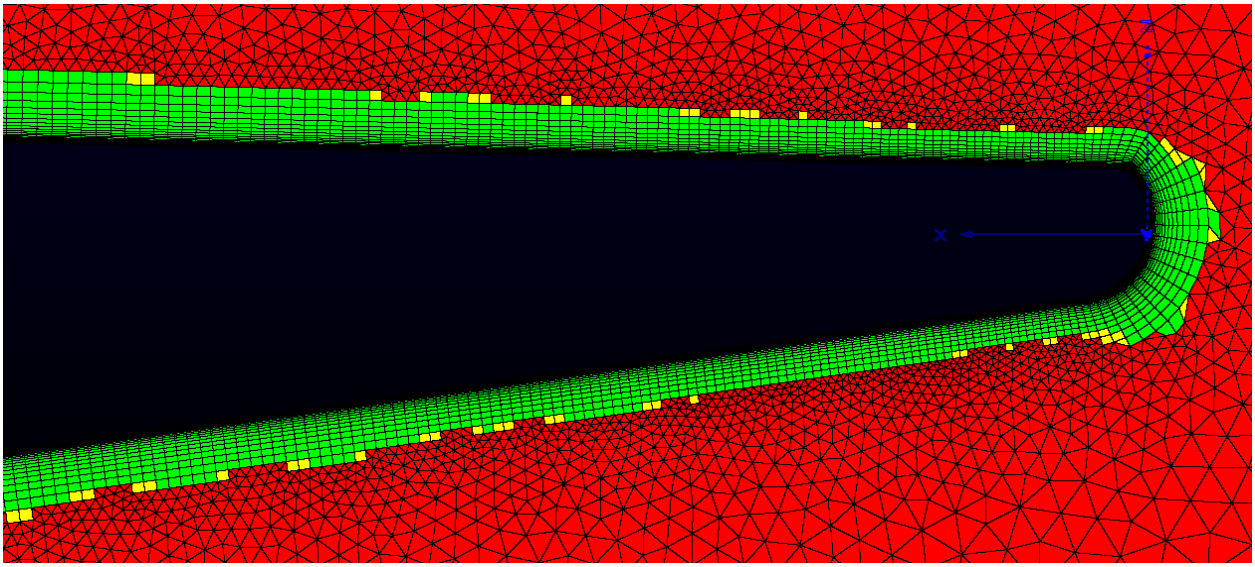


Figure 3.11

Hex Elements Resolve the Boundary Layer on the Nose of the GHV

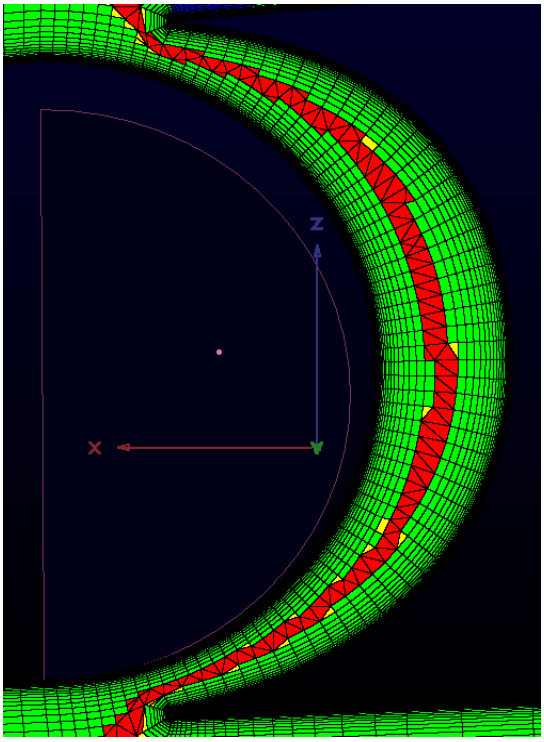


Figure 3.12

The Area between the Elevon and the Wing is Resolved

All viscous cases were run with the Spalart-Allmaras turbulence model. In addition, the LDFSS [18] flux with adaptive entropy fix was used due to the presence of the "carbuncle effect" in the other formulations tried. The limiter used was 'hvanleer,' a stencil based limiter which also takes into account a heuristic based pressure limiter. Because of the viscous nature of the solutions, flow features show up which do not exist in the inviscid solutions. Figure 3.13 shows a comparison of the inviscid and viscous solutions at Mach 6 with an angle of attack of -4 degrees. In the viscous solution, the presence of the boundary layer can be seen, as well as separation where the x-velocity goes negative. Figure 3.14 shows non-dimensional pressure for the same case. There are a few

things of note in these solutions. First, in the viscous solution, the separation drastically changes the nature of the reflected shocks in the solution. In addition, the shocks are much more diffuse compared to those of the adapted inviscid solution. This may be due to the adaptation producing a very crisp solution, the diffusive nature of the viscous solution, or a combination of the two. Figure 3.15 shows the difference in grid resolution between the two solutions. Unfortunately, the inviscid level of refinement is currently not feasible, with the viscous mesh already possessing 29 million nodes. If more time and computational resources were available, future work should include a grid refinement study to examine the resolution required to determine accurate conditions at the inlet to the combustor. In addition, the viscous solution could greatly benefit from mesh adaptation if possible.

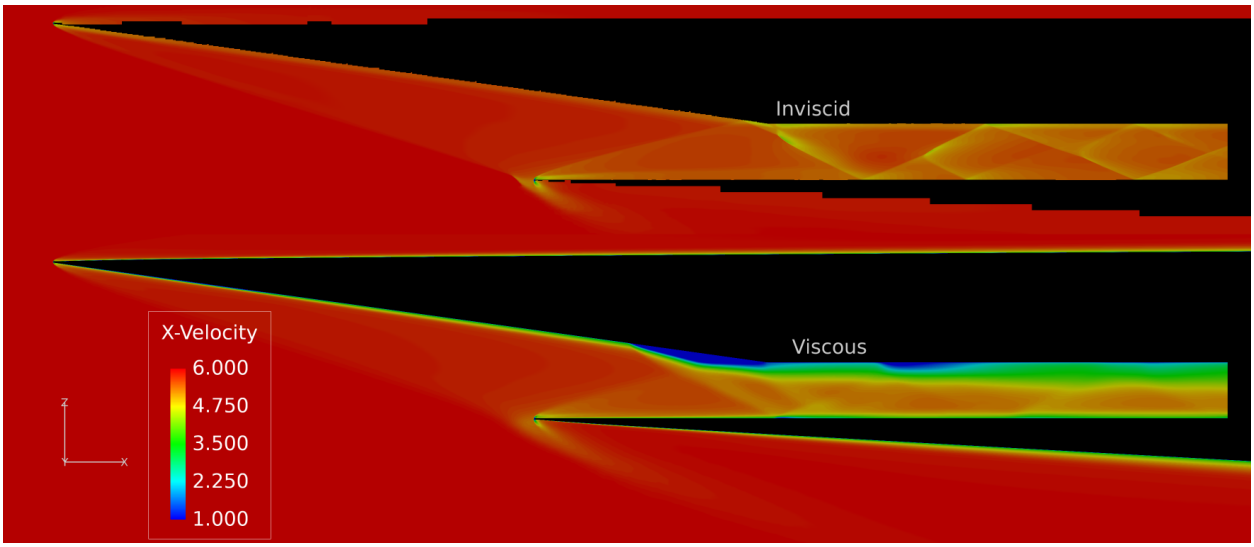


Figure 3.13

Separation which is Absent from Inviscid Solutions can Appear in Viscous Solutions

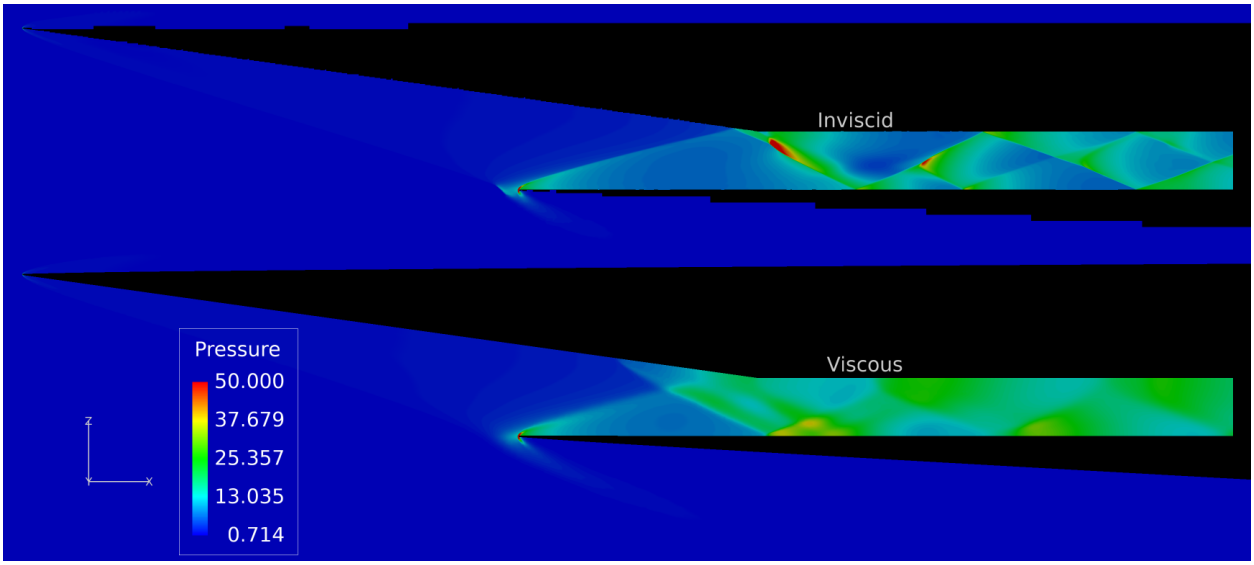


Figure 3.14

Inviscid vs. Viscous Non-dimensional Pressure at M6 and -4 degrees of Angle of Attack

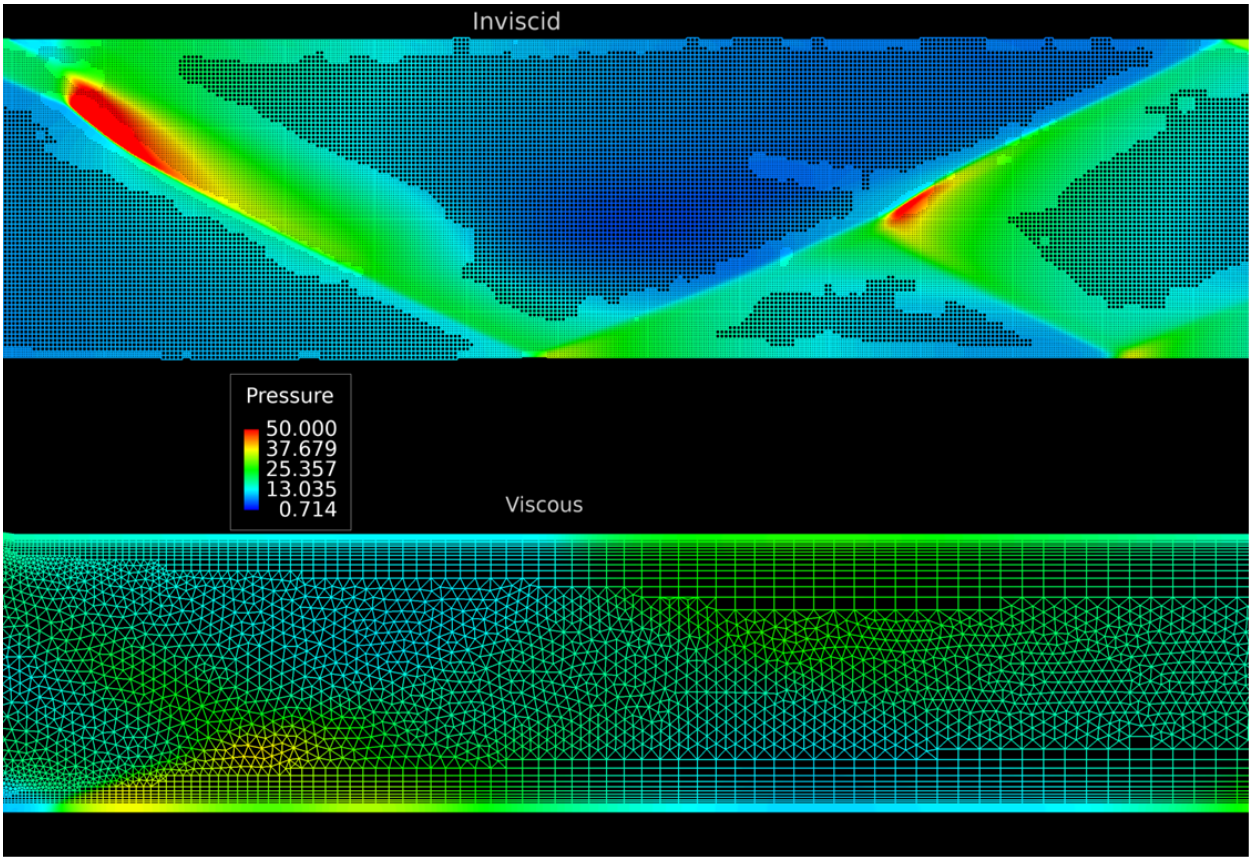


Figure 3.15

Inviscid vs. Viscous Mesh Fidelity in the Isolator

Each viscous case was run at a constant surface temperature, i.e., isothermal; however, the cases could easily be run using the adiabatic wall temperature or fully adiabatic. The surface temperatures used in the viscous cases were arbitrary but were chosen in an attempt to bracket the possible surface temperatures experienced by a hypersonic vehicle operating at the specified Mach number. The surface temperatures can have a significant effect on the solution, and in some cases change its nature entirely. Figures 3.16, 3.17, and 3.18 show viscous solutions on the symmetry

plane. The potentially drastic effects of varying surface temperature on the nature of the solution are shown in Figure 3.16. At a surface temperature of 353K, the solution looks as expected. There is a leading oblique shock and a shock train down the rest of the inlet and isolator. However, when the surface temperature is held at a constant 949K, the inlet unstarts, and a normal shock and a bow shock appear in the inlet to the engine. This causes the flow to go subsonic locally. This means the flow will now react to the back pressure being imposed at the inlet to the combustor. For all the simulations, this back pressure was set to an arbitrarily low pressure, as it was not expected to affect the solution in most cases. However, in this case, the subsonic flow will begin to accelerate again and the solution is no longer representative of a realistic flight condition. The unstart, in this case, is most likely due to a few factors. First, the incoming flow is at Mach 5, which is below the design Mach number of 6. This means the engine is at an off-design dynamic pressure. The aircraft is also at a high angle of attack where the shocks are stronger. Furthermore, since the surface temperature is so high, heat is being added to the flow. This could lead to thermal choking which would cause the shock train to exit the inlet and form a normal shock. Since this inlet is fully 3D and the geometry is complicated, a conventional normal shock is not formed. Instead, the shock looks like a combination of a normal and bow shock on the centerline of the vehicle. The engine also demonstrates a similar unstart behavior when run with an adiabatic wall, and the effects of surface temperature on the solution warrant further investigation. Figures 3.17 and 3.18 shows the more standard effects of the surface temperature on the solution. The difference between these solutions and the unstarted solutions is most likely due to the increased mach number and lower angles of attack.

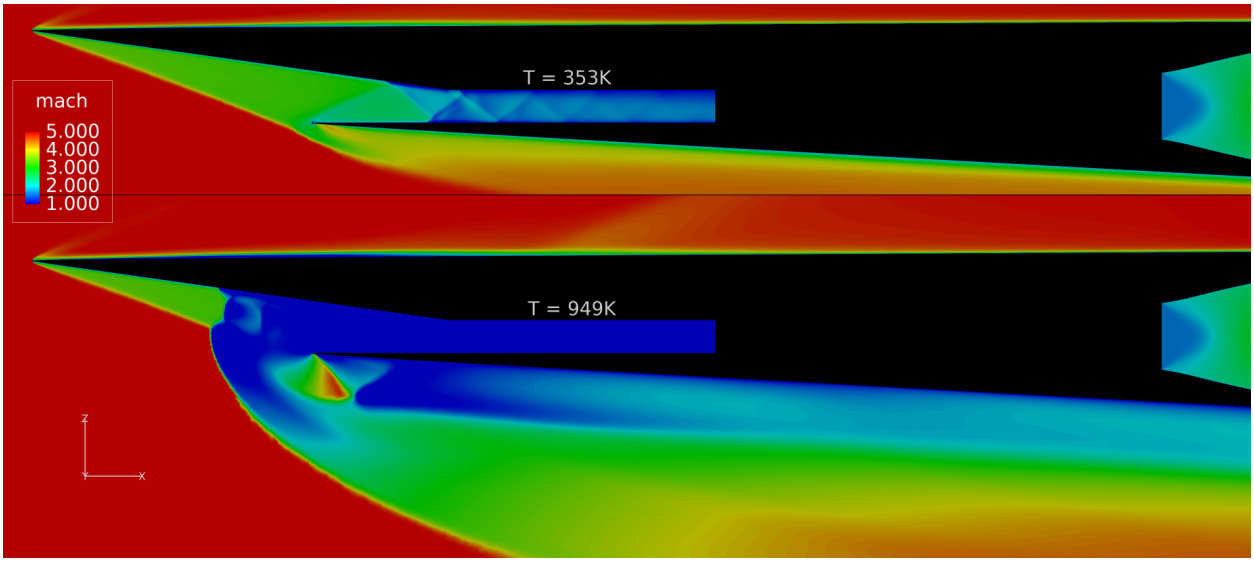


Figure 3.16

Viscous Simulation at Mach 5 and 6 Degree Angle of Attack at Varying Surface Temperatures

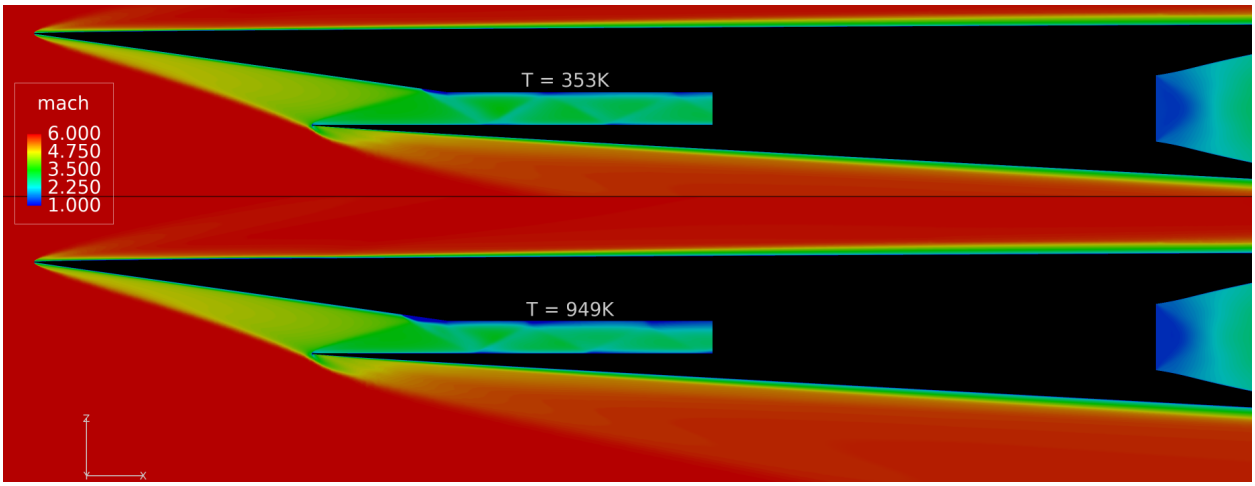


Figure 3.17

Viscous Simulation at Mach 6 and 0 Degree Angle of Attack at Varying Surface Temperatures

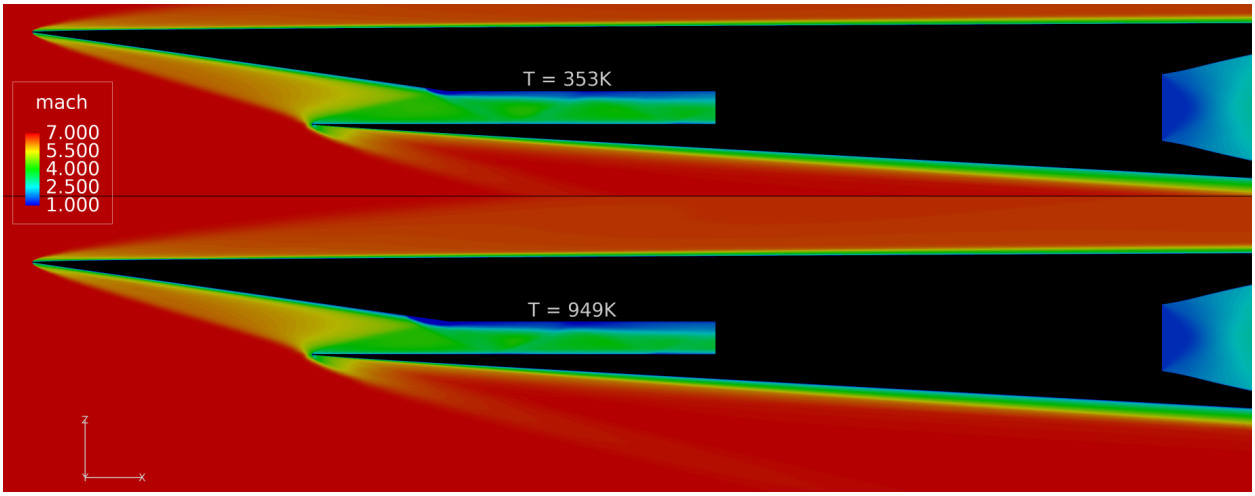


Figure 3.18

Viscous Simulation at Mach 7 and -4 Degree Angle of Attack at Varying Surface Temperatures

As seen in Figures 3.17 and 3.18, the increased surface temperature does not overly affect the solution. The same standard shock train is seen, with slightly increased separation on the top and bottom of the isolator due to the increased temperature. Even though the nature of the solutions does not drastically change in the more benign cases, the conditions at the inlet of the combustor can still change significantly.

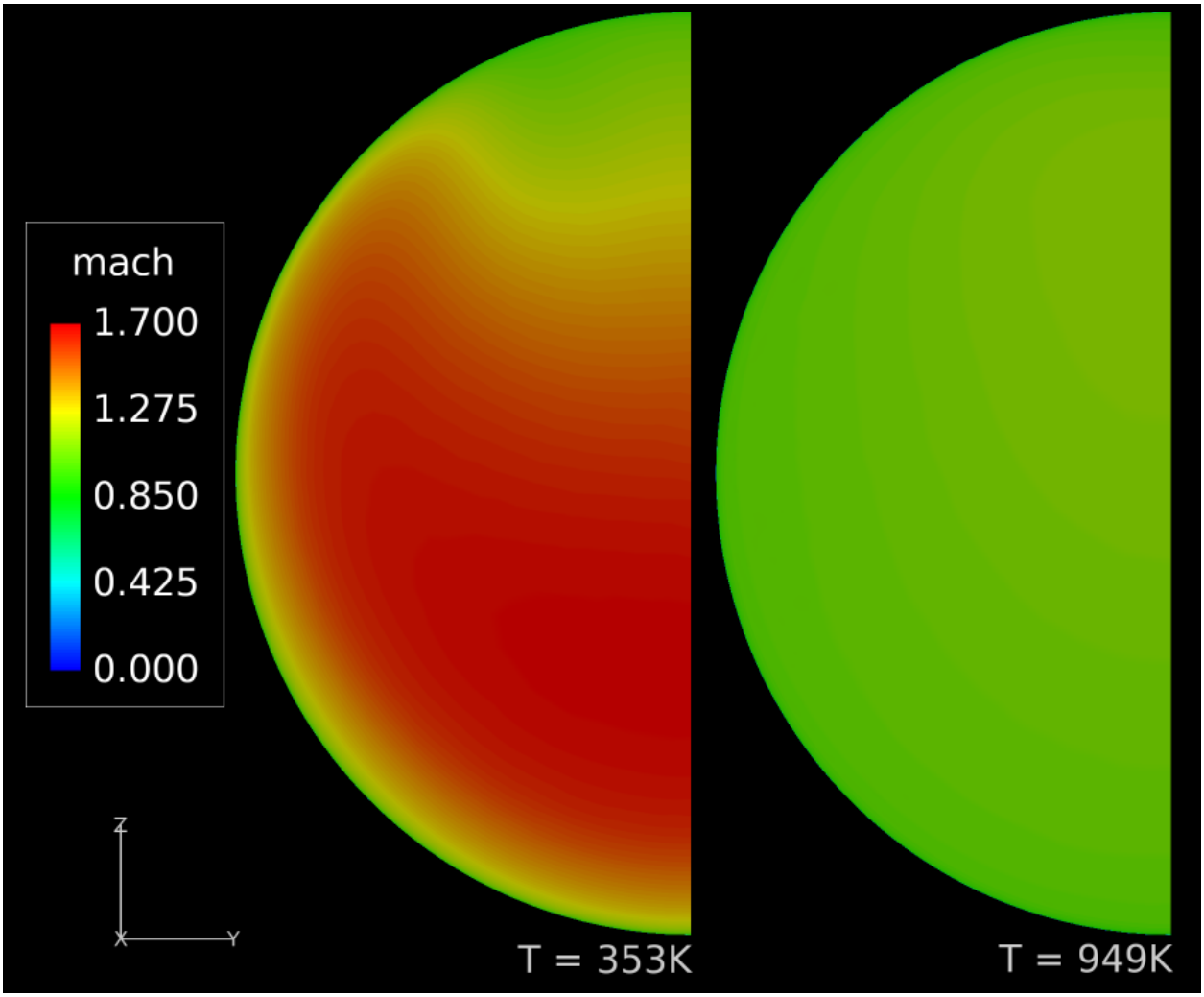


Figure 3.19

Inlet Mach number at Mach 5 and 6 degree Angle of Attack at Varying Surface Temperatures

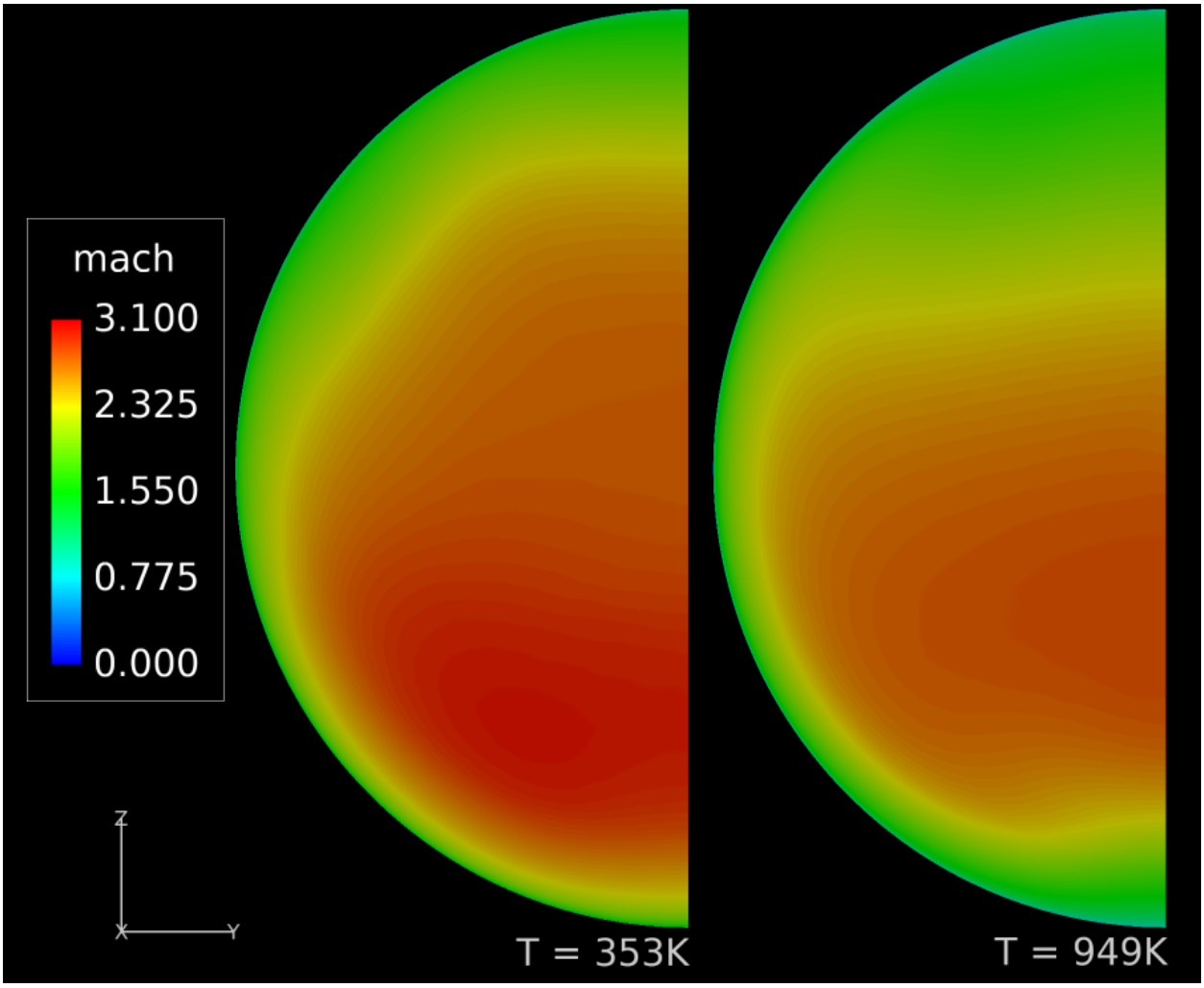


Figure 3.20

Inlet Mach Number at Mach 6 and 0 Degree Angle of Attack at Varying Surface Temperatures

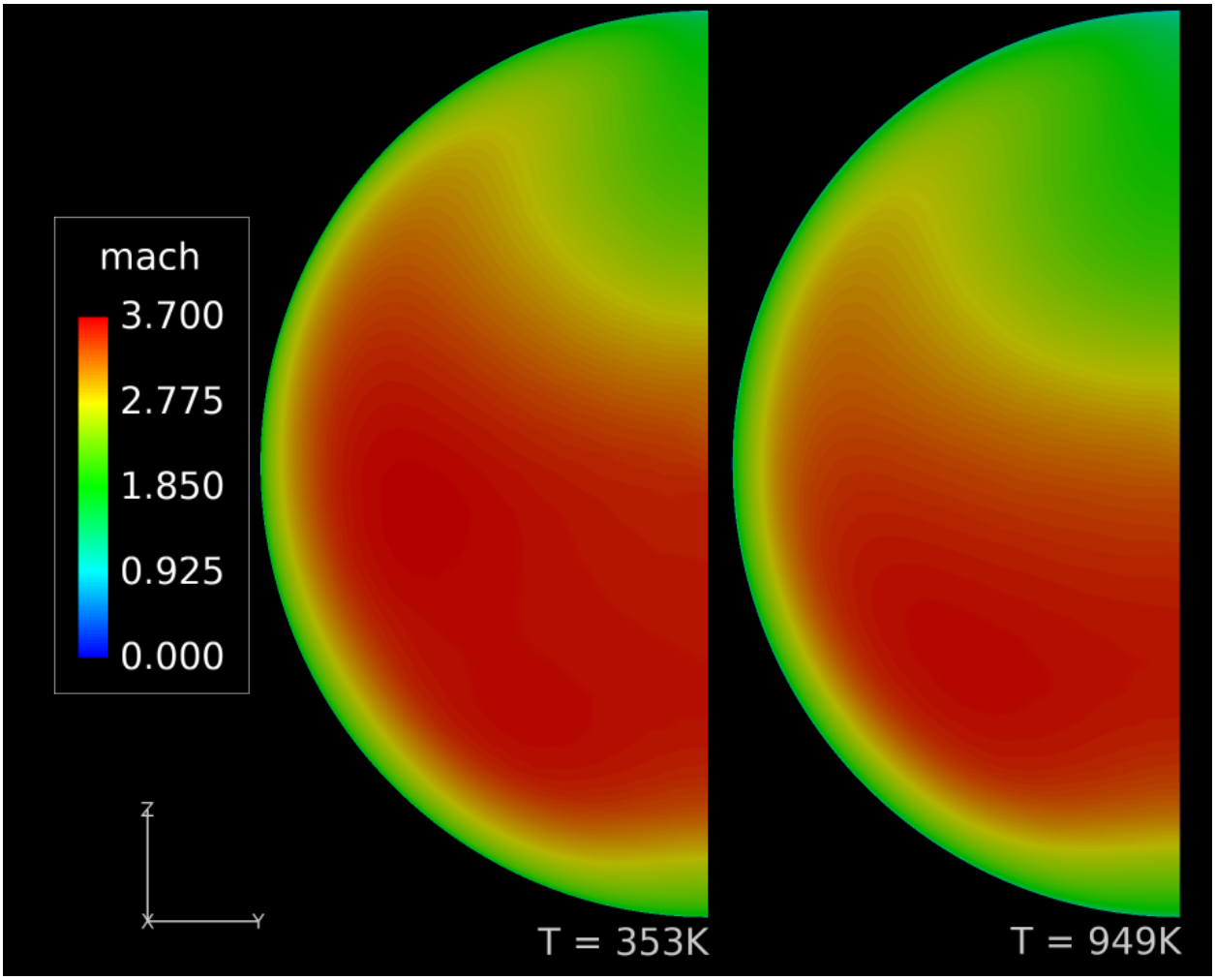


Figure 3.21

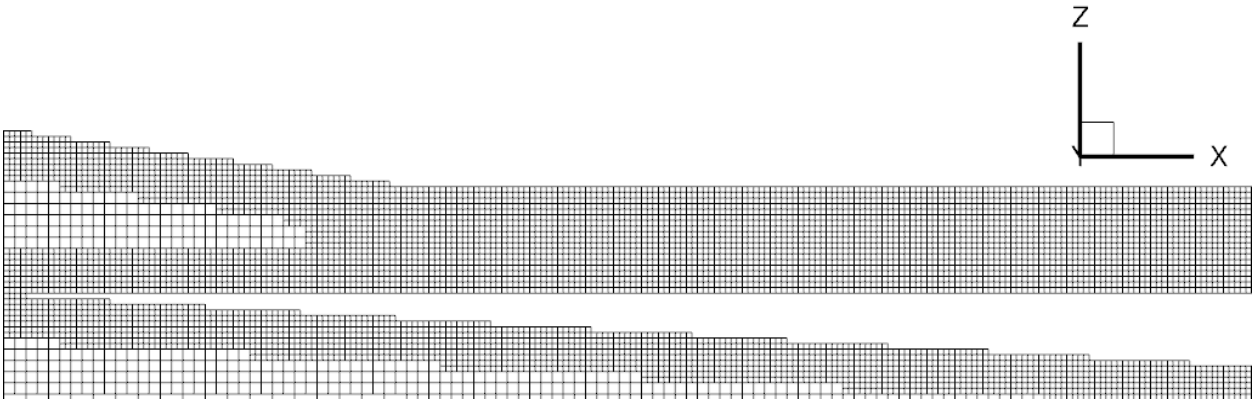
Inlet Mach number at Mach 7 and -4 Degree Angle of Attack at Varying Surface Temperatures

As expected, in Figure 3.19, the Mach number at the inlet is completely different due to the unstart condition. Figures 3.20 and 3.21 show that even when the surface temperature does not change the nature of the solution, it can affect the angle with which the shock strikes the inlet, resulting in different average flow variables, as shown in Table 3.4.

Table 3.4

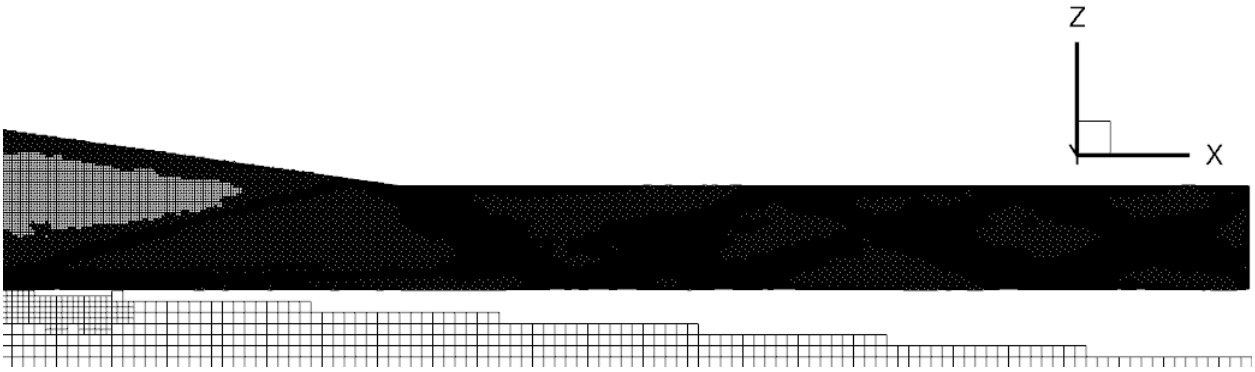
Effects of Average Quantities at the Inlet of the Combustor due to Varying Surface Temperature

	Surface Temperature – 353K			Surface Temperature – 950K		
Case	**Density**	**X-Velocity**	**Pressure**	**Density**	**X-Velocity**	**Pressure**
Mach = 5 A.O.A = 6	16.34	2.905	44.93	8.62	2.196	28.21
Mach = 6 A.O.A = 0	10.17	4.580	24.30	10.29	4.55	27.67
Mach = 7 A.O.A = -4	7.64	5.689	18.17	7.70	5.605	20.32



(a)

Unadapted Mesh

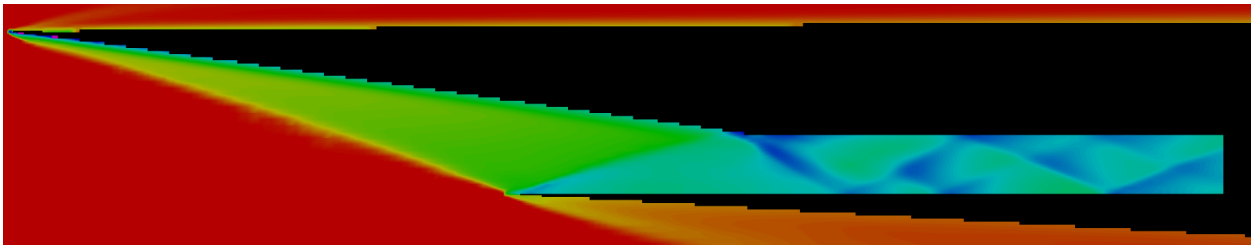


(b)

Adapted Mesh

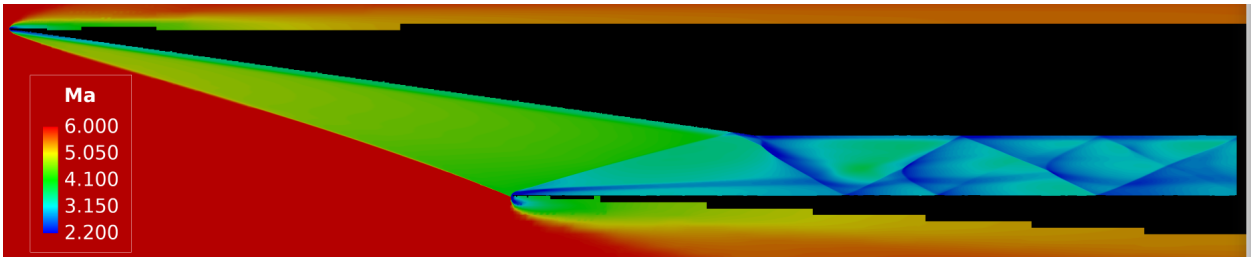
Figure 3.6

Mesh Before and After Adaptation



(a)

Before Adaptation



(b)

After Adaptation

Figure 3.7

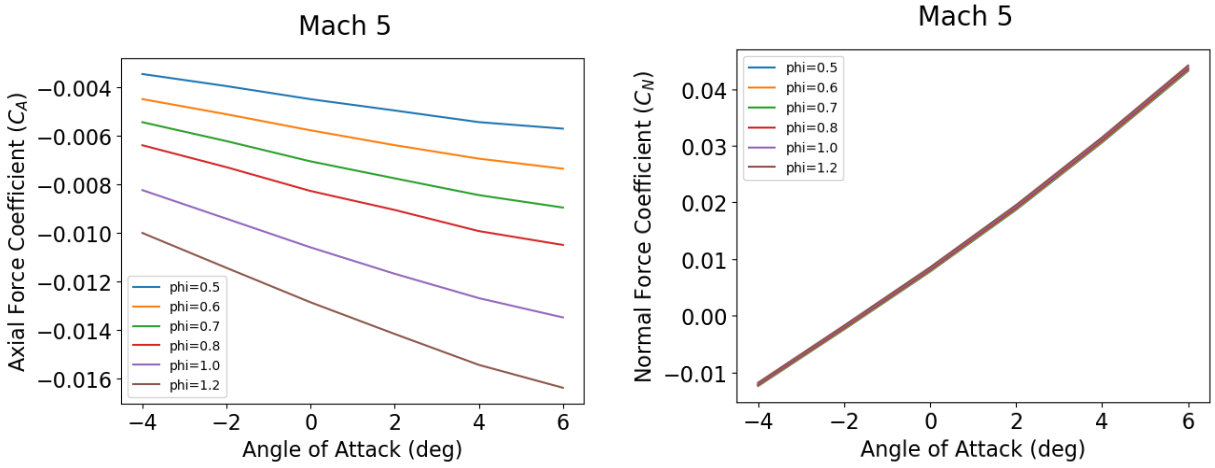
Mach 6, Zero angle of Attack Solution Before and after Adaptation

CHAPTER IV

RESULTS AND DISCUSSION

Inviscid Database

Plots of axial and normal force coefficients as a function of angle of attack and equivalence ratio (ϕ) are shown below for each Mach number in Figures 4.1, 4.2, and 4.3 where a negative axial force coefficient represents a net thrust and a positive normal force coefficient is oriented towards the top of the vehicle. The vehicle produced a net thrust in all cases. In addition, all cases followed a similar trend, with axial force increasing with increasing angle of attack and equivalence ratio. In addition, the normal force increased almost perfectly linearly with the angle of attack for all equivalence ratios at all Mach numbers.

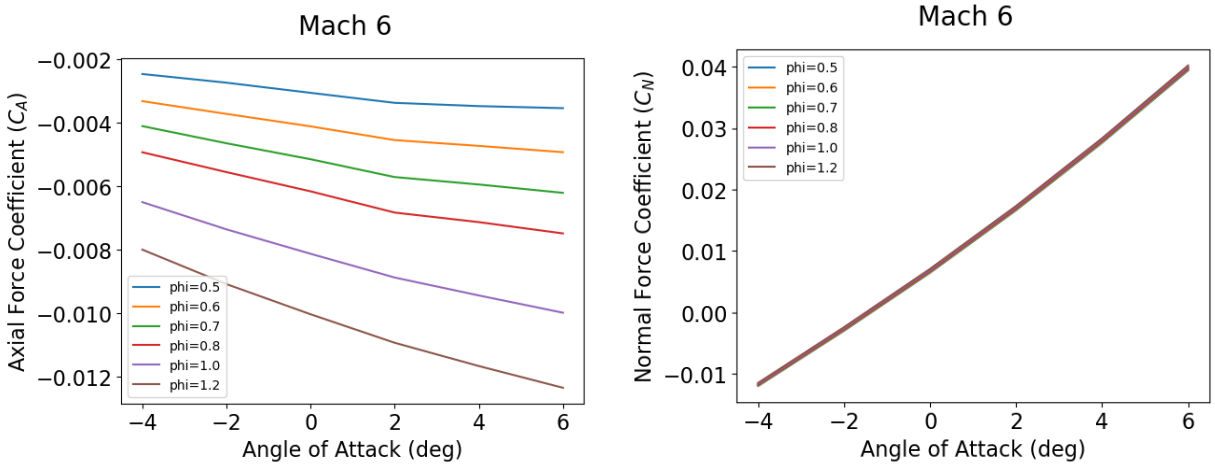


(a) (b)

Axial Force Coefficient Normal Force Coefficient

Figure 4.1

Axial and Normal Force Coefficients at Mach 5 as a Function of Angle of Attack and Equivalence Ratio (ϕ)

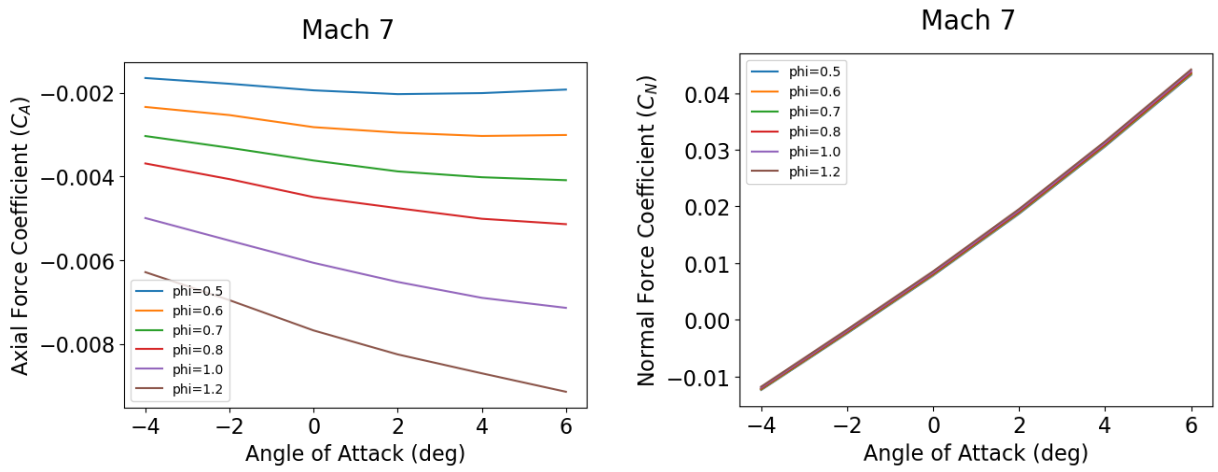


Axial Force Coefficient

Normal Force Coefficient

Figure 4.2

Axial and Normal Force Coefficients at Mach 6 as a Function of Angle of Attack and Equivalence Ratio (ϕ)



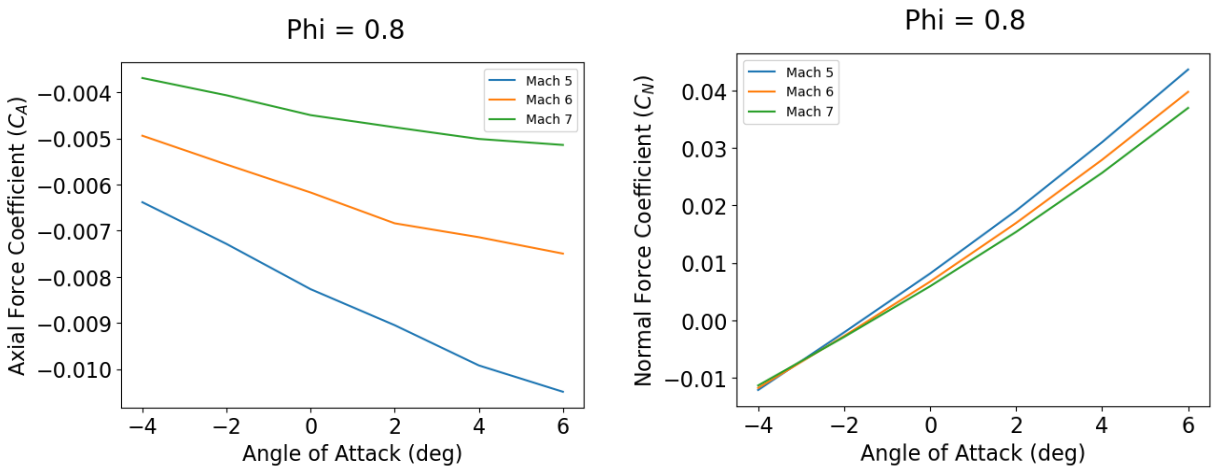
Axial Force Coefficient

Normal Force Coefficient

Figure 4.3

Axial and Normal Force Coefficients at Mach 7 as a Function of Angle of Attack and Equivalence ratio (ϕ)

Figure 4.4 shows the variation of axial and normal force coefficients as a function of angle of attack at different Mach numbers at an equivalence ratio of 0.8. The vehicle has the highest axial force coefficient at Mach 5. The normal force coefficient does not vary significantly between Mach numbers.



Axial Force Eoefficient

Normal Force Coefficient

Figure 4.4

Axial and Normal Force Coefficients at $\phi = 0.8$ as a Function of Mach Number and Angle of Attack

The full set of data including the trimmed elevon position may be found in the appendix.

Viscous Solutions

Figures 4.5, 4.6, and 4.7 show the results of the selected viscous solutions compared to the solutions from the inviscid database. All three cases demonstrate the expected behavior, similar to [5] where the viscous solutions predict less of an axial force than the inviscid solutions. In contrast, the normal force coefficient differs very little from the inviscid solution. The pitching moment for the inviscid cases was exactly zero due to the conditions being interpolated. The viscous solutions,

which use the same elevon deflection as the trimmed inviscid solutions do not show a zero moment, as seen in figure 4.7.

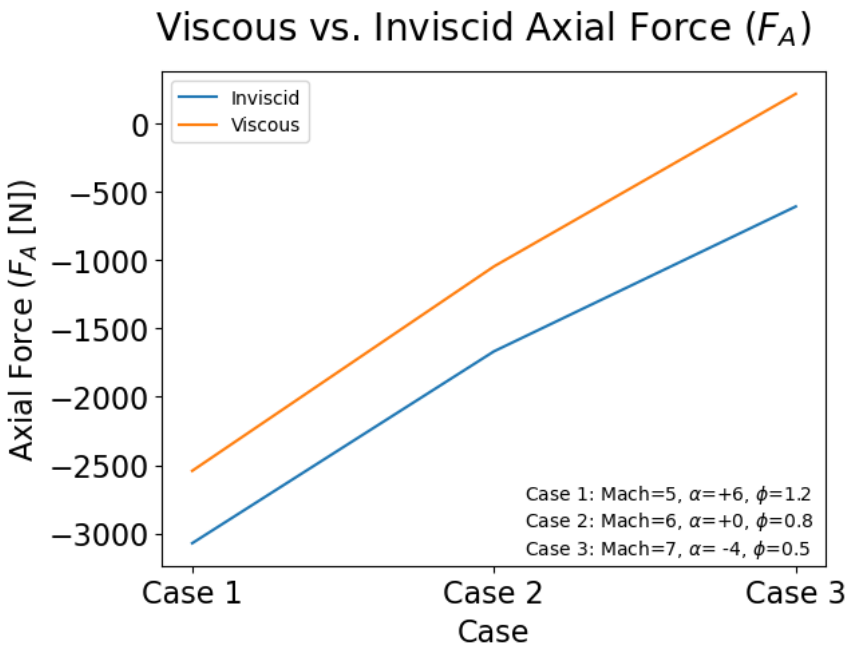


Figure 4.5

Axial Force for Viscous and Inviscid solutions

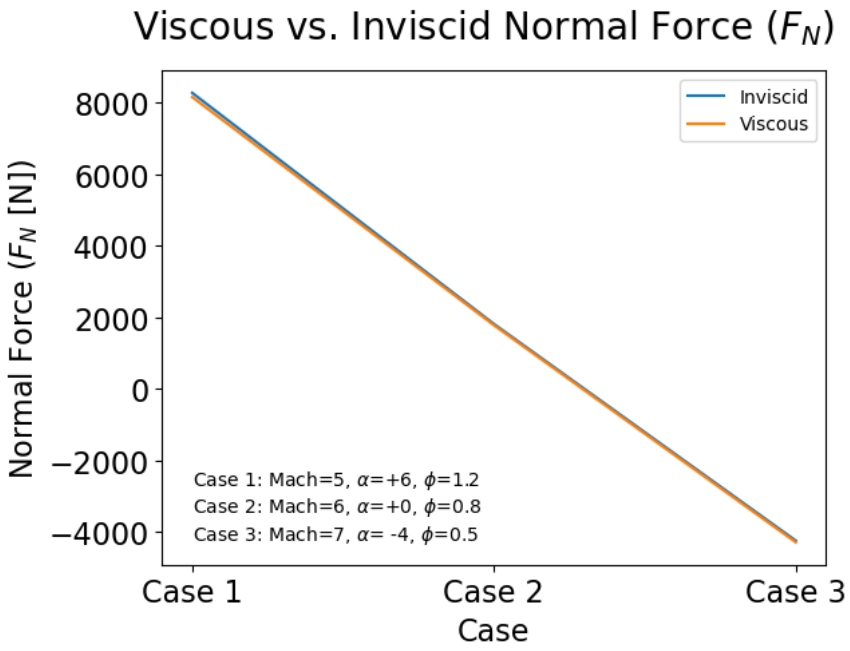


Figure 4.6

Normal Force for Viscous and Inviscid Solutions

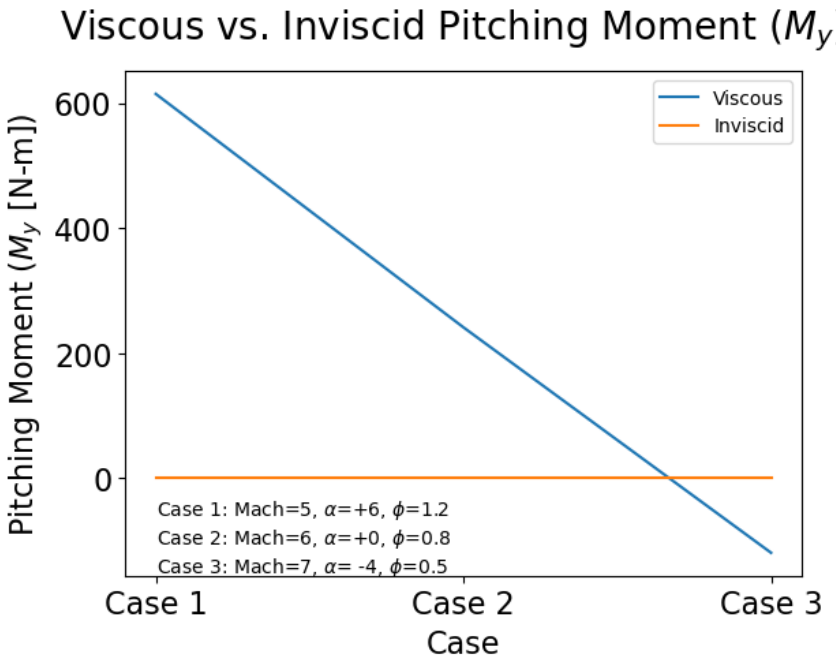


Figure 4.7

Pitching Moment for Viscous and Inviscid Solutions

The underprediction of axial force in the inviscid solution is a function of both the presence of viscous forces and the differing boundary conditions at the inlet to the combustor. The presence of viscous forces will always decrease the forward axial force on the vehicle when compared to the inviscid solution. The impact of the different boundary conditions due to the more complicated flow physics is less certain, however. In fact, in all three viscous cases used for this study, the boundary conditions of the viscous solutions actually increased the thrust of the engine. For example, in the Mach 6 case, the axial force coefficient was -0.00617 for the inviscid case. In the viscous case. the axial force coefficient was only -0.00387, slightly more than half. However,

if the viscous forces reported by FUN3D are removed from the viscous axial force coefficient, it becomes -0.00677, indicating the loss in axial force coefficient is solely due to the viscous forces. In essence, the decrease in axial force due to viscous shear stresses is slightly offset by an increase in force from the engine. This trend occurs in all 3 cases and is worthy of future investigation.

Figures 4.8, 4.9, and 4.10 show the values of Mach number on the symmetry plane of the three viscous cases which were run.

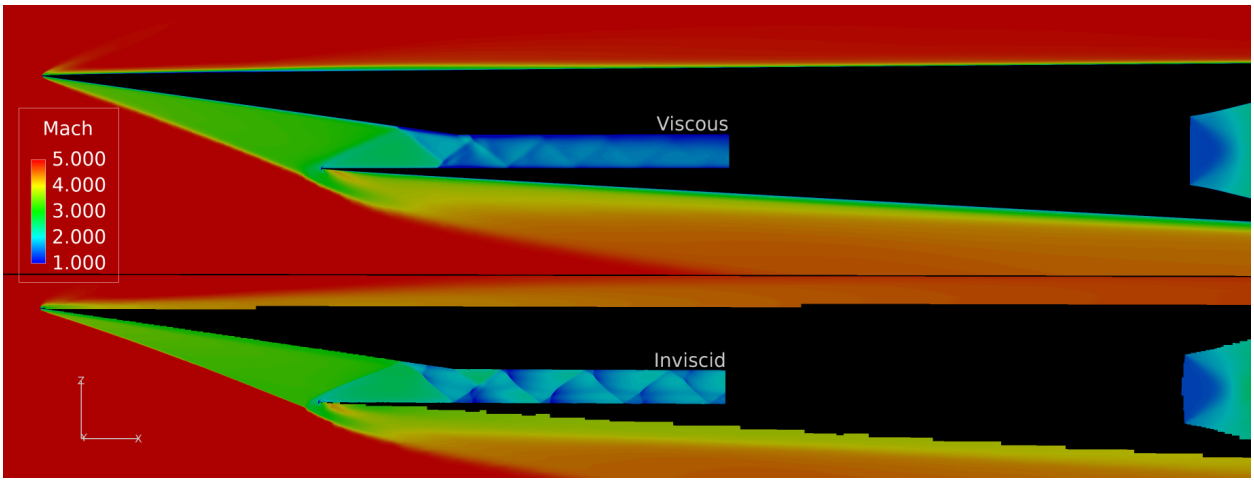


Figure 4.8

Mach Number on the Symmetry Plane for Mach 5 with an Angle of Attack of 6 Degrees

In the above figure, the viscous and inviscid solutions exhibit similar characteristics at the beginning of the inlet; the inlet shock angle does not vary much between the two. However, in the isolator, things begin to differ. The inviscid solution exhibits the expected shock train with the crisp shocks captured by the adapted mesh. In the viscous solution, the shocks are much more

diffuse. In addition, there is significant shock-induced separation on the top of the isolator where the reflected shocks impinge upon the surface. This separation grows until it exits out the domain at the entrance to the combustor. In addition, although it is faint in the viscous solution, it can be observed that the point at which the shock intersects the entrance to the combustor is different than in the inviscid solution. In the Mach 6 and 7 cases in Figures 4.9 and 4.10, the inviscid and viscous solutions appear mostly similar. In the Mach 6 case, which is at zero angle of attack, there is flow separation at the top of the isolator due to a combination of shock impingement and the flow turning a convex corner. In the Mach 7 case, this is much more pronounced, possibly because the flow has to turn a sharper corner due to the -4 degrees of angle of attack.

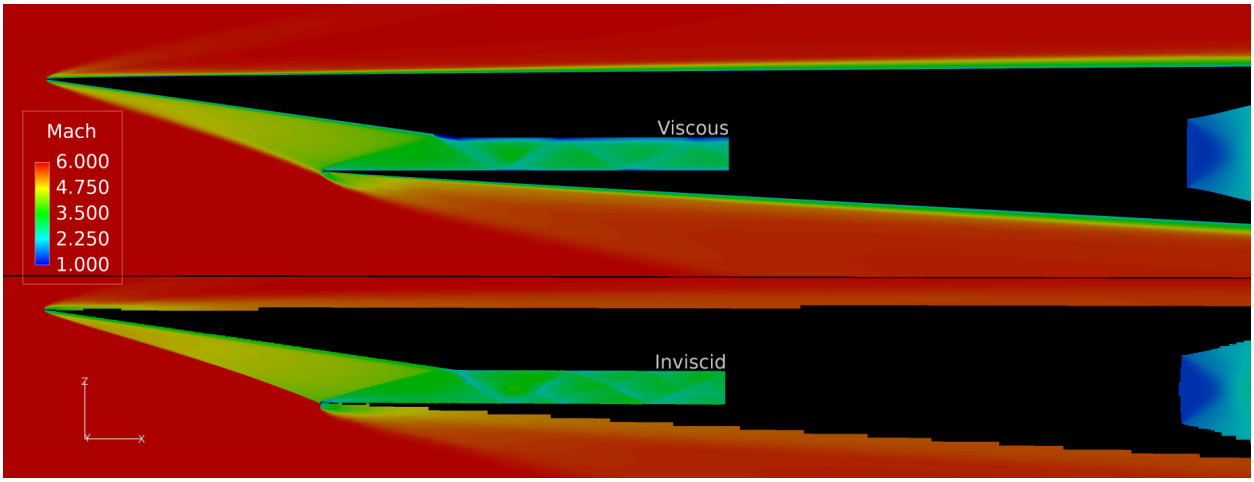


Figure 4.9

Mach Number on the Symmetry Plane for Mach 6 with an Angle of Attack of 0 Degrees

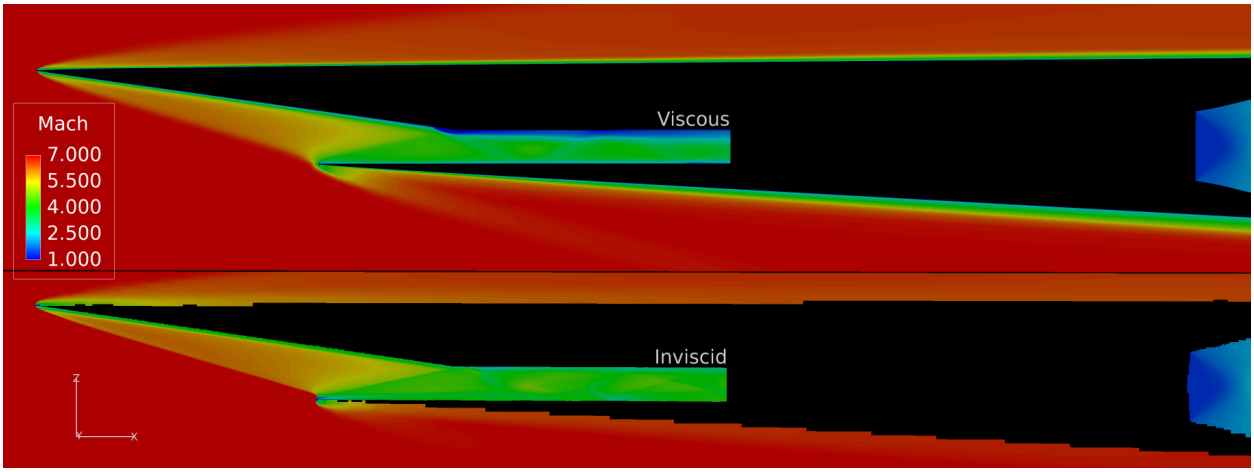


Figure 4.10

Mach Number on the Symmetry Plane for Mach 5 with an Angle of Attack of -4 Degrees

Limitations

There are a few limitations to this study which must be kept in mind. First, since the database was computed using an inviscid code, one can expect that axial force is being overestimated, while normal force should be relatively unaffected. The viscous solutions on the trimmed data points do not produce a zero pitching moment, suggesting that on a real vehicle there would need to be some room for error in the trimmed elevon position. The inviscid database should still be very appropriate for parametric or preliminary design, as the results are around 20% of the viscous results. The viscous and inviscid results both assume constant specific heats, which is possibly inaccurate in the inlet after the flow has been compressed through a few shocks. In the combustor, and into the nozzle specific heat has almost certainly changed due to the high heat and chemical reaction products.

In the current research, the inputs to the 1D combustor were obtained by averaging the primitive variables, i.e., density, velocity, and pressure. A worthy topic of future investigation would be to look at the effects of using averaged fluxes of mass, momentum, and energy, i.e., averaging the conserved variables as this would ensure that mass, momentum, and energy are conserved.

CHAPTER V

CONCLUSION

An inviscid trimmed powered database for a generic hypersonic vehicle was created. Combustion was modeled as quasi-one-dimensional heat addition with area variation. Net forward axial forces were achieved in all cases, with the highest axial force produced at Mach 5 at high angles of attack. Axial force increased with angle of attack and equivalence ratio. Normal force was almost exclusively a function of angle of attack, and was not particularly sensitive to Mach number or equivalence ratio. Selected trajectory points were studied with a viscous flow solver to evaluate the efficacy of the inviscid solutions. The viscous simulations predicted a lower axial force than the inviscid simulations in all cases, while normal force was nearly the same. The change in axial force between the viscous and inviscid solutions may be accounted for by the viscous shear stresses and the change in flow physics which modify the combustion predictions. In viscous and inviscid solutions, all parts of the vehicle were considered simultaneously, as to ensure their effects are coupled. One recommendation for future work would be to use averaged fluxes rather than averaged primitive variables as the input to the quasi-one-dimensional heat addition code. In addition, a sensitivity study on the heat addition code could be performed to determine the sensitivity to the input conditions provided from the CFD solvers. Further analysis into the inlet unstart due to high surface temperatures would also be of interest. Finally,

computational resources permitting, a full grid convergence study would be extremely beneficial to future work. Experimental data would be required to further evaluate the accuracy of the inviscid and viscous solutions. This would also help determine which assumptions may be made without losing significant accuracy, as the hypersonic CFD simulations involve trade-offs between model complexity and feasibility.

REFERENCES

[1] W. Heiser, D Pratt, D. Daley, and U. Mehta. *Hypersonic Airbreathing Propulsion*. AIAA Education Series, 1994. ix, 3, 4, 5

[2] M. Smart. Scramjets. *RTO-EN-AVT-150*, 2008. 2, 5

[3] Brent Ruttle, Jacob Stork, and Glenn Liston. Generic hypersonic vehicles for conceptual design analysis. Technical report, Air Force Research Lab, Wright-Patterson AFB, Ohio, September 2012. 8, 9

[4] S/habp mark v. Douglas Aircraft, 1964. Supersonic/Hypersonic Arbitrary Body Program. 9

[5] Charles E. Cockrell Jr., Walter C. Engelund, Robert D. Bittner, Dilley Jentink, Tom N., D. Arthur, and Abdelkader Frendi. Integrated aeropropulsive computational fluid dynamics methodology for the hyper-x flight experiment. *Journal of Spacecraft and Rockets*, 38, No.6, November-December 2001. 10, 55

[6] Shahriar Keshmiri, Richard Colgren, and Maj Mirmirani. Development of an aerodynamic database for a generic hypersonic air vehicle. *AIAA Paper 2005-6257*, 2005. 11

[7] Shahriar Keshmiri, Richard Colgren, Saeed Farokhi, and Maj Mirmirani. Ramjet and scramjet engine cycle analysis for a generic hypersonic vehicle. *AIAA Paper 2006-8158*, 2006. 12

[8] Maj Mirmirani, Chivey Wu, Andrew Clark, and Sangbum Choi. Modeling for control of a generic airbreathing hypersonic vehicle. *AIAA Paper 2005-6256*, August 2005. 13

[9] Heather Kline, Francisco Palacios, and Juan J. Alonso. Sensitivity of the performance of a 3-dimensional hypersonic inlet to shape deformations. *AIAA Paper 2014-3228*, June 2014. 15

[10] R. Baurle and R Gaffney. The art of extracting one-dimensional flow properties ofrom multi-dimensional datasets. *AIAA*, January 2007. 16

[11] Sun Shu, Zhang Hongying, and Wu Yizhao. The full flowpath analysis of a hypersonic vehicle. *Chinese Journal of Aeronautics*, November 2006. 16

[12] R. A. Baurle, T. Mathur, M. R. Gruber, and K. R. Jackson. A numerical experimental investigation of a scramjet combustor for hypersonic missile applications. *AIAA Paper 1998-3121*, July 1998. 17

[13] T. Langener, J. Steelant, P. Roncioni, P. Natale, and M. Marini. Preliminary performance analysis of the lapcat-mr2 by means of nose-to-tail computations. *AIAA Paper 2012-5872*, September 2012. 18, 19

[14] Maurice J. Zucrow and Joe D. Hoffman. *Gas Dynamics, Volume I*. Wiley, August 1976. 23

[15] Cart3d. NASA Ames, Moffett Field, CA. Inviscid Analysis Package, v1.5.5. 25

[16] Fun3d. NASA Langley, Hampton, VA. Unstructured CFD Code, V13.6. 25, 35

[17] Pointwise, inc. Pointwise, Inc., Forth Worth, TX. Mesh Generation Software, V18.4R2. 25, 35

[18] J.R. Edwards. A low-diffusion flux-splitting scheme for navier stokes calculations. *Computers & Fluids*, 26. 38

APPENDIX A

INVISCID DATABASE

Table A.1

Trimmed Elevon Deflection Angle as a Function of Mach Number, Angle of Attack, and Equivalence Ratio (positive is Elevon up)

		Mach		
Phi	AoA (deg)	5	6	7
0.5	-4	0.94383	2.34559	3.71086
	-2	2.97138	3.84874	4.19160
	0	4.59624	4.99150	4.28353
	2	5.63819	5.10595	3.82649
	4	5.38122	4.69906	2.81436
	6	4.37705	3.03918	1.91298
0.6	-4	0.68202	2.09899	3.47832
	-2	2.69777	3.58706	3.89783
	0	4.27849	4.65905	3.95592
	2	5.29029	4.73143	3.52604
	4	5.00466	4.37610	2.52350
	6	4.00058	2.74132	1.70553
0.7	-4	0.45074	1.85310	3.20575
	-2	2.44613	3.31892	3.64536
	0	3.96998	4.32604	3.69965
	2	4.95440	4.36167	3.25260
	4	4.61513	4.01861	2.26586
	6	3.67050	2.46039	1.50289
0.8	-4	0.22342	1.60429	3.02593
	-2	2.19218	3.04407	3.37590
	0	3.69443	4.01010	3.40949
	2	4.62186	4.00025	2.96366
	4	4.24518	3.74773	1.99431
	6	3.36382	2.18348	1.31054

1.0	-4	-0.22804	1.11163	2.57460
	-2	1.69844	2.51788	2.83916
	0	3.16015	3.44626	2.88292
	2	4.07654	3.58023	2.41702
	4	3.63992	3.21206	1.56702
	6	2.84776	1.67819	0.93370
1.2	-4	-0.66549	0.65935	2.14689
	-2	1.23705	2.02054	2.32084
	0	2.64320	2.90098	2.34834
	2	3.40290	2.98317	1.89770
	4	3.04117	2.69083	1.15102
	6	2.18430	1.22735	0.54410

Table A.2

Trimmed Axial Force Coefficient as a Function of Mach Number, Angle of Attack, and Equivalence Ratio

		Mach		
Phi	AoA (deg)	5	6	7
0.5	-4	-0.00345	-0.00248	-0.00165
	-2	-0.00394	-0.00275	-0.00179
	0	-0.00448	-0.00307	-0.00194
	2	-0.00495	-0.00339	-0.00203
	4	-0.00543	-0.00167	-0.00201
	6	-0.00570	-0.00355	-0.00193
0.6	-4	-0.00448	-0.00333	-0.00234
	-2	-0.00511	-0.00373	-0.00254
	0	-0.00577	-0.00413	-0.00282
	2	-0.00638	-0.00456	-0.00295
	4	-0.00694	-0.00281	-0.00303
	6	-0.00735	-0.00494	-0.00301
0.7	-4	-0.00543	-0.00412	-0.00304
	-2	-0.00621	-0.00466	-0.00332
	0	-0.00705	-0.00516	-0.00362
	2	-0.00775	-0.00572	-0.00388
	4	-0.00844	-0.00394	-0.00402
	6	-0.00895	-0.00622	-0.00409
0.8	-4	-0.00638	-0.00494	-0.00369
	-2	-0.00729	-0.00557	-0.00407
	0	-0.00827	-0.00617	-0.00450
	2	-0.00905	-0.00684	-0.00476
	4	-0.00992	-0.00503	-0.00501
	6	-0.01049	-0.00750	-0.00514

1.0	-4	-0.00823	-0.00651	-0.00500
	-2	-0.00941	-0.00737	-0.00553
	0	-0.01060	-0.00814	-0.00607
	2	-0.01168	-0.00889	-0.00652
	4	-0.01269	-0.00721	-0.00690
	6	-0.01348	-0.01000	-0.00714
1.2	-4	-0.01000	-0.00801	-0.00629
	-2	-0.01144	-0.00910	-0.00696
	0	-0.01286	-0.01005	-0.00768
	2	-0.01417	-0.01094	-0.00825
	4	-0.01544	-0.00928	-0.00871
	6	-0.01638	-0.01236	-0.00915

Table A.3

Trimmed Normal Force Coefficient as a Function of Mach Number, Angle of Attack, and Equivalence Ratio

		Mach		
Phi	AoA (deg)	5	6	7
0.5	-4	-0.01239	-0.01200	-0.01153
	-2	-0.00240	-0.00299	-0.00311
	0	0.00781	0.00645	0.00570
	2	0.01871	0.01663	0.01514
	4	0.03059	0.02744	0.02540
	6	0.04321	0.03940	0.03667
0.6	-4	-0.01231	-0.01192	-0.01147
	-2	-0.00229	-0.00290	-0.00304
	0	0.00792	0.00654	0.00578
	2	0.01884	0.01673	0.01522
	4	0.03073	0.02755	0.02549
	6	0.04336	0.03952	0.03676
0.7	-4	-0.01222	-0.01185	-0.01141
	-2	-0.00219	-0.00282	-0.00296
	0	0.00804	0.00664	0.00585
	2	0.01897	0.01684	0.01530
	4	0.03086	0.02765	0.02557
	6	0.04350	0.03963	0.03686
0.8	-4	-0.01213	-0.01177	-0.01134
	-2	-0.00210	-0.00273	-0.00289
	0	0.00815	0.00673	0.00593
	2	0.01909	0.01694	0.01539
	4	0.03100	0.02775	0.02567
	6	0.04364	0.03975	0.03695

1.0	-4	-0.01196	-0.01163	-0.01122
	-2	-0.00190	-0.00257	-0.00277
	0	0.00836	0.00691	0.00607
	2	0.01932	0.01714	0.01554
	4	0.03125	0.02794	0.02583
	6	0.04391	0.03997	0.03713
1.2	-4	-0.01181	-0.01149	-0.01111
	-2	-0.00172	-0.00241	-0.00264
	0	0.00856	0.00708	0.00622
	2	0.01955	0.01732	0.01570
	4	0.03149	0.02813	0.02600
	6	0.04416	0.04019	0.03732

APPENDIX B

FLOW SOLVER INPUT FILES

Cart3D INPUT FILE: input.cntl

```
#
#
# +-------------------------------------------------------------+
# | Steering and Control file for "flowCart" |
# | 3D Cut-Cell Cartesian Flow Solver |
# | |
# | see an interactive on-line example of this file at |
# | http://people.nas.nasa.gov/~aftosmis/cart3d/input_cntl.html |
# +-------------------------------------------------------------+
#
# NOTE: o Start Comments in this file with the "#" character
# o Blocks can come in any order
# o info within blocks can come in any order
#
#
#
$__Case_Information: # ...Specify Free Stream Quantities
Mach 6.0 # (double)
alpha 0.00 # (double) - angle of attack
beta 0.0 # (double) - sideslip angle

$__File_Name_Information:
MeshInfo Mesh.c3d.Info # Mesh info file (usually Mesh.c3d.Info)
MeshFile Mesh.mg.c3d # Mesh file

# --NOTE: ...surface triangulation specified in 'MeshInfo' file ------

$__Solver_Control_Information:
# Runge-Kutta Stage Coefficients
# stageCoef GradEval ->to run 1st order, set GradEval to 0 in all stages
# -------- -------
RK 0.0695 1 # van Leer 5-stage
RK 0.1602 1 # "optimally damped 2nd order scheme"
RK 0.2898 1 # AIAA 89-1933-CP (CFLopt = 2.5 1st order)
RK 0.5060 1 # (CFLopt = ~1.2 2nd order)
RK 1.0 1 #
                      # (CFLopt = 0.694)
                    # NOTE: GradEval = 0 = no new evaluation at this stage,
                    # GradEval = 1 = Yes, re-evaluate at this stage
CFL 1.2 # CFL number
Limiter 2 # (int) default is 1, organized in order of increasing
```

```
                # dissipation.
                # Limiter Type: 0 = no Limiter
                # 1 = Barth-Jespersen
                # 2 = van Leer
                # 3 = sin limiter
                # 4 = van Albada
                # 5 = MinMod
                #
FluxFun 0 # (int) - Flux Function: 0 = van Leer
                # 1 = van Leer Hanel
                # 2 = Colella 1998
                # 3 = HLLC
Precon 0 # (int) - Preconditioning: 0 = scalar timestep
wallBCtype 0 # Cut-Cell Boundary Condition type 0 = Agglomerated Normals
                # 1 = SubCell Resolution
nMGlev 1 # (int) - Number of Multi-Grid levels (1 = single grid)
MG_cycleType 2 # (int) - MultiGrid cycletype: 1 = "V-cycle", 2 = "W-cycle"
                # 'sawtooth' cycle is: nPre = 1, nPost = 0
MG_nPre 1 # (int) - no of pre-smoothing passes in multigrid
MG_nPost 1 # (int) - no of post-smoothing passes in multigrid


$__Boundary_Conditions: # BC types: 0 = FAR FIELD
                        # 1 = SYMMETRY
                        # 2 = INFLOW (specify all)
                        # 3 = OUTFLOW (simple extrap)
Dir_Lo_Hi 0 0 0 # (int) (0/1/2) direction (int) Low BC (int) Hi BC
Dir_Lo_Hi 1 1 0 # (int) (0/1/2) direction (int) Low BC (int) Hi BC
Dir_Lo_Hi 2 0 0 # (int) (0/1/2) direction (int) Low BC (int) Hi BC

# 2 is exit, 3 is inlet
# SurfBC ID rho u v w p
SurfBC 3 9.407 4.037 0.00 0.00 0.01
SurfBC 2 2.37867564 5.12454077 0.00 0.00 23.24038395

$__Convergence_History_reporting:
iForce 1 # (int) - Report residual information every iSkip cycles.
iHist 1 # (int) - Update 'HistoryFile' every iHist cycles.
nOrders 12 # (int) - Num of orders of Magnitude reduction in residual.

$__Partition_Information:
nPart 1 # (int) - Number of SubDomains to partition into:
```

```
type 1 # (int) - Type of partitioning: 1 = SpaceFillingCurve

$__Post_Processing:
# Pretty printed cutting planes
Yslices 0.0001


$__Force_Moment_Processing:
#
# ... Axis definitions (with respect to body axis directions (Xb,Yb,Zb)
# w/ usual stability and control orientation)
# see:
# http://people.nas.nasa.gov/~aftosmis/cart3d/clic/html/clic_doc.html#frames
Model_X_axis -Xb
Model_Y_axis Yb
Model_Z_axis -Zb

# ... reference area and length specifications

#$_Reference_Area %f compNumberList or compNameList
Reference_Area 1.0 all

#$_Reference_Length %f compNumberList or compNameList
Reference_Length 1. all

# ... Force and Moment Info
Force entire
Moment_Point 108.370 0.00 -4.401 entire


$__Design_Info:

# Objective Function: SUM of functionals (J)
# J = 0 -> W(P-T)^N
# J = 1 -> W(1-P/T)^N

# Ref. Frame = 0 Aerodynamic Frame
# = 1 Aircraft (Body) Frame

# Force Format:
#
# Name Force Frame J N Target Weight Bound GMP_Comp
```

```
# (String) (0,1,2) (0,1) (0,1) (int) (dble) (dble) (0)
#
    ↪ ------------------------------------------------------------------------
    ↪
optForce CD 0 0 0 1 0. 1. 0 entire
optForce CL 2 0 0 1 0. 0.2 0 entire
```

Cart3D ADAPTATION DRIVING SCRIPT (INPUT PORTION ONLY): aero.csh

```
#!/bin/csh -f

# $Id: aero.csh,v 1.25 2018/05/09 17:54:54 mnemec Exp $

# AERO: Adjoint Error Optimization
# Script to drive adjoint-based mesh refinement

# ATTENTION: requires Cart3D release 1.5 or newer

# M. Nemec, Marian.Nemec@nasa.gov
# Oct 2006, last update: May 2018

# Help:
# ------
# % ./aero.csh help

# Read tips, hints and documentation in $CART3D/doc/adjoint
# See examples in $CART3D/cases/samples_adapt

# Set user specified options below, defaults are suggested

# -------------
# Basic options
# -------------

# Number of adaptation cycles, e.g. if you pick 8 then the run will
   ↪ terminate
# after the flow solve in adapt08. Min value is 0, max value is 99.
set n_adapt_cycles = 7

# maxR for initial mesh (cubes)
set maxR = 8

# Spanwise orientation (-y_is_spanwise flag in flowCart)
set y_is_spanwise = 1 # {Yes, No} = {1, 0}

# Set mesh2d = 1 for 2D cases, mesh2d = 0 for 3D cases
set mesh2d = 0

# ----------------
# Advanced options
```

```
# ----------------

# Uncomment next line to control thread affinity (improve parallel
   ↪ performance)
# on linux machines
#setenv KMP_AFFINITY compact

# ----- Flow and adjoint solver settings -----

# Number of fine-grid flowCart iterations on initial mesh
set it_fc = 200
# Additional flowCart iterations on each new mesh
# cycle 1 2 3 4 5 6 7 8 9 10 11 12
set ws_it = ( 200 250 250 200 200 200 200 200 0 0 0 0 )
# Number of fine-grid adjointCart iterations on each mesh
set it_ad = 200

# Number of flowCart multigrid levels (default=3)
set mg_fc = 3
# Number of adjointCart multigrid levels (usually same as flowCart)
set mg_ad = 3

# Limiter: default 1 (Barth-Jespersen) is the most accurate, 2 (van Leer) is
# smoother and may offer deeper convergence. Limiter 5 (minmod) is most
   ↪ robust
# and 0 means no limiter.
set limiter = 2

# Functional averaging window in terms of flowCart mg-cycles. This is useful
# for cases that do not converge to steady-state. The averaged functional is
# reported in the fourth column of fun_con.dat and in all_outputs.avg.dat
# (default: avg_window = 1).
set avg_window = 1

# ----- Mesh adaptation settings ------

# Maximum number of cells in the penultimate mesh, i.e. number of cells
   ↪ allowed
# in the working mesh for the final embedding. The error estimation step
# requires ~6.9 GB per million cells. You can use this to gauge the largest
# mesh your memory resources allow. Default value is 9M, which assumes a
# machine with 64 GB of memory.
```

```
set max_cells2embed = 16000000

# Specify mesh growth for each adaptation cycle. Mesh growth should be
   ↪ greater
# than 1 and less than or equal to 8. Specifying 8 means that you allow
# refinement of every cell in the mesh. Recommended minimum growth is 1.1.
   ↪ We
# found the sequence below to work well for many problems: if your initial
   ↪ mesh
# has roughly 10,000 cells, then after 10 adaptations it will surpass 10
# million cells. (For 2D cases, we recommend growth factors of 1.2 for first
# two cycles and 1.4 for the rest.) If you enter 0 for any cycle, then the
   ↪ mesh
# growth is selected automatically in that cycle.
# cycle 0 1 2 3 4 5 6 7 8 9 10 11 12
set mesh_growth = ( 1.2 1.5 2.0 2.0 2.0 2.0 2.0 2.5 2.5 2.5 2.5 0 0 )

# Mesh growth can be selected automatically by enabling the auto_growth
# option. This is a new feature, where aero.csh sets the refinement
   ↪ threshold
# to the mean of the error distribution. This feature frequently yields
   ↪ better
# results than the default mesh_growth array.
set auto_growth = 0 # 0=no, 1=yes, default=0

# Set apc: adapt or interface propagation cycle
# a = adapt
# p = propagate interfaces (adapt mesh without reducing finest cell size)
# Use p-cycles sparingly. We recommend using one initial p-cycle to reduce
   ↪ the
# bias of the initial mesh. P-cycles should also be used once the volume
   ↪ mesh
# over-refines your surface triangulation.
# cycle 0 1 2 3 4 5 6 7 8 9 10
set apc = ( p a a a a a a a a a a )

# ----- Customization of initial mesh -----

# Use file name preSpec.c3d.cntl for preSpec regions (either BBoxes or XLevs
# for cubes or ABoxes for adapt) 0=no, 1=yes, default=0
set use_preSpec = 1
```

```
# Initial mesh parameters (cubes)
set cubes_a = 3 # angle criterion (-a)
set cubes_b = 3 # buffer layers (-b)

# Remesh: use refMesh.{mg.c3d,c3d.Info} to guide cell density of initial
   ↪ mesh
# (cubes -remesh option). If turned on, we recommend increasing mg_init
# (initial number of multigrid levels, see flag below) to 3 or 4. Note that
   ↪ the
# cubes_a flag is automatically set to 20 if it is less than 20. 0=no, 1=yes
   ↪ ,
# default=0
set use_remesh = 0

# Internal mesh (cubes): 0=no, 1=yes, default=0
set Internal = 0

# ----- Run control -----

# Do error analysis on finest mesh? 0=no, 1=yes, default=0
set error_on_finest = 0

# Exit on reaching the finest mesh, do not flow solve there. This is useful
   ↪ if
# you wish to run a different solver (e.g. the MPI version) on the finest
   ↪ mesh.
# The final adapt?? directory holds the final mesh, all the input files, as
# well as a FLOWCART file that contains the appropriate command line.
set skip_finest = 0 # 0=no, 1=yes, default=0

# Maximum level of refinement allowed in the mesh. This controls the size of
# the smallest cell in the mesh. Default value is 21, which is the maximum
# supported by Cart3D. If max_ref_level is reached before n_adapt_cycles,
   ↪ then
# propagation (p) cycles will be executed until n_adapt_cycles is satisfied.
set max_ref_level = 21

# Set extra refinement levels for the final mesh. This allows you to adapt
   ↪ the
# mesh multiple times with the same error map in the last adapt cycle,
   ↪ thereby
# bypassing the flow, adjoint, and error estimation steps. Use with caution:
```

```
# the mesh should be fine enough so that the error estimate is decreasing -
# preferably the solution should be in the Richardson region. This helps
# circumvent the memory limitations of the error estimation code. Default
   ↪ value
# is 0 and maximum allowed value is 3.
set final_mesh_xref = 0

# ------------------------------------------------
# EXPERT user options: flags below are rarely changed
# ------------------------------------------------

# Cut-cell gradients: 0=best robustness (default), 1=best accuracy
# If mesh2d=1, then we set tm=1 automatically
set tm = 0

# CFL number: usually ~1.1 but with power may be lower, i.e. 0.8
set cfl = 1.0
# Minimum CFL number, used in case of convergence problems with flowCart
set cflmin = 0.8

# Adaptation error tolerance. Run terminates if the error estimate falls
# below this value. At least 2 cycles will be run before terminating based
   ↪ on
# this tolerance to avoid triggering based on an inappropriate initial mesh.
set etol = 0.000001

# Grid sequencing (-gs) or multigrid (-mg) (-mg default)
# flowCart
set mg_gs_fc = '-mg'
# adjointCart
set mg_gs_ad = '-mg'

# Full multigrid: default is to use full multigrid, except in cases with
   ↪ power
# boundary conditions (automatic with warm starts)
set fmg = 0 # 0=no, 1=yes, default=1

# Polynomial multigrid, helps certain tough cases converge deeper
set pmg = 1 # 0=no, 1=yes, default=0

# Buffer limiter: improves stability for flows with strong off-body shocks
set buffLim = 1 # 0=no, 1=yes, default=0
```

```
# Number of multigrid levels for initial mesh (default 2, ramps up to mg_fc/
   ↪ ad)
set mg_init = 3

# Set names of executables (must be in path)
set flowCart = flowCart
set xsensit = xsensit
set adjointCart = adjointCart
set adjointErrorEst = adjointErrorEst_quad

# MPI prefix: uncomment next line and also remember to set the correct
   ↪ flowCart
# executable above. If not running MPI, comment out next line.
# set mpi_prefix = 'mpiexec -n 16'

# Flow solver warm-starts: 0=no, 1=yes, default=1
set use_warm_starts = 1

# Subcell resolution: 0=no, 1=yes, default=0
set subcell = 0

# Run adjoint solver in 1st-order mode. In hard cases where aero.csh
# consistently falls back on 1st-order mode after trying more accurate
# settings, this setting will short-circuit the process and go directly to
# 1st-order adjoints. The flow solution remains 2nd-order. The adjoints
   ↪ should
# still provide a consistent set of error estimates, which is probably safe
   ↪ for
# _relative_ errors and adaptive meshing. However, use caution: the error
# estimates may be inaccurate with respect to a 2nd-order run.
# default 0=auto, 1=force 1st order
set adj_first_order = 0

# In 3D cases with tm=1, error estimation is still done with tm=0 for
# robustness. This flag forces aero.csh to use tm=1 in error estimation.
   ↪ This
# is recommended only for simple (academic) cases that converge well. In
# general, the default (0) setting is _strongly_ recommended.
set err_TM1 = 0 # default 0=off, 1=force tm 1 for error estimation

# In 3D cases with tm=1, adjoint solutions are still done with tm=0 for
```

```
# robustness. This flag forces aero.csh to use tm=1 for the adjoint solves.
   ↪ In
# general, the default (0) setting is _strongly_ recommended. Note that
# adjoint convergence is monitored, so if divergence occurs then tm=0 is set
# during runtime.
set adj_TM1 = 0 # default 0=off, 1=force tm 1 for adjoint solutions

# If set, the flow solve on the final mesh will use *at least* this many
# iterations. flowCart will use whichever is greater: (1) this value (2) the
# ws_it array entry. This helps when you do not know how many adaptation
   ↪ cycles
# will be necessary to meet the termination conditions and you would like a
# well converged answer on the final mesh. Default value is 0.
set ws_it_min_final = 0

# When running optimization with adaptive meshing, this flag allows
   ↪ different
# levels of adjoint convergence for the mesh adaptation functional vs. the
# design functionals. The iterations for the adjoint solutions used in
   ↪ gradient
# computations are it_ad + delta_it_ad. The main idea is to use fewer
# iterations when building the mesh (for speed) and go for deeper converge
   ↪ in
# the gradient adjoints (for better accuracy). Default value is 0.
set delta_it_ad = 0

# Keep final error map in EMBED/Restart.XX.file, useful for cubes -remesh
set keep_error_maps = 0 # default 0=no, 1=yes

# Refine all cells: useful for uniform mesh refinement studies. This
   ↪ overrides
# the error map and forces adapt to refine all cells. The adjoint correction
# term and error estimate are reported.
set refine_all_cells = 0 # default 0=no, 1=yes

# adapt buffers (default 1)
set buf = 1

# Set the number of multigrid levels when aero.csh drops down to pMG due to
# convergence problems. Default value is 2, which means no geometric
# multigrid. In subsonic cases, 3 multigrid levels may be better. Note that
# this flag has no effect on the pmg multigrid levels when the pmg flag is
```

```
# selected above. It influences only the automatic run control of aero.csh.
set mg_pmg_auto = 2

# Fine tuning of mesh growth when performing extra refinements on the final
# mesh, i.e. when $final_mesh_xref>0 and $mesh_growth are being used.
# The mesh growth for each extra refinement is given by:
# ($mesh_growth-1)*$xref_fraction+1
# The main idea is that as extra refinement cycles are performed, the
# adaptation focuses on only the highest error cells. This is where the
   ↪ error
# map is most accurate and most adaptation is required. Each value should be
# between 0.2 and 1, and at most three extra refinements are allowed.
set xref_fraction = ( 1.0 1.0 0.8 )

# Safety factor used in aero_getResults.pl to terminate the run if the error
# indicator value increases by more than this factor in successive
# cycles. Default value is 8.
set error_safety_factor = 8

# Adaptation restart: Alternative to command line argument 'restart'
set adapt_restart = 0

# Adaptation jumpstart from existing mesh: Alternative to command line
# argument jumpstart
set adapt_jumpstart = 0

# Write Tecplot output files in binary format
set binaryIO = 1 # default 1=yes, 0=no

# Verbose mode for executables [flowCart/xsensit/adjointCart/adjointErrorEst
   ↪ ]
# 0=no, 1=yes, default=0
set verb = 0

# minimum mesh growth
set min_growth = '1.1'

# Adaptation threshold array: To set ath manually, unset mesh_growth
# and set the ath array (uncomment following two lines):
# cycle 0 1 2 3 4 5 6 7 8 9 10

#set ath = ( 32 16 8 4 2 1 1 1 1 1 1 )
```

```
#unset mesh_growth

# Output cell-wise errors, useful for making histograms: 0=off, 1=yes,
# default=0
set histo = 0
```

FUN3D SAMPLE INPUT FILE: fun3d.nml

```
&project
   project_rootname = "ghv"
   case_title = "ghv"
/

&governing_equations
   eqn_type = 'compressible'
   artificial_compress = 15.0
   viscous_terms = 'turbulent'
   chemical_kinetics = 'finite-rate'
   thermal_energy_model = 'non-equilib'
   prandtlnumber_molecular = 0.72
   schmidt_number = -1.0
   gas_radiation = 'off'
   rad_use_impl_lines = .false.
   multi_component_diff = .false.
/

&reference_physical_properties
   dim_input_type = 'nondimensional'
   gridlength_conversion = 1.0
   mach_number = 6.0
   vinf_ratio = 1.0
   reynolds_number = 5.342e06
   velocity = 0.0
   density = 0.0
   temperature = 220.9
   temperature_units = 'Kelvin'
   angle_of_attack = 0.00
   angle_of_yaw = 0.00
/

&boundary_conditions
   static_pressure_ratio(3) = 0.01
   q_set(2,1:5) = 2.36108821 5.07866533 0.00 0.00 22.1375048

   ! Tinf = 220.9 K
   ! T = 366.48 K (200 F)
   ! T/Tinf = 1.69
   ! For now set all to adiabatic
   wall_temperature(1) = 1.60
```

```
   wall_temp_flag(1) = .true.

   wall_temperature(4) = 1.60
   wall_temp_flag(6) = .true.

   wall_temperature(5) = 1.60
   wall_temp_flag(5) = .true.

   wall_temperature(7) = 1.60
   wall_temp_flag(7) = .true.

   wall_temperature(8) = 1.60
   wall_temp_flag(8) = .true.



/
 &inviscid_flux_method
   flux_construction = 'aldfss'
   flux_construction_lhs = 'vanleer'
   kappa_umuscl = -1.0
   flux_limiter = 'hvanalbada'
   smooth_limiter_coeff = 1.0
   freeze_limiter_iteration = -1
   first_order_iterations = 0
   multidm_option = 1
   fixed_direction = .true.
   recalc_dir_freq = 1
   adptv_entropy_fix = .false.
   rhs_u_eigenvalue_coef = 0.0
   lhs_u_eigenvalue_coef = 0.0
   rhs_a_eigenvalue_coef = 0.0
   lhs_a_eigenvalue_coef = 0.0
   entropy_fix = .false.
   temperature_fix = .false.
   re_min_vswch = 50.0
   re_max_vswch = 500.0
   pole_gradient = .false.
 /

 &turbulent_diffusion_models
   turbulence_model = 'sa'
```

```
   turb_intensity = -0.001
   turb_viscosity_ratio = -0.001
   reynolds_stress_model = 'linear'
   turb_compress_model = 'off'
   turb_conductivity_model = 'off'
   prandtlnumber_turbulent = 0.90
   schmidtnumber_turbulent = 1.0
   miv_hrles = .false.
   dw_rans = 0.2
   dw_les = 0.4
   strelets_des = .false.
/

&turbulence
   use_least_squares_gradients = .true.
/

&force_moment_integ_properties
   area_reference = 3.8880 ! m^2
   x_moment_length = 4.318
   y_moment_length = 4.318
   x_moment_center = 2.753
   z_moment_center = 0.112
/

&nonlinear_solver_parameters
   schedule_cfl = 1.0 10.0
   schedule_cflturb = 1.0 10.0
/

&code_run_control
   steps = 1
   stopping_tolerance = 1.0E-15
   restart_write_freq = 1000
   restart_read = "on"
/

&raw_grid
  grid_format = "aflr3"
  data_format = "default"
  swap_yz_axes = .false.
/
```

```
 &volume_output_variables
   export_to = 'fvuns'
   x = .false.
   y = .false.
   z = .false.
 /

 &boundary_output_variables
   yplus = .true.
   cf_x = .true.
   cf_y = .true.
   cf_z = .true.
   temperature = .true.
   cq = .true.
   heating = .true.
 /

 &global
   volume_animation_freq = -1
   boundary_animation_freq = 1000
/

 &component_parameters
   number_of_components = 6
   component_name(1) = 'Engine'
   component_type(1) = 'boundary'
   component_input(1) = '2,3,8'
   list_forces = .true.
   component_symmetry(1) = 2.0
   massflow_component(1) = 1

   component_name(2) = 'Entire'
   component_type(2) = 'boundary'
   component_input(2) = '1-9'
   component_symmetry(2) = 2.0

   allow_flow_through_forces = .true.

   component_name(3) = 'Body'
   component_type(3) = 'boundary'
   component_input(3) = '1'
```

```
  component_symmetry(3) = 2.0

  component_name(4) = 'Inlet'
  component_type(4) = 'boundary'
  component_input(4) = '6'
  component_symmetry(4) = 2.0

  component_name(5) = 'Isolator'
  component_type(5) = 'boundary'
  component_input(5) = '7'
  component_symmetry(5) = 2.0

  component_name(6) = 'Elevon'
  component_type(6) = 'boundary'
  component_input(6) = '4'
  component_symmetry(6) = 2.0
/

&partitioning
  use_64bit_partitioning = .false.
/
```

APPENDIX C

1D PROPULSION MODEL

PROPULSION BOUNDARY CONDITION MATLAB CODE: combustor_exit_GHV.m:

Written by: Dr. Kidambi Sreenivas

```
% Beginning and end of combustor (inches)

x3 = 0.0;
x4 = 54.3;

% Area at the beginning and end of combustor (in^2)

A3 = 12.289;
A4 = 50.317;

% Radius at the beginning and end of combustor (inches)

R3 = sqrt(A3/pi);
R4 = sqrt(A4/pi);

% Dimensional heating value of Ethylene (kJ/kg)

hv_dim = 50902.18;

% Ethylene Stoichiometric fuel-air ratio (non-dimensional)

FA_stoic = 0.0676;

% Adiabatic Efficiency of combustor (non-dimensional)

eta_b = 0.85;

% Ratio of specific heats

gamma = 1.4; % Air at STP

% Cp (kJ/kg.K)

Cp = 1.005; % Air at STP

% Prompt for inputs

prompt = {'rho3', 'u3', 'mdot', 'p3', 'phi'};
dlgtitle = 'Combustor Inlet Conditions';
```

```
dims = [1 35];



M = 6;
Tinf = 216.69;
alpha = 4;

answer = [ 11.47 4.877 28.20 ];
rhobar = answer(1);
ubar = answer(2);
pbar = answer(3);
mdot = rhobar * ubar * A3;
% % alpha rhobar ubar pbar mdot (from c3d)
definput = [ alpha rhobar ubar pbar mdot];

%answer = inputdlg(prompt, dlgtitle, dims, definput);
answer = definput;

% Retrieve variables of interest

AOA = answer(1);
rho3 = answer(2);
u3 = answer(3);
p3 = answer(4);
mdot_target = answer(5);


% rho3 = str2double(answer{1});
% u3 = str2double(answer{2});
% mdot = str2double(answer{3});
% p3 = str2double(answer{4});

% Adjust velocity so that the mass flow rate into the combustor is matched

mdot_in = rho3*u3*A3;
mdot_ratio = mdot_target/mdot_in;
u3 = u3*mdot_ratio;
mdot_in = rho3*u3*A3;

fprintf('M = %4.2f AOA = %4.2f\n', M, AOA);
```

```
for phi =[0.50000001 0.6 0.7 0.80 1.0 1.1999] %phi = [0.3, 0.4,
    ↪ 0.4999999999, 0.6, 0.7, 0.8, 1.0, 1.2]
    % Calculate other variables of interest at inlet

    c3 = SoS(gamma, p3, rho3);
    M3 = u3/c3;
    T3 = gamma*p3/rho3;
    T03 = T3*T0_T(gamma, M3);
    p03 = p3*p0_p(gamma, M3);

    % Compute exit total temperature

    FA_ratio = phi*FA_stoic;
    q = eta_b*hv_dim*FA_ratio;

    T04 = (q/Cp + T03*Tinf)/Tinf;

    % Compute mass flow rates


    mdot_fuel = FA_ratio*mdot_in;
    mdot_out = mdot_in + mdot_fuel;

    % Integrate ODE to find M4

    [x,M] = ode45(@(x,M) dMdx(x, M, gamma, x3, x4, R3, R4, T03, T04,
        ↪ mdot_fuel, mdot_in), [x3 x4], [0; M3]);

    M4 = M(end);
    T4 = T04/T0_T(gamma, M4);

    p4 = p3*(mdot_out/mdot_in)*(A3/A4)*(M3/M4)*sqrt(T04/T03*(1 + (gamma - 1)
        ↪ /2*M3^2)/(1 + (gamma -1)/2*M4^2));
    p04 = p4*p0_p(gamma, M4);

    rho4 = rho3*(p4/p3)*(T3/T4);

    u4 = u3*(M4/M3)*sqrt(T4/T3);
    mdot4 = rho4*u4*A4;
    c4 = u4/M4;

% X = ['phi = ', num2str(phi)];
```

```
% disp(X);
% X = ['rho4 = ', num2str(rho4)];
% disp(X);
% X = ['u4 = ', num2str(u4)];
% disp(X);
% X = ['p4 = ', num2str(p4)];
% disp(X);
% X = ['M4 = ', num2str(M4)];
% disp(X);
% X = ['p04 = ', num2str(p04)];
% disp(X);
% X = ['T04 = ', num2str(T04), ' T4 = ', num2str(T4)];
% disp(X);
% X = ['mdot4 = ', num2str(mdot4)];
% disp(X);mdot_fuel

    const4 = p4/rho4;
    rho_out = mdot_out/(M4*c4*A4);
    p_out = rho_out*const4;

    mdot_out = rho_out*u4*A4;

    thrust_mdot = mdot_out*u4 - mdot_in*u3;
    thrust_p = p4*A4 - p3*A3;

% X = ['rho_out = ', num2str(rho_out), ' p_out = ', num2str(p_out)];
% disp(X);
% X = ['mdot_in = ', num2str(mdot_in), ' mdot_out = ', num2str(mdot_out), '
   ↪ mdot_fuel = ', num2str(mdot_fuel)];
% disp(X);

    if (M4 < 1.3 || M4 > 1.5)
        fprintf('Mach number problem; M4 is outside range; M4 = %4.2f\n', M4
           ↪ );
    end


    %fprintf('Inflow: %12.8f %12.8f %12.8f %4.2f %4.2f %12.8f %12.8f %12.8f\
       ↪ n', phi, rho3, u3, 0.0, 0.0, p3, M3, mdot_in);
    %fprintf('Outflow: %12.8f %12.8f %12.8f %4.2f %4.2f %12.8f %12.8f %12.8f
       ↪ \n', phi, rho4, u4, 0.0, 0.0, p4, M4, mdot_out);
```

```
    fprintf('%4.2f %12.8f %12.8f %4.2f %4.2f %12.8f\n', phi, rho4, u4, 0.0,
       ↪ 0.0, p4);
    %fprintf('Thrust_mdot: %12.8f Thrust_p: %12.8f\n', thrust_mdot, thrust_p
       ↪ );
end

function c = SoS(gamma, p, rho)

   % Calculate the speed of sound

   c = sqrt(gamma*p/rho);
end

function M = Mach(u, c)

   % Calculate the local Mach number

   M = u/c;
end

function Tstag = T0_T(gamma, M)

   % Calculate the stagnation temperature

   Tstag = (1 + 0.5*(gamma - 1)*M^2);
end

function pstag = p0_p(gamma, M)

   % Calculate the stagnation temperature

   pstag = (1 + 0.5*(gamma - 1)*M^2)^(gamma/(gamma - 1));
end

function Area = A(x, x3, x4, R3, R4)

   % Calculate the area at any given cross-section

   R = R3 + (R4 - R3)/(x4 - x3)*(x - x3);
   Area = pi*R^2;
end
```

```
function dAreadx = dAdx(x, x3, x4, R3, R4)

   % Calculate the rate of change of area with distance along combustor

   R = R3 + (R4 - R3)/(x4 - x3)*(x - x3);
   dAreadx = 2*pi*R*(R4 - R3)/(x4 - x3);
end

function T = Temperature(x, x3, x4, T03, T04)

   % Calculate the temperature at a specified location

   theta = 5.0; % Combustor shape factor
   chi = (x - x3)/(x4 - x3);
   tau_b = T04/T03;
   T = 1 + (tau_b - 1)*(theta*chi/(1 + (theta - 1)*chi));
end

function dTempdx = dTdx(x, x3, x4, T03, T04)

    % Calculate the rate of change of temperature with distance

   theta = 5.0; % Combustor shape factor
   chi = (x - x3)/(x4 - x3);
   tau_b = T04/T03;

   term1 = 1 + (theta - 1)*chi;
   dTempdx = (tau_b - 1)/term1*(theta + (theta*chi*(theta - 1)/term1));
end

function dMachdx = dMdx(x, M, gamma, x3, x4, R3, R4, T03, T04, mdot_fuel,
    ↪ mdot_in)

  facM = 1 + (gamma - 1)/2*M.^2;
  Area = A(x, x3, x4, R3, R4);
  dAreadx = dAdx(x, x3, x4, R3, R4);
  T = Temperature(x, x3, x4, T03, T04);
  dTempdx = dTdx(x, x3, x4, T03, T04);
  dmdotdx = mdot_fuel/(x4 - x3);

  dMachdx = M.*facM/(1 - M.^2)*(-dAreadx/Area + (1 + gamma*M.^2)/2*(dTempdx/
     ↪ T + 2*dmdotdx/mdot_in));
```

```
end

function pexit = Rayleigh(gamma, M3, M4, p03)

   % Use Rayleigh flow equation to compute p04 given M3, M4, and p03

   term1 = (1 + gamma*M3^2)/(1 + gamma*M4^2);
   e = gamma/(gamma - 1);
   term2 = ((1 + (gamma - 1)/2*M4^2)/(1 + (gamma - 1)/2*M3^2))^e;
   pexit = p03*term1*term2;
end
```

APPENDIX D

CART3D GMP TAG PROCEDURE

Cannon DeBardelaben
Jason DeHay
April 20, 2020

Tagging Faces for Cart3D Propulsion Boundary Conditions Using Pointwise

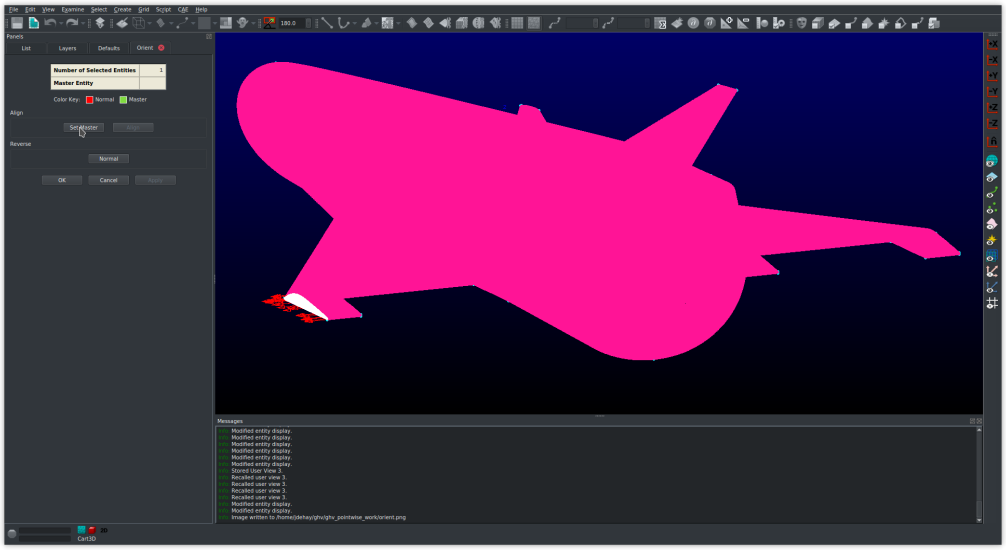
Step 1: CAE - > Select Solver - > Cart3D

a. This should automatically change dimension to 2D.

Step 2: Verify that all surface domains are pointed into the flow, i.e., all domains are pointing away from the geometry.

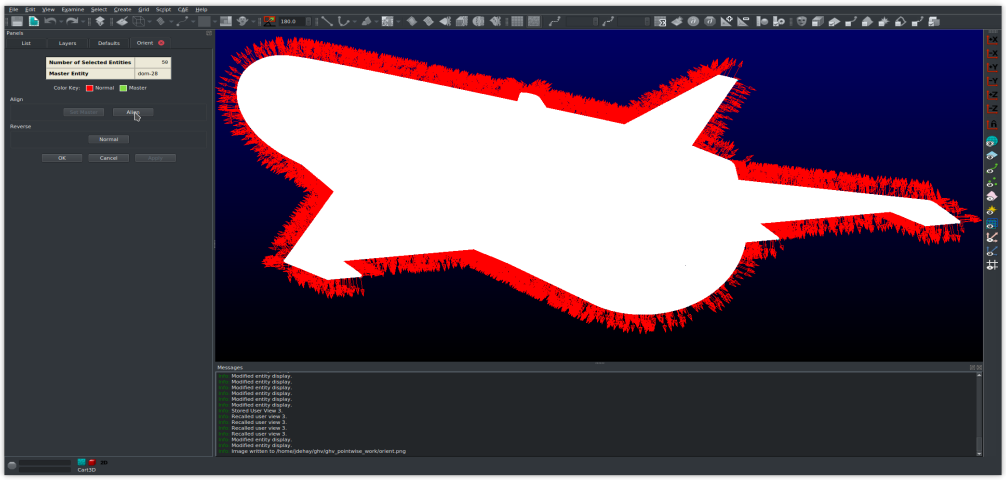
a. Select all domains and click Edit - > Orient

b. Select one domain and check if it is oriented correctly. If not, click Normal under the Reverse menu.



c. Select Set Master under the Align menu.

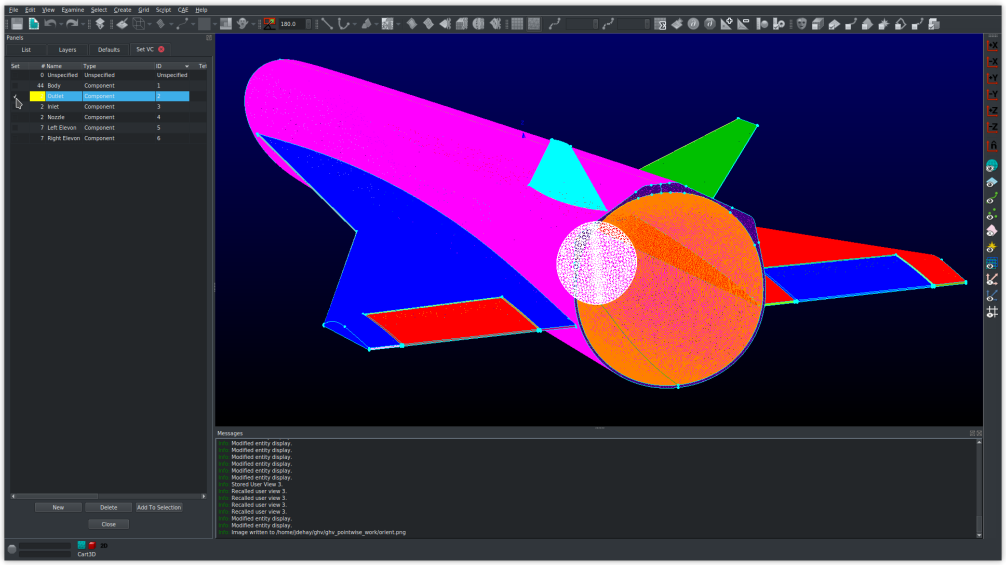
d. Select all domains and click Align under the Align menu.



d. Press OK to accept changes.

Step 3: Tag Surfaces Appropriate to the Specific Problem. In this case, we will tag the engine inlet face as one component, the engine outlet as another, and the rest of the geometry as a third component.

a. CAE - > Set Volume Conditions (NOT Boundary Conditions!).
b. Press New 3 times to create three new component IDs.
c. Re-name the Component IDs appropriately, and change Type to Component.



d. Assign domains to the appropriate component (e.g. all domains associated with the main body get selected and then set by clicking in the set box next to the Component ID)
e. Click close to apply changes.

Step 4: Export geometry in Cart3D .tri format.

a. Select all domains
b. File - > Export - > CAE
c. Type in file name and click Save.

If You'd Like to Intersect One or More Components, while keeping Component/GMP IDs:

Step 1: Perform the same steps as before, regarding orienting domains, and applying boundary conditions.

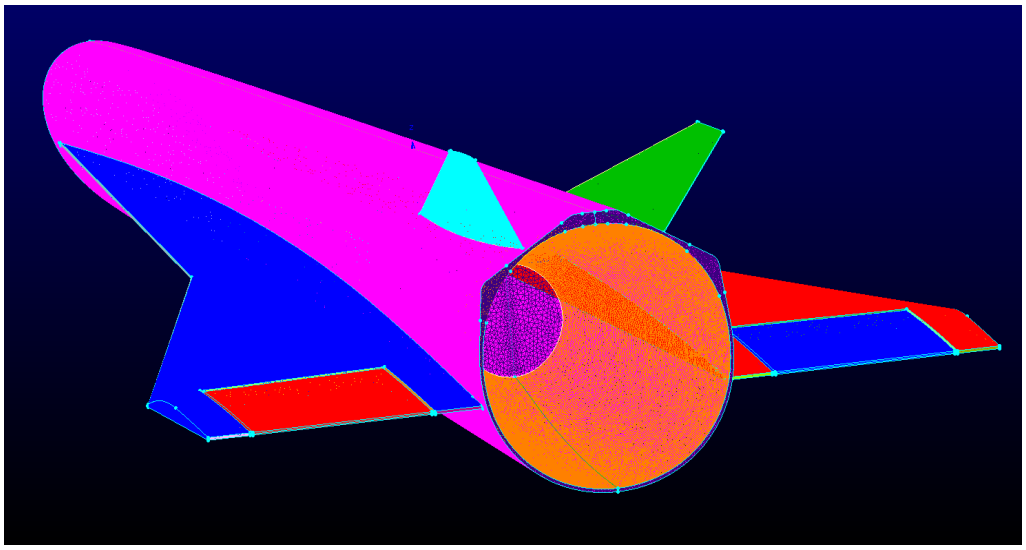
a. Write out each component to a separate *.a.tri file.

Step 2: Cart3D Commands for making GMP tags which are preserved after intersecting. In this case we will be observing a hyper-sonic vehicle with right and left elevons that we wish intersect with the main body of the vehicle; while retaining our previously defined component ID's through the "intersect" process. Begin by exporting each component separately as a *.a.tri file (i.e. left_elevon.a.tri, right_elevon.a.tri, and body.a.tri).

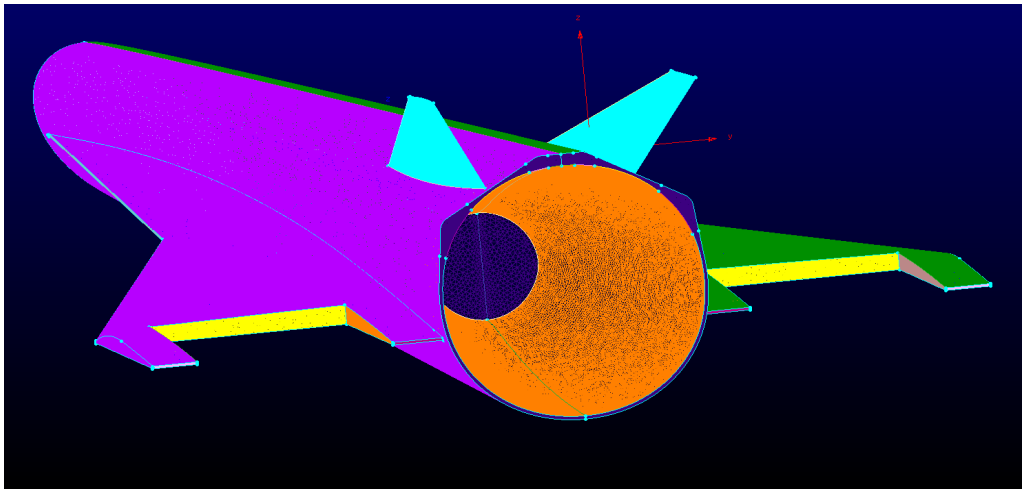
a. Select all domains associated with each component.

b. File → Export → CAE → type component name and select file type *.a.tri

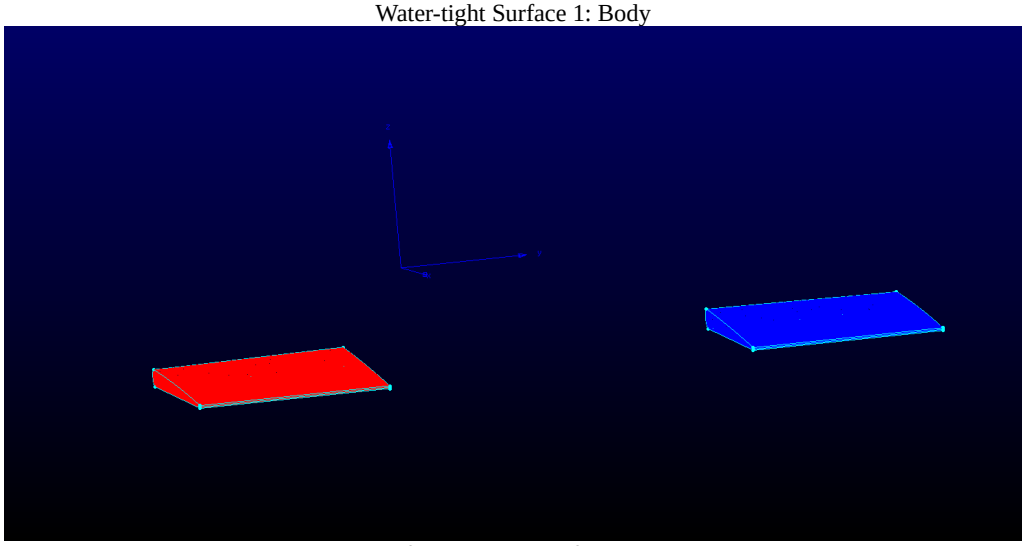
c. Click Save and repeat process for each component



Full Assembly



Water-tight Surface 1: Body



Water-tight Surfaces 2 and 3: Left and Right Elevons

a. Execute the following Cart3D commands in order.

```
trix -v -comp2gmp -o body.a body.a.tri
trix -v -comp2gmp -o left_elevon.a left_elevon.a.tri
trix -v -comp2gmp -o right_elevon.a right_elevon.a.tri
comp2tri -trix body.a.tri left_elevon.a.tri right_elevon.a.tri
intersect -T
```

Note: If there are degeneracies between the two components being intersected, the “-inflate” flag can be added to comp2tri which may help. In addition, one may want to verify the GMP numbers created by executing the “tail” command on the .tri files after trix to ensure the tags are continuous. If offsets in GMP tag numbers are needed, they can offset with the “-add2gmp %s” flag to the trix command.

VITA

Cannon DeBardelaben graduated from Franklin High School in Franklin, Tennessee. Cannon got his undergraduate degree from The University of Tennessee at Chattanooga before deciding to stay on as a graduate student in order to further his knowledge of aviation, one of his lifelong passions. In his free time, he is a rock-climber and an avid supporter of Liverpool Football Club. Cannon graduated with his Master's of Science in Engineering in August 2021 and is continuing to do hypersonics research as a research engineer in Dayton, Ohio.