Fall 2005

Tools for Analysis of Complex Geometries for a Combined Cycle SCRAM Jet / Rocket

James Moss
Embry-Riddle Aeronautical University - Daytona Beach

Tools for Analysis of Complex Geometries for a Combined Cycle SCRAM Jet / Rocket

by

James Moss

A Thesis submitted to the

Graduate Studies Office

In partial Fulfillment of the Requirements for the Degree of

Master of Science in Aerospace Engineering

Embry-Riddle Aeronautical University

Daytona Beach, Florida

Fall 2005

Tools for Analysis of Complex Geometries for a Combined Cycle SCRAM Jet / Rocket

by
James Moss

This thesis was prepared under the direction of the candidate's thesis committee chairman, Dr. Eric R. Perrell, Department of Aerospace Engineering, and has been approved by the members of his thesis committee. It was submitted to the Aerospace Engineering Department and was accepted in partial fulfillment of the requirements for the degree of Master of Science in Aerospace Engineering.

THESIS COMMITTEE:

[illegible]
Dr. Eric R. Perrell
Chairman

[illegible]
Dr. Axel Rohde
Member

[illegible]
Dr. Hany Nakhla
Member

[illegible] 7/18/05
Department Chair, Aerospace Engineering Date

Acknowledgements

Dr. Perrell has spent years working to develop his CFD code, hyp. He has done so with the intent of making the code not only a powerful analysis tool, but one open to the public for free use and modification. The author would like to express his appreciation of Dr. Perrell's efforts with this code and for being allowed to help improve the code's capabilities. Without this tool, this project could not have been completed in the manner in which it has been.

The advice of many other professors went into this project. Specifically, the author would like to thank Dr. Gupta for his help with some of the advanced compressible flow problems.

The model being examined in CFD is far too large to have been analyzed on any conventional machine. The majority of the CFD analysis was carried out using the 3000+ processor six teraflop supercomputer Lemieux, at the Pittsburgh Supercomputing Center. The author greatly appreciates the computer time made available for this project and the assistance provided by the consulting staff.

A number of people spent a great deal of time reading early versions of portions of this document. The author would specifically like to thank David McCall for spending more time and effort reviewing the document than was healthy for him.

The author wishes to thank his family and friends for their encouragement, support, and for bearing with his poor temper when nothing seemed to be working as it should.

<u>THANK YOU</u>

ABSTRACT

Author:	James Moss
Title:	Tools for Analysis of Complex Geometries for a Combined Cycle SCRAM Jet / Rocket
Institution:	Embry-Riddle Aeronautical University
Degree:	Master of Science in Aerospace Engineering
Year:	2005

A preliminary design for a variable geometry combined cycle supersonic combustion ram jet (SCRAM jet) / rocket was developed. A precise geometry was selected by parametric analysis and the resultant geometry was analyzed using a computational fluid dynamics (CFD) code developed by Dr. Eric Perrell, and modified by, among others, the author.

This paper is broken into several sections discussing, separately, the design innovations used to permit the geometry of the craft to be adjusted to conform to its flight conditions, the 2D analysis tool developed for the parametric analysis, development of the grid for CFD analysis, and modifications made to the CFD code to accommodate the complex geometries that necessarily were incorporated in the grid. Results are discussed at the end.

Table of Contents

List of Tables

List of Figures

1. Introduction

Rockets have been used, since Sputnik, to launch payloads into orbit. Capacity and efficiency have improved continuously since that time. Rocket propulsion has proved to be a very effective, in fact the only, means, to date, of getting a payload out of the Earth's atmosphere. One critical inefficiency of this form of propulsion is the necessity of carrying and consuming onboard oxidizer even while the rocket is passing through oxygen rich atmosphere. Extra energy is consumed simply to transport this oxidizer for consumption later in the flight.

Typically, to get a payload into Earth orbit, main engines are supplemented with booster rockets which detach and drop, expended, to Earth as the craft ascends. Conventional rocket designs typically are built in multiple stages, so that as the craft ascends, successive engines fire, burn out, and are released from the craft, decreasing the load that the upper stages need to lift. The problem with the use of booster rockets and multiple stages is that the booster rockets and discharged stages are not always reusable, necessitating remanufacturing. While this is good for the aerospace manufacturing industry, it adds excess cost to missions and limits the frequency of launches, subject to the rate at which new components can be produced.

One popular concept for reducing the cost of sending payloads to orbit is the reusable Single Stage To Orbit vehicle (SSTO). Supersonic combustion ram jet (SCRAM jet) operation would make an ideal intermediate phase between conventional jet propulsion and rocket propulsion for an SSTO, accelerating a craft up to very high speeds and operating at extreme altitudes, without the necessity of carrying the oxygen required for that portion of the flight. This requires a multi-cycle design.

This intent of this project was to carry out the design and computational fluid dynamic (CFD) analysis of a combined cycle scramjet / rocket. The intent was that the craft would operate in scramjet mode as long as sufficient oxygen could be ingested to provide significant thrust. Once sufficient oxygen is no longer available, the inlet would close and the craft would switch to onboard oxygen, operating as a rocket.

The design and analysis process devolved into four basic steps: preliminary design of the craft, selection of a precise geometry using 2D calculations, creation of a grid for analysis, and analysis of the model using the open source FORTRAN 90 CFD code "hyp" developed by Dr. Eric Perrell.[1]

Due to some of the complex geometries present within the grid, some modifications to hyp were necessary. There was considerable overlap between the development of the grid and modification and testing of the CFD code. As the author's understanding of the code improved, and as the capabilities of the code were enhanced, changes were made within the grid.

This document is divided into several sections to discuss each of the above processes separately.

2. Preliminary Design:

Brief overview of scramjets

The supersonic combustion ram jet (SCRAM jet) is, geometrically, the simplest air breathing propulsion system for aeronautical/aerospace applications yet developed. Unfortunately, there has been very little success demonstrating successful applications of the design.

Description

Scramjets differ from all other forms of propulsion in that the combustion within the engine takes place in a supersonic air stream. A scramjet consists of an inlet, a combustion chamber, and a nozzle. No moving parts are required within the engine. Fuel is injected either at or close to the entrance of the combustion chamber. Due to the finite speed of reaction, it may be possible to introduce the fuel before the combustion chamber, allowing shockwaves to help disperse the fuel into the air stream and, provided the shocks increase temperature enough, ignite the mixture. As the operation of the scramjet is almost completely dependent upon the characteristics of the flow entering the engine, the design of the aircraft body needs to be centered on the propulsion system.

Successful tests

Ground tests

Numerous tests have successfully been completed, demonstrating thrust generated by a scramjet in supersonic wind tunnels. A couple of instances are noted here:

- NASA's X-43A: This was an unmanned, hydrogen fueled scramjet. Testing was carried out using NASA's non-vitiated air supply wind-tunnel.*
- Pratt & Whitney: HyTech: This was a prototype of the NASA / US Air Force GDE-1 JP-7 fueled scramjet engine. The fuel was used as coolant for the hull. The heat, together with a catalyst, broke down the fuel to more combustible elements before injection. Test was

* http://facilities.grc.nasa.gov/htf/htf_caps.html

carried out from September 2002 to May 2003, operating at Mach 4.5 and Mach 6.5 in the GASL wind-tunnel.[2]

Flight tests

Only two successful in flight tests have been conducted with a scramjet generating positive thrust.

- On March 27, 2004, NASA's second unmanned X-43A was released from NASA's B52, mounted on a Pegasus missile. The missile accelerated the test vehicle and brought it up to its test altitude, then detached. The scramjet operated for 10 seconds, accelerating the test vehicle to Mach 7.
- On November 16, 2004, the third X-43A craft was launched in the same manner. This craft reached approximately Mach 9.8 flying at 33.5km (110,000 feet).[†]

Combined cycles

A number of proposals have been made concerning the use of "combined cycle" propulsion systems, utilizing different means of propulsion at different flight conditions. One design, proposed by Kanda and Kudo[3] describes a 4 (propulsion) cycle SSTO (single stage to orbit) craft, switching between ejector-jet, ramjet, scramjet, and rocket modes. The design calls for a fixed geometry, based around a modification of the conventional scramjet models. Rocket engines supply part of the flow and thrust in all three configurations. Hydrogen was used as the fuel. Calculations of the airflow and combustion within the combustion chamber

[†] http://www.nasa.gov/missions/research/x43-main.html

were carried out using 1D analysis. Flow in the nozzle was calculated with 2D Prandtl-Meyer expansion waves.

Fuels

Hydrogen

Hydrogen is possibly the most efficient fuel, from a pure combustion point of view. Additionally, hydrogen/oxygen combustion has the benefit that aside from trace amounts, the only combustion product is H_2O. The principle disadvantages of H_2 as a fuel are storage requirements and production and transport costs. Additionally, even liquefied, H_2 has a low density, requiring large and heavy storage tanks.

Kerosene

Kerosene fuels, including jet propulsion fuel JP-1 and rocket propulsion fuel RP-1, are not defined by a single chemical species, but a mixture of species, defined as much by the distillation process as by the actual chemical composition. The simplicity of storage and easy availability of these fuels makes them an attractive option for propulsion. Combustion of such complex hydrocarbons can be very slow. However, by heating up the fuel (by using the fuel as a coolant for the craft's hull) and then passing the fuel through a catalyst, the hydrocarbon fuels can be broken down into simpler combustibles which react much more quickly. These fuels tend to build up deposits along the cooling passages, reducing the thermal conductivity and decreasing flow, over time.[4]

Both global reaction rates and reaction mechanisms for kerosene fuels have proved very difficult to find in the literature:

- Reference 5 includes a quasi-global reaction mechanism for RP-1, using 17 steps and 10 species.
- Reference 6 lays out a reaction mechanism for JP-1 (modeled as a mixture of n-decane and toluene) using 167 reactions and 63 species

Propane

Propane, in liquid form, has a density close to kerosene fuels, with a slight advantage in maximum theoretical I_{sp}. Cooling and compression requirements are modest compared with hydrogen.[4] At the same time, its high volatility allows propane to vaporize quickly, allowing the fuel to enter the air stream in gaseous form (or if entering in liquid form, to vaporize very quickly) minimizing or eliminating one step in the combustion process, permitting faster combustion. The time required to complete combustion becomes critical for reactions taking place in a supersonic air stream.

Propane was ultimately selected as the fuel for this design because it makes a nice compromise between the benefits and drawbacks of kerosene fuels and hydrogen.

Design approach

Similar to the design laid out by Kanda and Kudo, a multi-cycle propulsion system was considered, allowing for the most efficient operation at the varying speeds and ambient air pressures involved, going from horizontal take-off to orbit. A final design would incorporate at least a subsonic/supersonic jet engine, scramjet propulsion, and rocket propulsion. For this project, it was decided to limit the configurations considered to the scramjet and rocket cycles, leaving the details of the conventional jet cycle for later.

Due to the difficulties that have been experienced working with scramjets to date, the bulk of this paper concerns the scramjet cycle.

The general dimensions of the Kanda and Kudo design were used as a starting point for this design process.

Design options considered

Full versus half nozzle

Generally, scramjets are modeled using a half-nozzle, figure 1a. The lower surface of the aft end of the craft acts as a nozzle surface. No other physical boundaries constrain the flow leaving the combustion chamber.

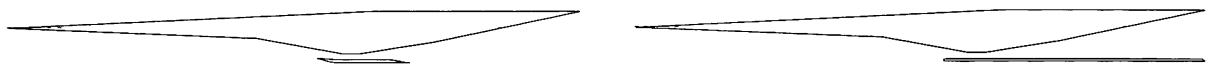


Figure 1: Full vs half nozzle configurations

Typically, the half nozzle configuration is used as a mechanism for reducing weight and drag. Because the flow is supersonic, after a certain distance from the throat, the cowl no longer has any impact on the flow felt at any point along the nozzle upper surface.

For the application considered, the nozzle is expected to deliver maximum thrust both in rocket propulsion mode and throughout the whole range of operating Mach numbers for the scramjet mode. Determining where the aft end of the cowl could be truncated without affecting the thrust generated requires knowing the flow conditions throughout the nozzle for all operating conditions at which the nozzle is expected to produce thrust.

Because these flow conditions cannot be properly predicted before analysis of the model is complete, a full nozzle (fig. 1b) was used in this project. The full nozzle is formed by extending the combustion chamber cowl and walls aft, to the end of the craft. For the two dimensional preliminary calculations, this configuration made it easier to approximate thrust.

Double versus single combustion chamber

Typically, a single combustion section, whether divided into multiple combustors or used as a single section, is considered for scramjets, with the combustion chamber built into the bottom surface of the aircraft. For this project, the effect of using two combustor sections, built into the upper and lower surfaces, was investigated.

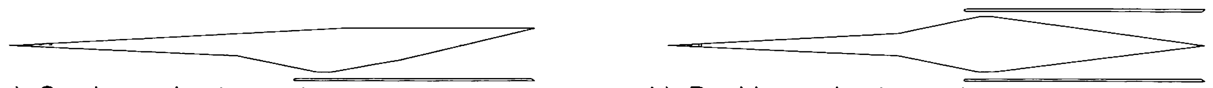

a) Single combustor section b) Double combustor section

Figure 2: Single vs double combustor section configurations

Variable geometry inlet ramps

In order to maximize the amount of air ingested by the scramjet, it is desirable to ensure that the shock leading into the jet lands on the lip of the combustion chamber. In the literature surveyed, two methods were used to ensure that this condition was met at varying Mach numbers:

1. With a fixed geometry inlet, the attitude of the craft must be adjusted. Particularly for the double combustion chamber configuration, this would be ineffective. Additionally, it was felt that separating the control of the incoming shockwave from attitude of the craft would permit greater efficiency in the aircraft design and operation.
2. An adjustable flap can be built into the leading edge of the cowl. The leading edge of this flap would angle inward to reach the shock. While this mechanism meets the requirement of placing the shock on the cowl inlet, it does so at the expense of reducing mass capture area in most flight conditions.[7]

An alternative which should avoid the problems with the above two methods was considered. In this method, an articulated pair of ramps joins the bow to the combustion chamber. The articulation permits the forward ramp to be adjusted so that the shock starting at this ramp can be placed on the lip of the combustion chamber at any flight Mach number within the operational range of the craft (fig. 3).

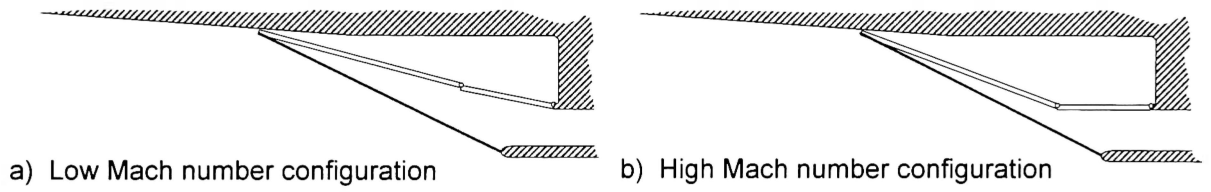
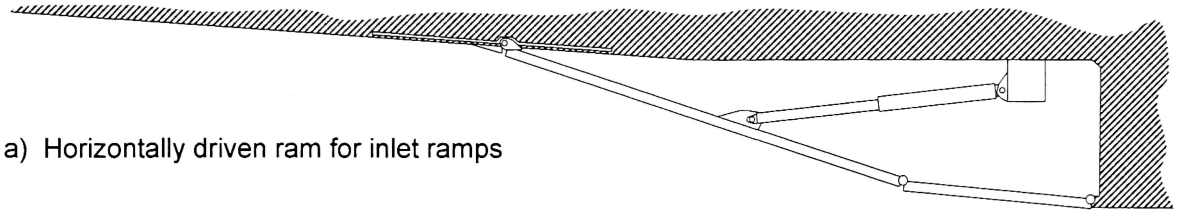
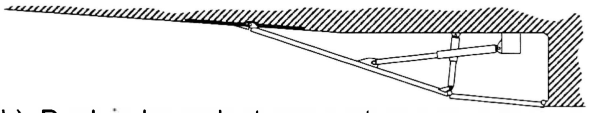

a) Low Mach number configuration
b) High Mach number configuration

Figure 3: Inlet ramps – low vs high Mach number configurations

To enable this movement between the two ramps, the two ramps are hinged together, with the hinged joints built into the webs of the supporting frames. The aft edge of the aft ramp is similarly hinged to an overhang at the start of the horizontal portion of the combustion chamber. With this arrangement, allowance must be made for movement of the forward ramp leading edge. This movement is accommodated with a series of sliding bearing plates that ride in grooved tracks milled out of a large frame welded to the bow shell plating. A short fixed angle ramp precedes the hinged connection between the lower ramp and the sliding connectors to protect the gap at the foot of the lower ramp (fig. 5).

The ram driving the ramps could run either vertically or horizontally. Since a horizontally driven ram would impose lower loads on the sliding bearing, this was the preferred arrangement. Use of a horizontal ram imposes a practical limitation on the movement of the aft ramp. At the fully extended position (lowest flight Mach number), the two ramps, ideally, should be in line, sharing a common angle with respect to the craft's longitudinal axis. As the ram draws back, the angle of the aft ramp could be permitted to increase or decrease. If both directions were used, a second set of rams would be required to control the direction that the aft ramp moved. Rather than impose additional control requirements, the aft ramp was originally constrained such that it can move, in scramjet mode, only between the horizontal and the same angle as the lower ramp. This requirement is checked in the 2D analysis.


a) Horizontally driven ram for inlet ramps


b) Dual, independent ram system concave outward

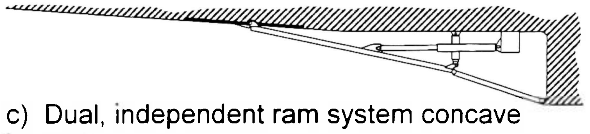

c) Dual, independent ram system concave inward

Figure 4: Ramp control arrangements

As calculations progressed, it became increasingly apparent that this restriction on the movement of the ramps considerably reduced the available thrust. The restriction was consequently removed, with the acknowledgement that the second set of rams would be necessary for proper control of the inlet.

Switching over to rocket mode, the ramps are brought up to the stops, so that the upper ramp seals off the combustion chamber inlet.

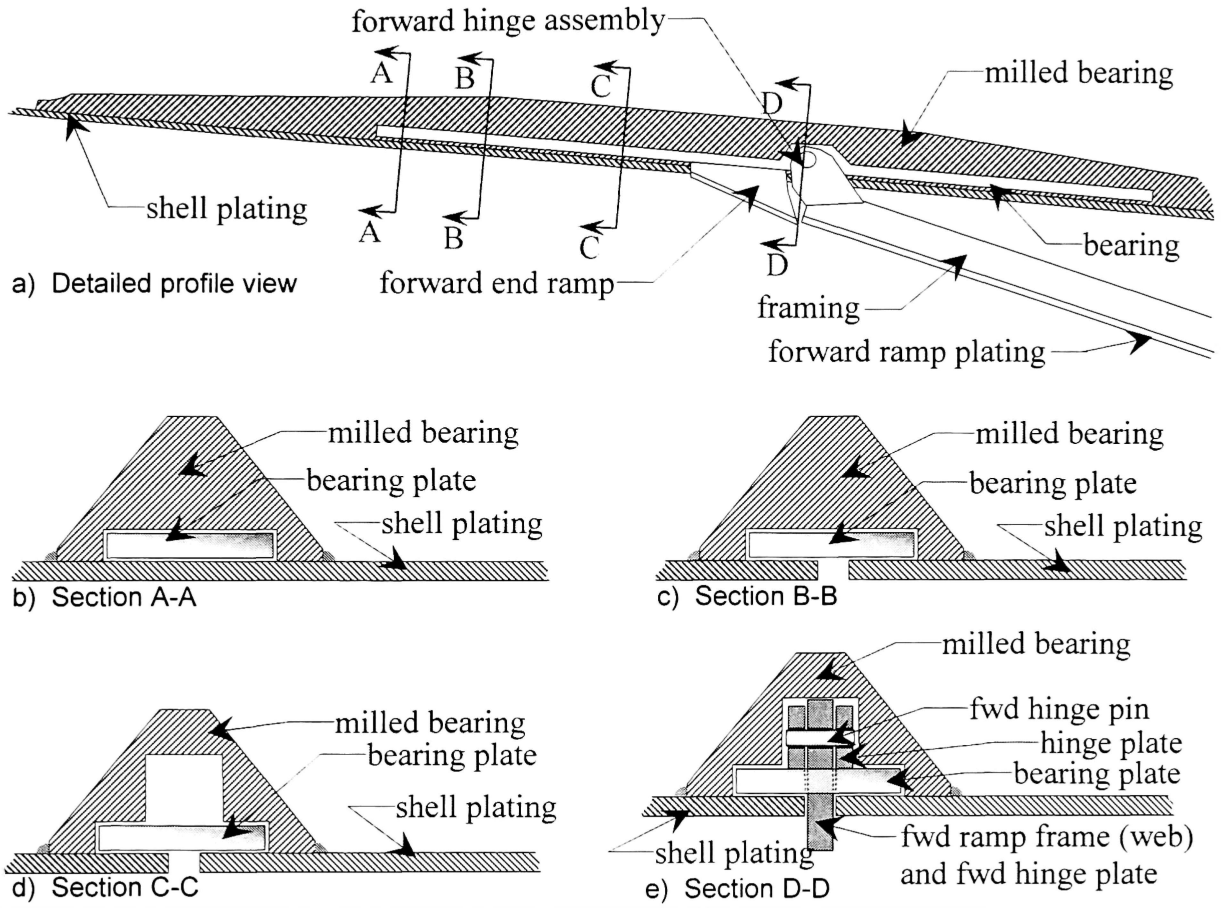


Figure 5: Forward inlet ramp leading edge - pivot/bearing plate arrangement

Variable geometry nozzle

While the flow entering the nozzle in scramjet mode is supersonic, and a diverging nozzle is required to accelerate the flow, in conventional jet and rocket modes, the flow entering the nozzle will be subsonic. A converging-diverging nozzle is required to accelerate the flow, when in these modes, to supersonic velocities. A variable geometry nozzle is required to accommodate the desired modes of operation.

To accomplish this variable geometry, a second set of articulated ramps, operating on the same principle as the inlet ramps, are used at the combustion chamber exit. In scramjet mode, the forward ramp is flush with the combustion chamber wall and the aft ramp rests flat on top of the nozzle plating. For the rocket configuration (and the conventional jet configuration if/when it is incorporated), a ram drives the two ramps downward at the hinged connection. The aft end of the aft ramp pivots on a bearing plate which slides forward inside a milled frame, dragging a cover plate behind to seal the slits in the nozzle plating. Figure 6, below, demonstrates the mechanism for varying the nozzle geometry.

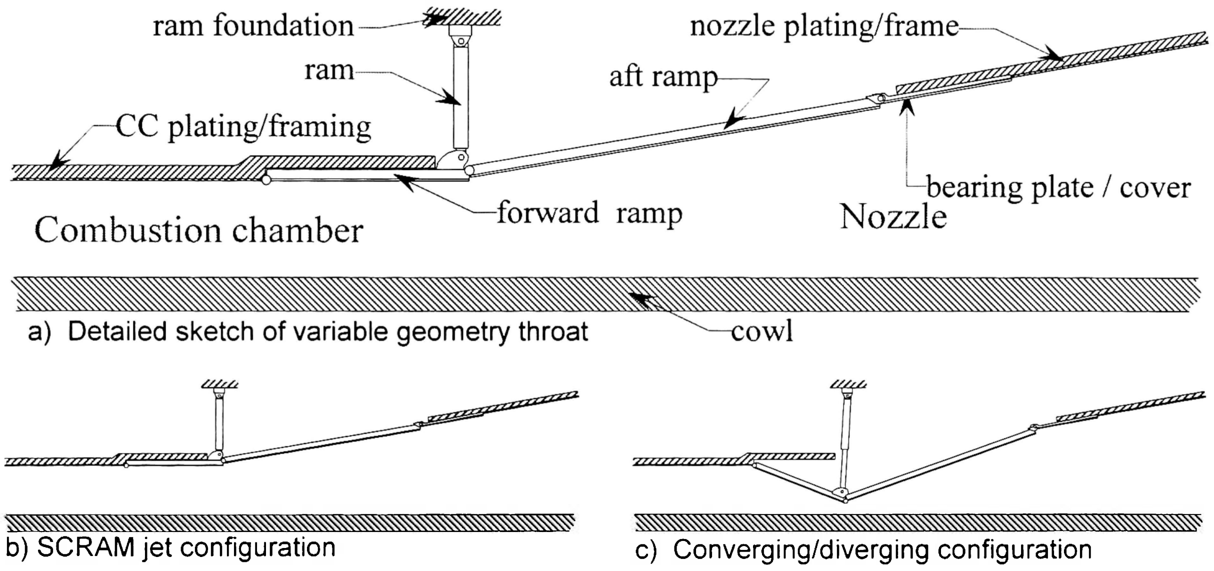


Figure 6: Variable geometry nozzle

Cowl support structure

In order to provide support for the combustor cowl, baffles are installed, running the length of the combustor and nozzle, dividing the combustor and nozzle across the craft's width. In order to accommodate the articulated inlet ramps, the internal baffles are raked backward from the cowl lip. This structure can be seen in figure 7. This arrangement resulted in complex geometries when the grid for CFD analysis was generated.

The air blocks below the craft must mesh smoothly with the blocks to the side of the craft. All blocks used with the CFD code HYP must be structured blocks, requiring 6 sides, each opposite pair having equal dimensions. Looking in two dimensions at the air blocks within the inlet region (between the forward raked side supports and ahead of the internal baffles), these blocks are constrained to a three sided region defined by the hull, the leading edges of the internal baffles, and the lower edges of the side supports. In order to produce the structured blocks required, the three sided region needed to be broken up into several 4 sided regions (figure 8a). This subdivision resulted in rotated local axes and forced some peculiar dependencies upon the dimensions of the block sides (figure 8b).

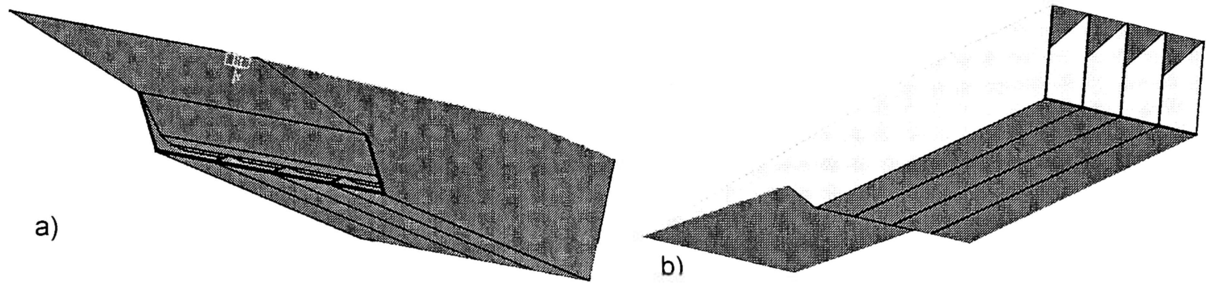


Figure 7: Cowl supporting internal baffles

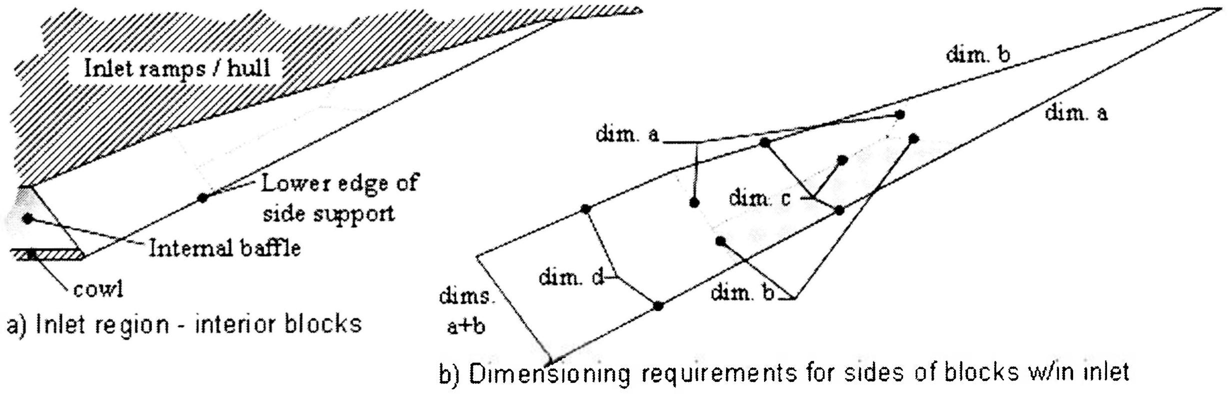

Inlet ramps / hull
Lower edge of side support
Internal baffle
cowl
a) Inlet region - interior blocks
dim. b
dim. a
dim. a
dim. c
dim. d
dim. b
dims. a+b
b) Dimensioning requirements for sides of blocks w/in inlet

Figure 8: Grid complications in way of inlet region

Ignition

Generally, elevated temperatures behind shocks generated at the bow, inlet ramp, or fuel injection jets are expected to initiate combustion. Temperatures behind the bow shocks and inlet ramps calculated in the process outlined in the following sections were found to be fairly low, insufficient to result in immediate ignition. To get around this, and reduce operational dependence upon flight condition, two options are considered as alternative or supplemental means of ignition:

- A small, forward facing step or block (fig. 9a) could be installed, near the front of the combustor, resulting in a small normal shock, with the concurrent significant temperature increase behind the shock. With a small enough step, the shock would dissipate fairly quickly, without affecting the majority of the flow within the combustor. The baffles inside the combustor/nozzle may also result in shocks strong enough to initiate combustion. This means of ignition is still dependent upon flight condition.
- A set of continuously sparking electrical ignitors (fig. 9b) located in each combustor section would provide regions of significantly elevated temperatures irrespective of the flight condition. Provided the gap used is small, the cost in electrical power would be fairly minor for a large craft compared with other internal electrical loads.

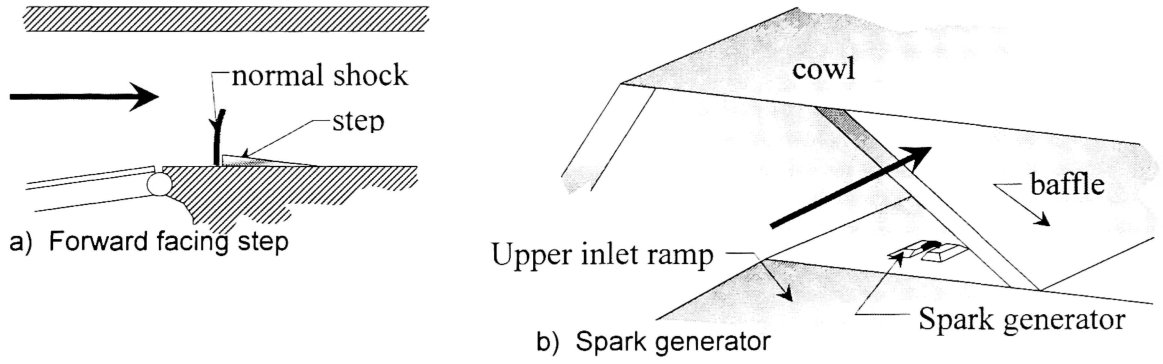


Figure 9: Ignition initiating devices

3. Two Dimensional Analysis

The preceding section describes the general mechanisms required in a variable geometry, combined cycle, scramjet / rocket. It gives little assistance in working out the precise geometry required to make the system work or the angles required at each ramp for any given flight condition. While the intention was to carry out analysis of designs using a Computational Fluid Dynamics (CFD) code, another, quicker, method was needed to select an appropriate geometry and develop a first approximation for the angles required. A two-dimensional analysis, combining a very large Excel spreadsheet with Excel Visual Basic automation, was developed to meet this need.

The 2-D analysis developed permits, depending upon the speed of the computer, twenty or more geometries to be evaluated in an hour or less. The output, in the form of tables and plots of net thrust and temperature and pressure at selected points throughout the flow path, can be used to tailor the final geometry to conform to specific mission requirements, in terms of operational flight Mach numbers, peak thrust Mach number, operating temperatures, etc.

The following sections detail the calculations used for a single selected set of flight and fixed geometric parameters. The analysis was set up to, at the touch of a button, automatically carry out these calculations for a selected set of fixed geometric parameters at a selected altitude, from Mach 5 to Mach 15 at Mach 1 increments.

Selection of inlet ramp configurations

For any specified geometry and flight condition, only one configuration of the inlet ramps (ε_2, ε_3) will place the shock from the lower ramp leading edge on the lip of the combustion chamber cowl. Figure 10 lays out the additional parameters needed to calculate the required angles between the forward ramp and the hull (ε_2) and between the two inlet ramps (ε_3).

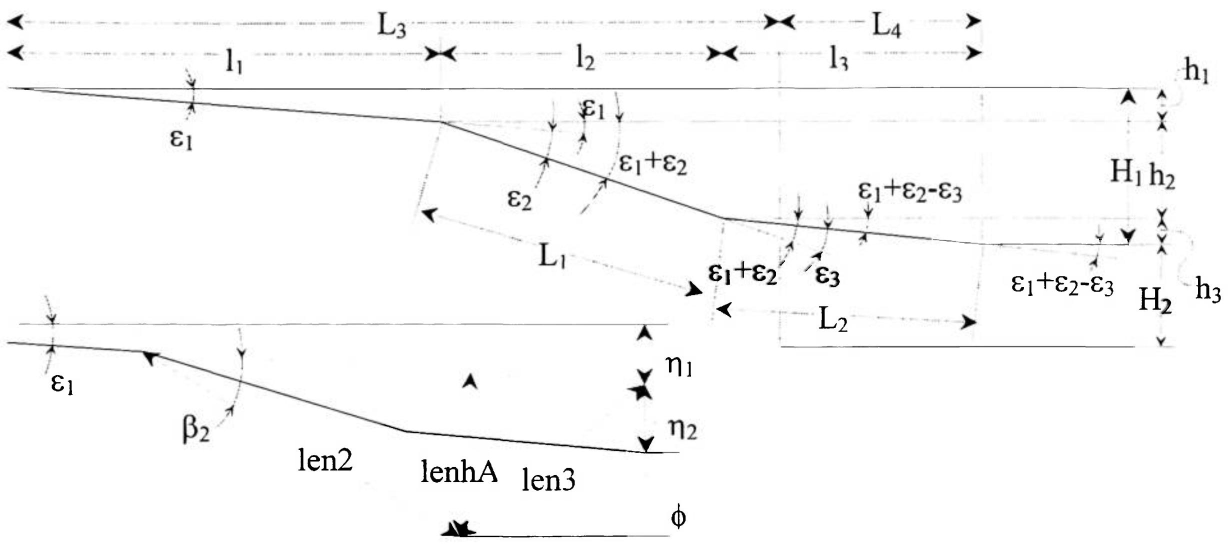


Figure 10: Inlet geometry calculated parameters

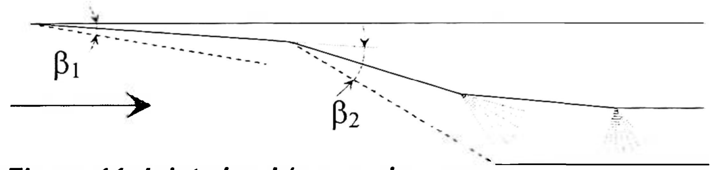


Figure 11: Inlet shock/expansion waves

When wave angle β_2, calculated from oblique shock theory, is equal to the angle of a line drawn from the forward ramp leading edge to the cowl lip, calculated from the geometry, minus the bow angle (ε_1), the assumed ε_2 and ε_3 must be correct.

Selection of ε_2 and ε_3 is an iterative process. An initial value of ε_2 is assumed. Based upon the following calculations, this value is iteratively adjusted. For the incoming air stream, standard values for γ and R are used:

$\gamma = 1.4$

$R = 287$ J / (kg K)

Oblique shock calculations - Bow shock

(Equations 2 - 6 are found in Anderson[8], chapter 4.)

Angle of attack of the craft is not fully implemented throughout the spreadsheet. For this reason, α is left as 0 degrees.

$$\theta_{1,bottomCC} = \varepsilon_1 - \alpha \tag{1}$$

While the wave angle β is normally found by an iterative calculation of the equation:

$$\tan(\theta) = 2 \cdot \cot(\beta) \cdot \left[\frac{M_1^2 \cdot \sin^2(\beta) - 1}{M_1^2 \cdot (\gamma + \cos(2\beta)) + 2} \right] \tag{2}$$

From this equation, Anderson derives three auxiliary equations, which yield an explicit solution for the wave angle:

$$\lambda = \sqrt{(M^2 - 1)^2 - 3 \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M^2\right) \cdot \left(1 + \frac{\gamma + 1}{2} \cdot M^2\right) \cdot \tan^2(\theta)}$$

$$\chi = \left\{ (M^2 - 1)^3 - 9 \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M^2\right) \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M^2 + \frac{\gamma + 1}{4} \cdot M^4\right) \cdot \tan^2(\theta) \right\} \Big/ \lambda^3$$

$$\tan(\beta) = \frac{M^2 - 1 + 2 \cdot \lambda \cdot \cos\left[(4\pi \cdot \delta + a\cos(\chi)) \big/ 3 \right]}{3 \cdot \left(1 + \frac{\gamma - 1}{2} \cdot M^2\right) \cdot \tan(\theta)} \tag{3}$$

The delta (δ) in this expression is 1 for the weak shock solution and 0 for the strong shock. The weak shock solution can be safely assumed. β can be obtained directly from the above 3 equations.

From the standard oblique shock relations:

$$M_{n1} = M_1 \cdot \sin(\beta_1) \tag{4}$$

$$M_{n2} = \sqrt{\frac{M_{n1}^2 + 2/(\gamma - 1)}{[2 \cdot \gamma \cdot M_{n1}^2/(\gamma - 1)] - 1}} \tag{5}$$

$$M_2 = M_{n2} / \sin(\beta_1 - \theta_1) \tag{6}$$

The Mach number (M_3) leaving the shock at the lower ramp, as well as the wave angle (β_2), are calculated similarly.

Geometry

Independent of ε_2 and ε_3:

$$\eta_1 = (L_3 + L_4) \cdot \tan(\varepsilon_1) \tag{7}$$

$$\eta_2 = (H_1 - \eta_1) \tag{8}$$

$$\phi = a\tan\left({(H_2 + \eta_2)} \big/ {L_4} \right) \tag{9}$$

Dependent on ε_2 and ε_3 (ε_2 is iteratively adjusted):

$$l_2 = L_1 \cdot \cos(\varepsilon_1 + \varepsilon_2) \tag{10}$$

$$h_2 = L_1 \cdot \sin(\varepsilon_1 + \varepsilon_2) \tag{11}$$

l_3 is found by iterative calculation of the following equation (eq. 12):

$$l_3' = \sqrt{L_2^2 - (H_1 - h_2 - \tan(\varepsilon_1) \cdot (L_3 + L_4 - l_2 - l_3))^2} \tag{12}$$

$$h_3 = \sqrt{L_2^2 - l_3^2} \tag{13}$$

$$h_1 = H_1 - h_2 - h_3 \tag{14}$$

$$l_1 = h_1 / \tan(\varepsilon_1) \tag{15}$$

$$\varepsilon_3 = \varepsilon_1 + \varepsilon_2 - a\sin(h_3 / L_2) \tag{16}$$

$$len2 = \sqrt{(L_3 - l_1)^2 + (H_1 + H_2 - h_1)^2} \tag{17}$$

$$len3 = L_4 / \cos(\phi) \tag{18}$$

$$lenhA = len3 \cdot \sin(\varepsilon_1 + \phi) \tag{19}$$

$$\beta_2 = a\sin(lenhA / len2) \tag{20}$$

If the β_2 calculated by geometry is greater than that calculated by shock relations, ε_2 needs to be increased. If the value calculated from geometry is less, ε_2 needs to be decreased.

Combustor length

(Equations 22, 23, 26, and 27 are found in Anderson[8] *chapter 4. Equations 30 - 34 are found in Turns*[9] *chapter 2. Equation 37 is found in Turns*[9] *chapter 5.)*

In order to calculate the minimum length of the combustion chamber necessary to complete combustion, pressure and temperature entering the combustion chamber must be known.

Free-stream conditions were selected based upon the atmospheric pressure used in the Kanda/Kudo calculations. $P_\infty = 0.02566$ atm. T_∞ was interpolated, from this pressure, using atmospheric tables[‡].

Pressure and temperature following each shock were calculated from normal shock relations. Expansion fans develop at the junction of the two inlet ramps and at the junction between the aft ramp and the straight section of the combustion chamber. The Mach number following each expansion fan is found by an iterative solution of the Prandtl-Meyer function. For the first expansion fan:

$$\theta_4 = \varepsilon_3 \tag{21}$$

$$\nu_{M3} = \sqrt{\frac{\gamma+1}{\gamma-1}} \cdot a\tan\left(\sqrt{\frac{\gamma-1}{\gamma+1}} \cdot \left(M_3^2 - 1\right)\right) - a\tan\left(\sqrt{M_3^2 - 1}\right) \tag{22}$$

$$\nu_{M4} = \theta_4 + \nu_{M3} \tag{23}$$

$$M_4' = \sqrt{1 + \frac{\gamma+1}{\gamma-1} \cdot \tan^2\left[\sqrt{\frac{\gamma-1}{\gamma+1}} \cdot \left(\nu_{M4} + a\tan\left(\sqrt{M^2 - 1}\right)\right)\right]} \tag{24}$$

M_4 is calculated iteratively from eq. 24 (derived from eq. (22)). The Mach number (M_5) from the second expansion fan is calculated similarly, using M_4 from the first expansion fan and:

$$\theta_5 = \varepsilon_1 + \varepsilon_2 - \varepsilon_3 \tag{25}$$

[‡] http://www.usatoday.com/weather/wstdatmo.htm

Knowing that total pressure and temperature do not change through expansions, static pressure and temperature after each expansion fan can be calculated:

$$T_4 = T_3 \cdot \frac{1 + \frac{1}{2}(\gamma - 1) \cdot M_3^2}{1 + \frac{1}{2}(\gamma - 1) \cdot M_4^2} \tag{26}$$

$$P_4 = P_3 \cdot \left[\frac{1 + \frac{1}{2}(\gamma - 1) \cdot M_3^2}{1 + \frac{1}{2}(\gamma - 1) \cdot M_4^2} \right]^{\gamma/(\gamma-1)} \tag{27}$$

T_5 and *P_5* are calculated in the same manner. Velocity entering the combustion chamber can be found:

$$V_5 = M_5 \cdot \sqrt{\gamma \cdot R \cdot T_5} \tag{28}$$

As noted above, propane (C_3H_8) was selected for the fuel for this calculation. The 2-D analysis spreadsheet/program is set up to accommodate any hydrocarbon fuel comprised only of hydrogen and carbon atoms, provided that a single set of global reaction rate coefficients of the same form as used currently (eq. (37)) could be provided. A fuel oxidizer ratio (ϕ) must be assumed:

$$\varphi = \frac{(m_{ox}/m_F)_{stoich}}{(m_{ox}/m_F)} \tag{29}$$

For simplicity, this value was assumed to be one. (The calculation is not currently set up to accommodate any other value of ϕ, though only minimal changes would be required to rectify this.) A simplified air mixture, comprised of 21% O_2 and 79% N_2 is used. Using ϕ, the number of moles (N_i) and molar concentrations ($[N_i]$) of O_2, N_2, and C_3H_8 are found by stoichiometry. For adiabatic flame temperature calculation, complete combustion is assumed, with only CO_2 and H_2O as products.

Enthalpy of formation ($\bar{h}^0_{f,i,Tref}$) and curve-fits for specific enthalpy ($\bar{h}_{i,T}$) for the fuel as well as thermodynamics tables for O_2, N_2, CO_2, and H_2O were taken from Anderson[8]. Enthalpies of formation are taken at 298 K.

An adiabatic flame temperature must be assumed. For incoming air temperatures (T_5) and the assumed adiabatic flame temperature, changes in sensible enthalpies ($\Delta\bar{h}_{s,i}$) and constant pressure specific heats (C_{Pi}) are interpolated from the thermodynamics tables for all species except the fuel. Specific enthalpies are then:

$$\bar{h}_{i,T} = \bar{h}^0_{f,i,Tref} + \Delta\bar{h}_{s,i,T} \tag{30}$$

Specific enthalpy and C_P of the fuel are calculated from the curve-fits. Based upon stoichiometry, the assumption of complete combustion, with CO_2 and H_2O as the only products, the product molar concentrations are calculated. For $\phi = 1$, all O_2 and C_3H_8 are consumed. Number of moles of N_2 is unchanged.

For constant pressure combustion, total enthalpy of the products and reactants must be equal:

$$H_{react} = \sum_{react} \bar{h}_i \cdot N_i = \sum_{react} \left(\bar{h}^0_{f,i,Tref} + \Delta\bar{h}_{s,i,T} \right) \cdot N_i \tag{31}$$

$$H_{prod} = \sum_{prod} \bar{h}_i \cdot N_i = \sum_{prod} \left(\bar{h}^0_{f,i,Tref} + \Delta\bar{h}_{s,i,T} \right) \cdot N_i \tag{32}$$

$$\Delta\bar{h}_{s,i,T} = \int_{T_{ref}}^{T} C_{P,i} dT \tag{33}$$

Assuming constant C_P:

$$\Delta\bar{h}_{s,i,T} = C_{P,i} \cdot \left(T - T_{ref} \right) \tag{34}$$

From this, a new adiabatic flame temperature can be calculated:

$$T_{ad} = 298K + \frac{\sum_{react} N_i \cdot \left(\bar{h}^0_{f,i,298K} + \Delta\bar{h}_{s,i,T} \right) - \sum_{prod} N_i \cdot \bar{h}^0_{f,i,298K}}{\sum_{prod} N_i \cdot C_{P\,i}} \tag{35}$$

This value of adiabatic flame temperature is input as a new approximation, and the calculation is iterated until it converges.

An average temperature for the combusting region, is assumed:

$$T_{avg} = (T_{ad} - T_5)/2 \tag{36}$$

In order to estimate time to complete combustion (t_c), reaction rates for combustion were required. Any moderately detailed combustion mechanism for propane would be prohibitively complex, provided that it could be found at all. Parameters (A, $E_a/(Ru\ T)$, m, n) for a single step global reaction rate for propane/air combustion are found in Turns[9], corresponding to the equation:

$$\frac{d[C_xH_y]}{dt} = -A \cdot e^{-E_a/_{Ru \cdot T}} [C_xH_y]^m \cdot [O_2]^n \tag{37}$$

Integrating this equation, and setting $[C_xH_y] = 0$ for $t = t_c$, time to complete combustion (t_c) can be approximated:

$$t_c = \frac{[C_xH_y]_{init}^{1-m}}{(1-m) \cdot A \cdot e^{-E_a/_{Ru \cdot T}} \cdot [O_2]_{init}^n} \tag{38}$$

Length of combustion is then the product of the velocity of the flow and t_c:

$$l_c = V_5 \cdot t_c \tag{39}$$

This value can be used to set a minimum acceptable length of the combustion chamber. This assumes reaction rather than mixing limited combustion. Typical l_cs found in this manner ranged from 0.005 m to 0.03 m. Finding that this length of combustion chamber would be unnecessarily small for a craft on the order of 70 meters long, the length of the combustion chamber was instead taken from the Kanda/Kudo model. 2.5 meters was selected. This provides a large margin of safety to allow for mixing as well as providing leeway for errors resulting from the numerous assumptions used in calculating length of combustion.

Thrust

(Equations 44, 47, 48, and 49 are found in Anderson[8] chapter 3. Equation 50 is found in Anderson[8] chapter 2. Equations 40 and 41 are found in Turns[9] chapter 2.)

Mixture C_P, molecular weight ($\overline{M}_{mix}$), gas constant (R), and specific heat ratio (γ) for the mixture entering and exiting the combustion chamber are found as follows:

$$C_{P,mix} = \sum C_{P,i} \cdot N_i \Big/ \sum N_i \tag{40}$$

$$\overline{M}_{mix} = \sum \overline{M}_i \cdot N_i \Big/ \sum N_i \tag{41}$$

$$R = Ru / \overline{M}_{mix} \tag{42}$$

$$\gamma = \frac{C_{p,mix}}{C_{p,mix} - R} \tag{43}$$

C_P and R must use the same units. Using the thermodynamics tables in Turns[9], $C_{P\,mix}$ must be divided by $\overline{M}_{mix}$ to correct units.

Using T_{ad} as the temperature exiting the combustion chamber, and the relation for change in density due to heat addition in a constant area duct:

$$\frac{\rho_5}{\rho_{exit}} = \left(\frac{1 + \gamma_5 \cdot M_5^2}{1 + \gamma_{exit} \cdot M_{exit}^2} \right) \cdot \left(\frac{M_{exit}^2}{M_5^2} \right) \tag{44}$$

From continuity, again assuming constant area duct:

$$\rho_5 \cdot V_5 = \rho_{exit} \cdot V_{exit} \tag{45}$$

Remembering: $M_{exit} = V_{exit} \Big/ \sqrt{\gamma_{exit} \cdot R_{exit} \cdot T_{exit}}$

$$V_{exit}^2 - V_{exit} \cdot \frac{V_5 \cdot \left(1 + \gamma_{in} \cdot M_{in}^2\right)}{\gamma_{exit} \cdot M_{exit}^2} + R_{exit} \cdot T_{ad} = 0 \tag{46}$$

This can be solved easily with the quadratic equation.

From another relation for constant area duct with heat addition:

$$P_{exit} = P_5 \cdot \left(\frac{1 + \gamma_{exit} \cdot M_{exit}^2}{1 + \gamma_5 \cdot M_5^2} \right) \tag{47}$$

ρ_{exit} is calculated from the equation of state: $\rho = \dfrac{P}{R \cdot T}$

Total exit pressure and density are calculated from static pressure, density, and the Mach number as follows:

$$P_0 = P \cdot \left(1 + \tfrac{1}{2}(\gamma - 1) \cdot M^2\right)^{\gamma/\gamma - 1} \tag{48}$$

$$\rho_0 = \rho \cdot \left(1 + \tfrac{1}{2}(\gamma - 1) \cdot M^2\right)^{1/\gamma - 1} \tag{49}$$

Throat height (h^*) is developed from the area-Mach number relation, assuming that the width of the flow path does not change.

An initial value is assumed for the Mach number (M_e) exiting the nozzle. This value is iteratively calculated from the area-Mach number relation. Where double combustion chambers are assumed, the exit height is the vertical distance between the combustion chamber cowl and the center-plane (H_1 + H_2). Where a single combustion chamber is used, the exit height is H_1 + H_2 + $H_{cp\text{-}top}$.

Static pressure and density at the nozzle exit are developed using the total pressure and total density relations. Static temperature is developed using the equation of state. Exit velocity is developed from the definitions of Mach number and speed of sound.

Thrust per unit width is developed from conservation of momentum. Looking at the flow path starting from the combustion chamber inlet:

$$\rho_e \cdot V_e^2 \cdot A_e - \rho_i \cdot V_i^2 \cdot A_i + P_e \cdot A_e - P_i \cdot A_i = T \tag{50}$$

$$\rho_e \cdot V_e^2 \cdot h_e - \rho_i \cdot V_i^2 \cdot h_i + P_e \cdot h_e - P_i \cdot h_i = T / width \tag{51}$$

Where two combustion chambers are used, this value would, of course, be multiplied by 2.

Form drag is comprised of the pressure forces acting on the forward surfaces in the axial direction. For the double combustion chamber design:

$$\begin{aligned}
\tfrac{1}{2}D_{form} &= P_2 \cdot A_{Bow} \cdot \sin(\varepsilon_1) + P_3 \cdot A_{fwdRamp} \cdot \sin(\varepsilon_1 + \varepsilon_2) + P_4 \cdot A_{aftRamp} \cdot \sin(\varepsilon_1 + \varepsilon_2 - \varepsilon_3) \\
\tfrac{1}{2}D_{form} / W &= P_2 \cdot (A_{Bow} / W) \cdot \sin(\varepsilon_1) + P_3 \cdot (A_{fwdRamp} / W) \cdot \sin(\varepsilon_1 + \varepsilon_2) + P_4 \cdot (A_{aftRamp} / W) \cdot \sin(\varepsilon_1 + \varepsilon_2 - \varepsilon_3) \\
\tfrac{1}{2}D_{form} / W &= P_2 \cdot L_{Bow} \cdot \sin(\varepsilon_1) + P_3 \cdot L_1 \cdot \sin(\varepsilon_1 + \varepsilon_2) + P_4 \cdot L_2 \cdot \sin(\varepsilon_1 + \varepsilon_2 - \varepsilon_3)
\end{aligned} \tag{52}$$

where: $$L_{Bow} = \frac{L_3 + L_4 - L_1 \cdot \cos(\varepsilon_1 + \varepsilon_2) - L_2 \cdot \cos(\varepsilon_1 + \varepsilon_2 - \varepsilon_3)}{\cos(\varepsilon_1)} \tag{52a}$$

For the single combustion chamber design, the above value, plus the axial force acting on the upper surface, would comprise the full form drag. The pressure on the upper surface is calculated using the same oblique shock formulas that were discussed previously:

$$\begin{gathered}
D_{form} / W = \\
P_2 \cdot L_{Bow} \cdot \sin(\varepsilon_1) + P_3 \cdot L_1 \cdot \sin(\varepsilon_1 + \varepsilon_2) + P_4 \cdot L_2 \cdot \sin(\varepsilon_1 + \varepsilon_2 - \varepsilon_3) + P_{upper} \cdot L_{Bow,upper} \cdot \sin(\varepsilon_{1,top})
\end{gathered} \tag{53}$$

where: $$L_{Bow,upper} = \frac{H_{cp-top}}{\sin(\varepsilon_{1\,top})} \tag{53a}$$

Without knowing surface roughness, skin friction cannot be derived, however, a brief examination of the geometry suggests that there should be relatively little difference amongst the various possible configurations, with the exception that two combustor designs, having significantly greater surface area, would have significantly greater friction drag than the single combustor designs.

Error Checking

Geometric

If ε_3 is negative inside the operating Mach regime of the craft, the forward and aft ramps are joined at a concave downward angle. Since allowing the ramps to flex in downward in this manner would require a second set of rams, driven vertically, designs calling for this angle were initially discounted. This problem tends to occur at low Mach numbers.

For some configurations, the direct geometric calculation results in the aft ramp angling upwards toward the combustor horizontal section. This means that the leading edge of the ramp partially obstructs the flow. This is unacceptable in scramjet operation. This problem occurs at high Mach numbers.

Shocks inside the nozzle

The exit pressure was generally found to be lower than atmospheric pressure, requiring a shock to bring the pressure up. If a normal shock occurs inside the nozzle, the exit flow will be subsonic and the above thrust calculations would be invalid. The design would also be unacceptable. If oblique shocks occur outside the nozzle, initiating at the nozzle exit, the above thrust calculations are valid. This issue can be settled by assuming a normal shock at the nozzle exit, and calculating the pressure behind the shock using normal shock relations. If the pressure behind the exit shock is greater than the atmospheric pressure, no normal shock develops within the nozzle and a pair of oblique shocks develop at the exit.

Preliminary Design Assumptions

In the above analysis, a great number of assumptions, some fairly questionable, are made. Since the purpose of the analysis was strictly to select a design, rather than perform a thorough analysis, this was considered acceptable. The following are some of the critical assumptions made:

- Ideal gas: The equation of state was used by itself or implicitly in other equations used throughout the calculation. Generally, this is considered an acceptable assumption.
- Dissociation: Dissociation was not considered. While not a major issue below hypersonic speeds, aerodynamic heating at speeds above Mach 5 can result in significant dissociation of O_2, and even N_2 at high speeds and where normal shocks take place.
- Fine details of design: Shocks and expansions not detailed in the 2-D analysis may take place in way of some fine details of the design, such as the rounded leading edges of the bow and cowl, the forward faces of the cowl supporting baffles, the sides of the combustor, joints for the variable angle ramps, etc. Additionally, flow slipping to the side, off the bow plating, rather than traveling straight up to the combustor, will significantly affect the shape of the inlet shocks and expansions leading into the combustion chamber at the outsides.
- Global reaction rate: The use of a global reaction rate for propane, rather than a detailed reaction mechanism makes the calculated combustor length suspect.
- Constant C_P: Use of constant C_P to calculate adiabatic flame temperature is highly suspect, particularly given the drastic change in calculated temperature and pressure across the combustor. This assumption is, however, frequently used.

- Constant pressure combustion: The constant pressure assumption for calculation of T_{ad} was clearly inaccurate, though necessary to get an approximation.
- Combustor exit temperature: Use of T_{ad} as the combustor exit temperature is highly inaccurate, given the change in velocity and significant change in temperature across the combustion chamber. As an alternative, an attempt was made to modify the plug flow reactor calculation for use with global reaction rates, however the author was unable to obtain reasonable results by this method.

Results of the 2D analysis

Tables 1a, 1b, and 1c summarize the results of the last run of geometric variations (mods) tested after the last adjustments to the 2-D analysis. Plots of net thrust/width versus flight Mach number for selected mods are included in Appendix 1.

Table 1a: Summary of geometries tested

Mod:		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**10**
ε_1	°	3	3	3	3	3	3	3	3	3
L_1	m	5.3	6	5.3	5	5	5	5	5	5
L_2	m	1	1	2	2	2	2	2	2	2
L_3	m	35	35	35	35	35	33	30	28	28
L_4	m	1	1	1	1	0.5	2	2	2	2
H_1	m	3	3	3	3	3	3	3	3	3
H_2	m	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
$\varepsilon_{1,top}$	°	3	3	3	3	3	3	3	3	3
$H_{cp\text{-}top}$	m	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2
ε_3 **test**		fail	fail	fail	fail	fail	pass	pass	pass	pass
M_{max}		10	11	15	15	15	10	11	12	12
$M_{thr,single}$		10	11	15	15	15	10	9	8	7
$M_{thr,max,single}$		6	10	11	15	15	15	7	6	5
$Thr_{max,dbl}$	N/m	-13,219	-16,511	-14,743	-13,326	-21,370	-7,397	-10,923	-13,747	-13,747
$Thr_{max,single}$	N/m	47,229	68,335	109,075	97,338	119,588	32,235	24,043	18,763	12,261
$\Delta\varepsilon_{2,Mmax}$	°	0.458	0.375	0.149	0.143	0.151	0.423	0.298	0.221	0.221

Table 1b: Summary of geometries tested – continued

Mod:		**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
$\square_1$	o	3	3	3	3	3	3	2 7	2 9	3
L_1	m	5	5	5	5	5	5	5	5	5
L_2	m	2	2	2	2	2	2	2	2	2
L_3	m	28	31	31	31	31	30	30	30	35
L_4	m	2	2	2	2	2	2	2	2	1
H_1	m	3	3	3 5	3 1	2 8	2 9	2 9	2 9	3
H_2	m	0 5	0 5	0 5	0 5	0 5	0 5	0 5	0 5	0 5
$\varepsilon_{1,top}$	o	3	3	3	3	3	3	2 7	2 9	3
$H_{cp\text{-}top}$	m	2 9	2 7	2 7	2 7	2 7	2 7	2 7	2 7	2 5
ε_3 **test**		pass	pass	pass	pass	fail	pass	pass	pass	fail
M_{max}		12	10	15	11	9	10	11	10	15
$M_{thr,single}$		9	10	5	9	9	10	10	10	15
$M_{thr,max,single}$		6	6	6	5	6	9	7	6	7
$Thr_{max,dbl}$	N/m	-13,747	-9,652	-26,919	-12,409	-5,209	-8,338	-11,424	-9,288	-13,326
$Thr_{max,single}$	N/m	25,726	30,108	1,627	23,434	52,758	35,484	27,988	32,532	97,338
$\Delta\varepsilon_{2,Mmax}$	o	0 221	0 406	0 103	0 293	0 606	0 413	0 289	0 403	0 143

Table 1c: Summary of geometries tested - continued

Mod:		**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
ε_1	o	3	3 1	2 7	2 7	2 7	2 7	2 7	2 7
L_1	m	5	5	5	5	5	4 7	4 5	4
L_2	m	2	2	2	2	2	2	2	2
L_3	m	35	35	35	35	35	35	35	35
L_4	m	1	1	1	1	1	1	1	1
H_1	m	3	3	3	3	3	3	3	3
H_2	m	0 5	0 5	0 5	0 5	0 5	0 5	0 5	0 5
$\varepsilon_{1,top}$	o	3	3	3	3 1	2 9	2 9	2 9	2 9
$H_{cp\text{-}top}$	m	2 7	2 7	2 7	2 7	2 7	2 7	2 7	2 7
ε_3 **test**		fail	fail	fail	fail	fail	fail	fail	fail
M_{max}		15	15	15	15	15	15	15	15
$M_{thr,single}$		15	15	15	15	15	15	15	15
$M_{thr,max,single}$		6	15	15	15	15	15	14	12
$Thr_{max,dbl}$	N/m	-13,326	-12,596	-16,455	-16,455	-16,455	-15,280	-14,381	-13,496
$Thr_{max,single}$	N/m	116,459	128,644	76,854	76,269	77,426	62,161	53,410	37,229
$\Delta\varepsilon_{2,Mmax}$	o	0 143	0 148	0 129	0 129	0 129	0 124	0 120	0 112

Line drawings of the critical geometries in the inlet region for 3 selected mods are provided below to give a rough picture of the variety of geometries examined.

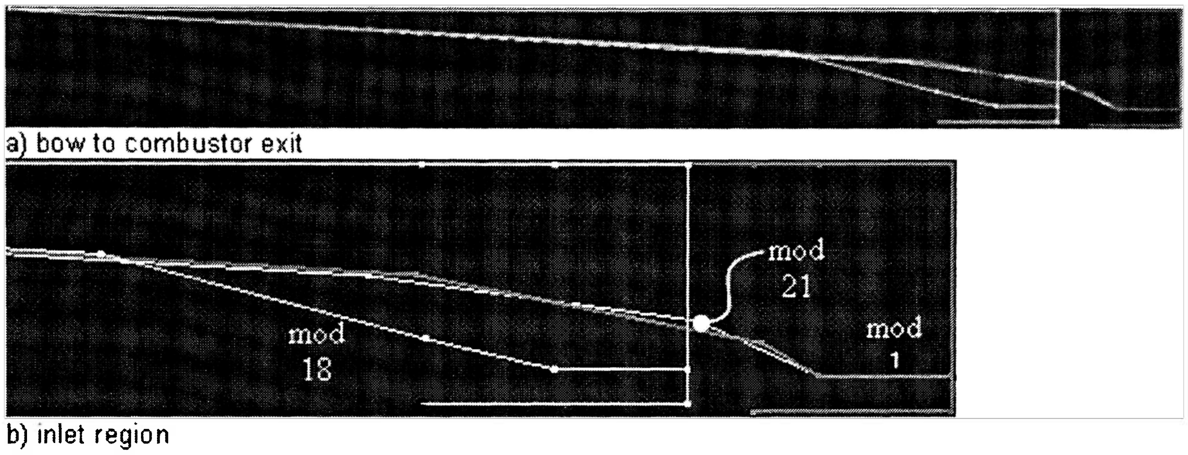


Figure 12: Line drawings of inlet regions of selected mods, mach 5 configurations

For all geometries, it was found that the use of double combustors resulted in less thrust being generated than form drag. It is assumed that this is the result of having smaller nozzle exit areas, not permitting the flow to fully expand. For much larger craft, craft with smaller combustion chambers, or when operating on a higher air/fuel ratio, where the nozzle exit area is great enough that a normal shock would be generated within the nozzle of a single combustor design, possibly the double combustor would prove to be effective.

Notation used in 2-D analysis results table

ε_3 test: The values in this row indicate compliance with the first geometric error test (i.e., passes with a positive or zero value of ε_3). As previously indicated, a negative value for ε_3 indicates that a second set of rams would be required to control the movement of the inlet ramps. Originally, this was deemed unacceptable. From the above tables, it can be seen that the requirement for a positive or zero ε_3 severely restricts both the available power and the range of positive thrust. As a consequence, this constraint was reconsidered and, while the value is still checked for, the final design used in the CFD calculations requires a second set of rams for control of the inlet ramps.

$\underline{M_{max}}$: The values in this row indicate the maximum Mach number at which the single combustor design was found to comply with the second geometric requirement (i.e., failure when the aft ramp partially obstructs flow into the combustor). Calculations were only carried out to Mach 15; hence, where $M_{max} = 15$, there is no indication that the design would fail the second geometric test at higher Mach numbers.

$\underline{M_{thr,single}}$: The values in this row indicate the maximum Mach number calculated at which the single combustor design continued to produce more thrust than form drag. Because calculations were not carried out beyond Mach 15, where $M_{thr,single} = 15$, the craft may or may not continue to generate positive net thrust above Mach 15.

$\underline{M_{thr,max,single}}$: The values in this row indicate the Mach number at which maximum thrust was generated for the single combustor design.

$\underline{Thr_{max,dbl}}$: The values in this row indicate the maximum net thrust (thrust minus form drag) calculated for the double combustor configuration. For this configuration, this value is net thrust per combustor (½ total net thrust).

$\underline{Thr_{max,single}}$: The values in this row indicate the maximum net thrust (thrust minus form drag) calculated for the single combustor configuration.

$\underline{\varepsilon_{2,Mmax}}$: The values in this row indicate the change in forward inlet ramp angle between the configuration at $M_{thr,single}$ and the configuration at the prior Mach number. This value is an indicator of the degree of control required in positioning the ramps at high Mach number. Additionally, as Mach number increases and dissociation behind the shock becomes significant, the shock is expected to curve inward, requiring higher angles at the forward ramp than are calculated in the 2-D analysis to force the shock wave to remain on the lip of the combustion chamber.

Compilation

The 2-D analysis spreadsheet/program was not initially set up for easy interpretation by any user other than the author, with useful output integrated directly into the rest of the calculations. Excel also has a limit to the number of columns that can be written, limiting the number of design modifications that can be stored in the spreadsheet at once. In order to produce a more comprehensible output and to store more than the 17 designs permitted in the analysis program, a function was built into the analysis program, allowing the user to append the output for any or all designs to a text file on the C drive. A separate spreadsheet/program was developed to extract this data into a series of tables and plots, with the scale of the plots automatically corrected.

Selected mod

Based upon the 2-D analysis, the single combustor version of mod 21 was selected for the CFD analysis. The single combustor version of mod 21 has the highest high end thrust. As can be seen in the Thrust vs Mach number plot below, the net thrust for mod 21 increases throughout the examined Mach number range, with every indication that it should continue to increase for some range past Mach 15.

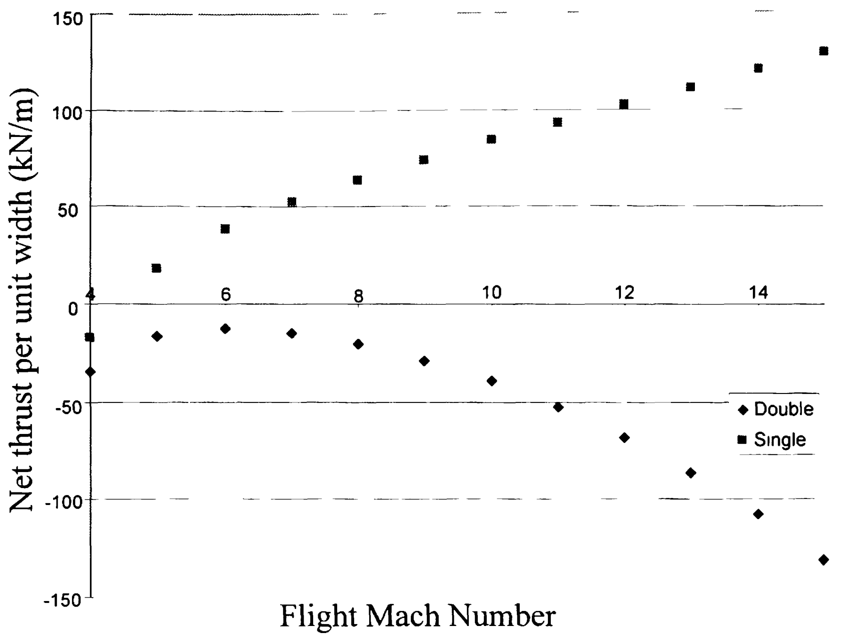


Figure 13: Net thrust vs flight Mach number for selected design@ design altitude

Additionally, while a very fine degree of control is required at the inlet ramps at high Mach number, the degree of control required for this mod is generally better than the one which was required for the other high thrust designs.

After mod 21 was selected, the automated loop used to run the calculation from Mach 5 to Mach 15 was modified to run up to Mach 20. Using the geometry from mod 21, the analysis was carried out again, varying altitude from sea level to 50 kilometers. This was done to identify an effective ceiling for operation in scramjet mode and to verify that, at lower altitudes, increased atmospheric pressure, with the corresponding increase in form drag, would not result in a negative net thrust.

Table 2a: Selected mod – net thrust at varying altitudes

Altitude (km)	**design**	**0**	**1**	**5**	**10**	**15**	**20**
Mach #	**Net thrust per unit width (kN/m)**						
4	-17.6	-702.9	-624.6	-372.1	-177.6	-81.8	-37.2
5	18.4	513.3	473.4	327.0	185.1	86.3	39.2
6	38.5	1,232.5	1,116.6	721.8	386.2	182.4	82.9
7	52.6	1,760.3	1,588.1	1,009.6	528.6	248.6	113.0
8	64.0	2,184.7	1,967.6	1,241.4	643.4	301.9	137.2
9	74.2	2,562.3	2,305.4	1,448.3	746.3	349.7	159.0
10	83.8	2,912.9	2,619.5	1,641.8	843.0	394.7	179.4
11	93.0	3,246.1	2,918.4	1,827.0	936.0	438.0	199.1
12	102.1	3,567.0	3,206.8	2,006.7	1,027.2	480.5	218.5
13	111.0	3,875.5	3,485.1	2,182.5	1,117.2	522.6	237.6
14	119.9	4,174.6	3,754.8	2,354.4	1,206.3	564.3	256.6
15	128.6	4,464.2	4,016.5	2,521.3	1,294.5	605.7	275.4
16	137.3	4,734.8	4,264.3	2,684.0	1,381.1	646.7	294.0
17	145.7	4,991.0	4,497.3	2,840.4	1,465.5	686.6	312.1
18	153.9	5,226.1	4,715.9	2,986.0	1,547.9	725.5	329.8
19	161.8	5,433.2	4,906.4	3,123.3	1,626.5	763.4	347.1
20	169.1	5,615.1	5,076.9	3,245.1	1,699.4	798.3	362.9

Table 2b: Selected mod – net thrust at varying altitudes

Altitude (km)	**25**	**30**	**35**	**40**	**45**	**50**
Mach #	**Net thrust per unit width (kN/m)**					
4	-17.2	-8.1	-3.9	-2.0	-1.0	-0.6
5	18.0	8.3	3.8	1.8	0.9	0.4
6	37.6	17.4	8.0	3.9	2.0	1.0
7	51.4	23.9	11.1	5.4	2.7	1.4
8	62.5	29.2	13.6	6.6	3.4	1.8
9	72.5	33.8	15.8	7.7	4.0	2.1
10	81.8	38.2	17.8	8.8	4.5	2.4
11	90.8	42.5	19.8	9.8	5.0	2.6
12	99.7	46.6	21.8	10.7	5.5	2.9
13	108.4	50.7	23.7	11.7	6.0	3.1
14	117.1	54.7	25.6	12.6	6.4	3.4
15	125.6	58.7	27.4	13.5	6.9	3.6
16	134.1	62.6	29.2	14.3	7.3	3.8
17	142.3	66.5	31.0	15.2	7.7	4.1
18	150.3	70.2	32.7	16.0	8.1	4.3
19	158.0	73.7	34.2	16.7	8.5	4.4
20	165.1	77.0	35.7	17.4	8.8	4.6

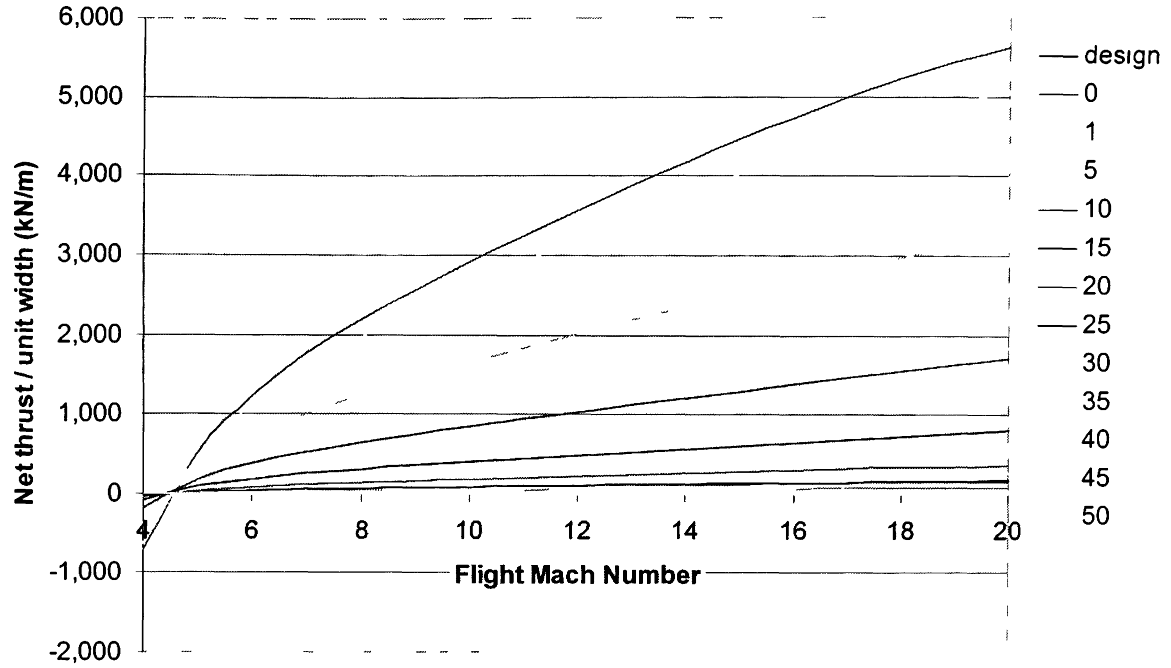


Figure 14: Selected design – net thrust vs flight Mach number at varying altitudes

As can be seen from the above plot and tables, the selected design can be expected to perform very well over a wide range of altitudes and flight Mach numbers.

4. Generation of grid for CFD calculations

Geometry

The essential geometry was developed using the Computer Aided Design program CATIA, by Dassault Systemes. The model was constrained in such a way that the entire geometry could be defined using the same geometric parameters used in the 2-D analysis. For a single design, this means that the inlet ramp configuration can be adjusted for any given Mach number very quickly by adjusting the value of ε_3. An IGS file was made from the CATIA model.

Construction of the grid

The grid for the CFD calculations was developed using the Gridgen software by Pointwise, Inc. Screen shots of the model with and without the surrounding air blocks shown are provided in figures 16 and 17 respectively.

General grid construction information

The following terminology must be understood when using Gridgen. This terminology is pertinent to the discussion that follows:

- Block: A three dimensional enclosed region, defined within the grid file. Blocks are used to define all of the fluid filled regions which are to be modeled.
- Domain: The surfaces, defined within the grid file, which form the boundaries of the blocks. Domains are used both at the solid surfaces of models and at connections between blocks.

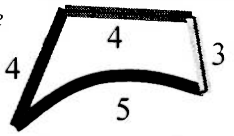
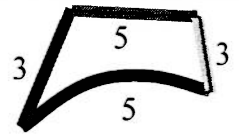
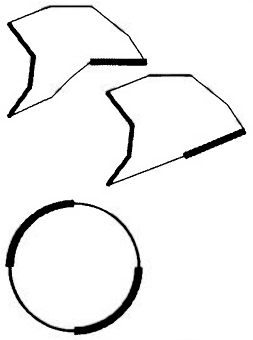
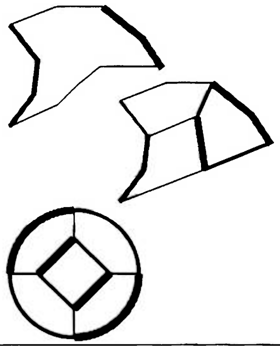
- Connector: A line, defined within the grid file, which defines the perimeter of a domain. The connector is the building block of the grid. A connector can be straight or curved. Even when a connector is curved, the lines which define the actual cells within the block will be straight. The connector is defined by a series of straight segments.
- Database: A line or surface, defined in an IGS or a database file, separate from the actual grid file, which can be used as a constraint when creating the connectors/domains.
- Dimension: This is the number of points used to define the connector. The dimensions of domains and blocks are functions of the dimensions of the connectors that define their boundaries.
- Cells: Each block is comprised of one or more cells, determined by the dimensions of the block.

Domains

Gridgen offers two options for generating domains and blocks, structured and unstructured. Structured domains require four sides, with opposite sides having equal dimensions (resulting in all cells within the blocks having six sides). The CFD code used can utilize only structured blocks and domains.

Each side of a domain is defined by one or more connectors. Any two connecting sides should join at a concave inward angle. The closer two connecting sides come to an angle of 90 degrees at their connection, the lower the chances are of problems developing in the block definition.

Table 3: Guidelines for structured domains

Item	Unacceptable		Acceptable	
a.	*Dimensions of opposite sides unequal*		*Dimensions of opposite sides equal*	
b.	*Connecting sides joined at 180 degrees or concave outward angle. (Blue and bold red lines indicate opposite sides)*[§]		*Connecting sides joined at a concave inward angle. (Blue and bold lines, red and black, indicate pairs of opposite sides.)*	

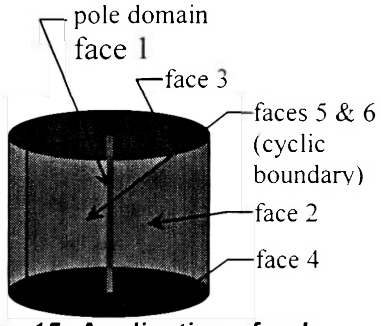


Figure 15: Application of pole domains

One exception can be made to the above guidelines. When creating cylindrical blocks, one of the faces can be a pole domain. This domain will be the axis of the cylinder. Using the pole domain, the outer shell of the block comprises the domain opposite the pole domain. The top and bottom comprise the third and fourth faces. The fifth and sixth faces use the same domain, running from the pole domain to the side, from the cylinder top to the cylinder bottom. Faces 5 and 6 are assigned a special type of boundary condition referred to as a cyclic boundary conditions.

Where possible, high aspect ratio cells should be avoided. (See the geometry subsection of the CFD section.) Additionally, care should be taken to ensure that cells along

[§] Structured grids can, and occasionally are, built similar to the lower of these three samples, however, the "hinge points" at which the sides meet tend to cause problems.

the domain boundary do not have converging sides which cross when extrapolated outward from the domain by the length of the sides. (See the ghost cells subsection of the CFD section.)

Blocks

As with the domains, structured blocks are required by the CFD code. Similar to the structured domains, each structured block is comprised of 3 pairs of opposite sides of equal dimensions.

Initial grid construction

Surrounding Air Mass

The IGS files created in CATIA were imported into Gridgen. The IGS files provide sufficient database entities to grid all of the basic elements of the craft. Connectors were built on selected database entities. Using these connectors, the surfaces of the craft were laid out as domains.

The air mass surrounding the craft was modeled as a rectangular box, running from 3 meters forward of the craft to 9.45 meters aft, 9.5 meters above to 9.4 meters below, and 5 meters to the side. The height of the air mass above and below the craft were selected such that no Mach line from the boundary would impinge on the model.

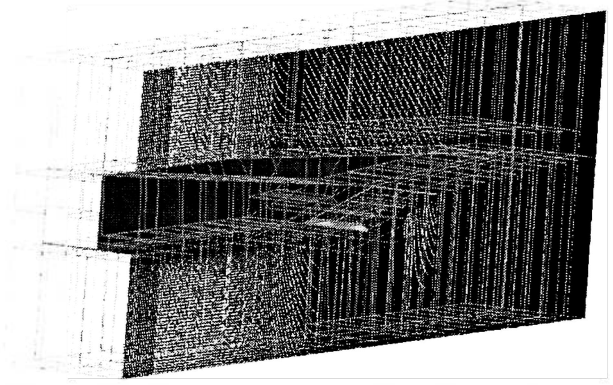


Figure 16: Air mass surrounding the craft

Symmetry

The craft is symmetric across a plane running down the center of the craft along its longitudinal and vertical axes. In order to minimize computational requirements, it was felt that the model could safely be cut in half along this plane, with a slip-wall boundary condition used in way of the plane of symmetry.

Consideration was given to using only a single combustor section, further reducing the computational requirements. The author decided that the flow in way of the sides of the bow ramps, the shock waves resulting from the leading edges of the baffles, and the flow effects off the sides of the nozzle, if a half nozzle were to be modeled, were of such significance that any reasonably accurate calculation should deal with these effects.

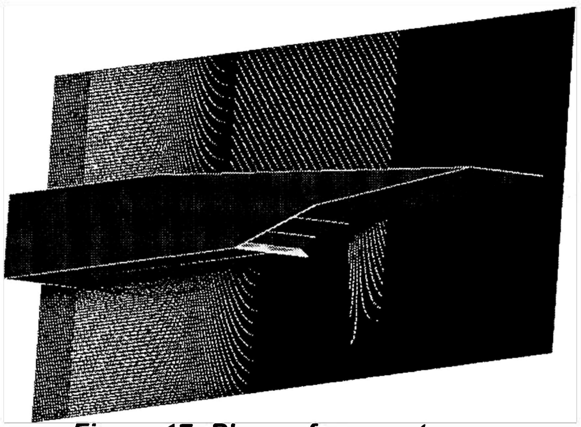


Figure 17: Plane of symmetry

Boundary Layer

When taking into account viscous effects at the walls, the boundary layer must be modeled. It was decided that the grid should be developed such that no less than 10 cells were within the boundary layer throughout the flow-path.

An approximate formula for boundary layer thickness was developed from a plot of Mach number versus boundary layer thickness in Schlichting[10].

$$\delta = \left(0.0287M^3 + 0.2079M^2 + 0.4229M + 3.1103\right) \times \sqrt{\nu L / V} \tag{54}$$

Where:

δ = boundary layer thickness (m)

M = Mach number

ν = kinematic viscosity (m^2/s)

L = Distance from surface leading edge to location at which   is calculated (m)

V = Flow velocity (m/s)

Using this formula, boundary layer thickness is calculated in a separate sheet within the 2-D analysis spreadsheet, at selected points throughout the flow-path. Acknowledging that the boundary layer decreases towards zero as the length of flow along a surfaces decreases to zero, boundary layer thickness was calculated 1 cm aft of the leading edge of each pertinent plane and at the end of each plane. The boundary layer near the leading edge was generally found to be approximately 0.03 mm or greater, while the boundary layer thicknesses at the ends can be up to 15 mm thick. In order to assure that no less than 10 cells were used through the majority of the boundary layers, and considering that at the termination of any given plane, two different boundary layer thicknesses needed to be considered, it was decided to assume a uniform boundary layer thickness of 0.03 mm for the purposes of grid generation. Each connector running from the model to the perimeter of the air mass was split at 0.03 mm from the model. Each of these 0.03 mm connectors was given a dimension of 11. Connectors within the combustor and nozzle were similarly split and dimensioned.

Fuel injection

Rather than attempting to design special injectors and then build them into the model, a perforated injector sheet is assumed. This sheet operates as a perforated plate, set flush in the hull plating, through which the fuel is forced at high pressure, entering the air stream in an even sheet across the flow, already vaporized.

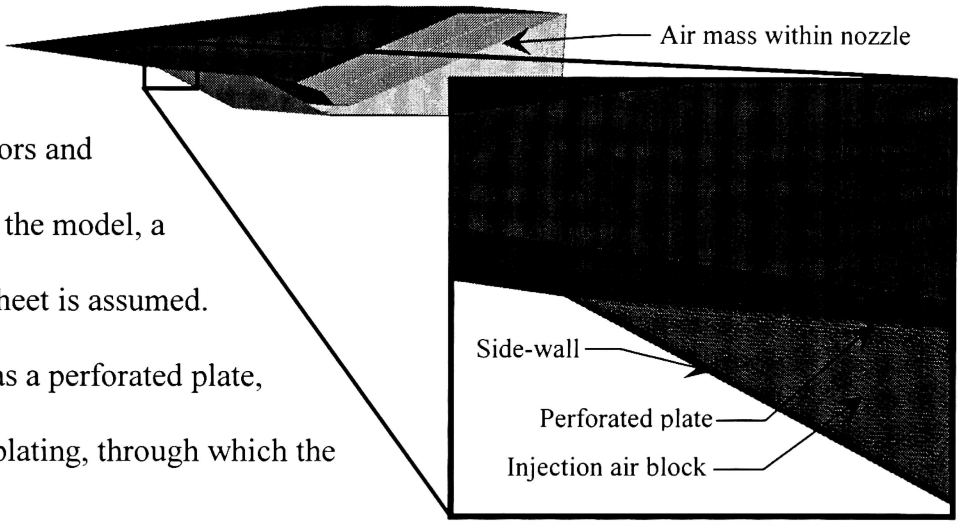


Figure 18: Fuel injection plate

Using this assumption, a block was built just aft of the leading edge of the sides of the combustion chamber walls. This places the fuel inlet upstream of the forward ramp. This location was chosen for three reasons:

1. Placing the fuel inlet forward of the combustor inlet shocks facilitates mixing, leading into the combustor while using the shocks to supplement dissipation.
2. In the 2-D analysis, the inlet shocks were found to produce insufficient temperatures to result in immediate combustion, hence the majority of the fuel was expected to reach the combustion chamber still unburned, but well mixed with the incoming air.
3. Having the block in this location reduced grid density problems for the two blocks leading into the combustors.

Grid density

The detail of calculation is controlled by the density of the grid. The smaller the cells, the more detailed the calculation. High grid density (many small cells) also corresponds to higher calculation time. In order to maximize the quality of the calculation, a high grid density is used where fine detail of the calculation is critical, most particularly in way of the combustion chamber and, as previously noted, in way of the boundary layer.

Table 4: Vertical grid densities

Component	**Vertical: Length (m)**	**Vertical: Dimensions**
Combustor	0.5	51
Nozzle exıt	5.991	51
Lower block (aır) ıwo ınlet face	13	56
Lower block (aır) ıwo combustor	9 4	61
Upper block (aır) ıwo ınlet face	12	56
Upper block (aır) ıwo model upper surface	9.5	61

In the axial direction, the combustors again had the highest grid density, at 100 cells/m. Forward and aft of the combustors, grid density was decreased as smoothly as possible.

Changes in grid densities were smoothed, as much as possible, by adjusting the grid point distribution of selected connectors.

Load balancing

The CFD calculation was to be carried out on a very large cluster, where each block would be calculated by a separate processor. Operating in this manner, if a single block has significantly more cells than the others, all of the other processors wait on the one running the calculation for the large block. This is a highly inefficient use of the limited processor time available on the large supercomputing clusters. To avoid this delay, it was necessary to subdivide the blocks initially produced so that all blocks have roughly the same number of cells and, more importantly, no block has excessively more cells than the rest.

After the fluid filled space around the craft was fully defined by blocks, all of the blocks defining this region were cataloged in an Excel spreadsheet. The spreadsheet was set up so that the user could indicate the number of sub-blocks that any given block would be broken down to. The spreadsheet was automated to list all of the subdivided blocks, as they would be listed when created in Gridgen. A set of calculations was set up to help the user to determine exactly how to divide each block. As the main blocks were broken down, this block tracking spreadsheet was kept up to date. Appendix 2 is the final listing of the actual blocks used in the grid.

As modifications were made to the CFD code, it became necessary to adjust the grid. Some blocks were rebuilt while others were further split. The cells within some blocks needed to be subdivided beyond what could be carried out in Gridgen. In order to maintain balanced loading, the above spreadsheet needed to be used many times after the blocks were originally subdivided. In order to facilitate filling the spreadsheet, a small FORTRAN program, blocksummary, was created to extract the total number of blocks from the exported grid file (grid.grd); the dimensions of each of the blocks from the per block grid files were then extracted using a separate utility packed with the CFD code. Using this data, blocksummary writes out a text file (blocksummary.txt) listing the dimensions for each block. This information can, with a little data manipulation, be copied into the above Excel spreadsheet, saving a great deal of time versus copying the dimension to the sheet manually.

Modifications to the grid

As noted previously, while learning and modifying the CFD code, it was found necessary to make modifications to the grid. In general, modifications made to the grid at this point were carried out for one of two reasons:

1. To prevent generation of negative volumes when extrapolating ghost cells. (Ghost cells and negative volumes, as well as means of adjusting the grid to prevent negative volume, are discussed in the ghost cells subsection of the CFD section.)
2. To reduce the aspect ratio of a cell.

Two methods were employed to reduce aspect ratios of cells:

- Redefinition of the connectors defining the boundary layer.
- Subdivision of blocks, using a separate utility, external to the grid generation program. (This method is discussed in detail in the "Grid_subdiv" subsection of the CFD section.)

The connectors defining the ten cell depth of the boundary layer below the craft nozzle were doubled in thickness and the grid point distribution along these cells was set to equal spacing. Subsequent to this correction, geometry and calculation errors in these cells did not develop.

The connectors defining the boundary layer at the bottom of the insides of the nozzles were increased in depth, ranging from 0.03 mm at the combustor inlet to 0.07 mm at the combustor exit and scaling upward in the nozzle to 0.13 mm at the nozzle exit. The connectors forward of the inlet and at the tops of the combustors and nozzles were resized as follows:

1. The boundary layer connectors in the inlet region contained between the plane of symmetry and the forward swept baffle supporting the cowl at the outside of the craft were resized from 0.08 mm at the forward connectors to 0.30 mm at the combustor inlet.
2. Inside the combustors, the boundary layer connectors range from 0.30 mm at the inlet to 1.0 mm at the exit.
3. Inside the nozzles, the boundary layer connectors range from 1.0 mm at the inlet to 5.0 mm at the exit.

Modification of the boundary layer connectors in this manner brings the boundary layer cells assumed in the model closer to the boundary layer thickness profile assumed from Schlichting[10]. The boundary layer thickness at the nozzle exit, even at the top, is still considerably less than that predicted, however, this was a compromise between reducing aspect ratio and minimizing time consuming changes which further increases in cell size at the nozzle exit would necessitate throughout the remainder of the grid.

5. Computational Fluid Dynamics code

Brief explanation of the code

As noted in the introduction, CFD analysis was carried out using the open source FORTRAN 90 CFD code *hyp*, developed by Dr. Eric Perrell. *Hyp* takes a series of files providing the grid and boundary conditions for one or more associated fixed volume blocks of gas along with a file giving the free stream conditions. On a cell by cell basis, *hyp* calculates the fluxes entering and exiting each face of the cell, at the same time determining the flow conditions (temperature, pressure, and composition of the gasses) within the cell. These calculations are carried out on successive cells using the fluxes and conditions from the bounding cells. Calculation of the fluxes is carried out by Steger-Warming flux vector splitting. Combustion calculations are carried out using Arrhenius reaction rate coefficients, treating each cell as a constant volume fixed mass reactor.

Within *hyp*, different methodologies can be used for the analysis, depending upon values entered in the main input file. The variables controlling the methodology used by *hyp* are:

- <u>mode</u>: *This variable defines what type of analysis is to be run. Explanations of the different modes are provided below.*
- <u>implicit</u>: *Indicates that the fluxes and sources are evaluated at the following time steps, according to a first order Taylor series expansion in time. The alternatives are semi-implicit and explicit.*
- Semi-implicit: *Indicates that only the sources (chemical reactions) are carried out at the following time step.*

- explicit: *(This is not a variable, but is the default method if both implicit and semi-implicit are set to .false..) Indicates that the fluxes and sources are evaluated at the current time step.*
- viscous: *Specifies if the calculations must take into account viscosity or not.*

The following modes are available:

1. equilibrium air
2. premixed thermochemical equilibrium
3. non-premixed thermochemical equilibrium
4. chemical non-equilibrium, vibrational/rotational equilibrium
5. chemical/vibrational non-equilibrium, rotational equilibrium
6. chemical/vibrational/rotational non-equilibrium

Modes 2 and 3 are general thermochemical equilibrium modes, in which the elemental composition is specified at the inflow(s). Modes 4, 5, and 6 are general chemical non-equilibrium modes, in which the molecular composition is specified at the inflow(s).

For this project, the model was run in mode 5. The implicit option was set as false. Due to time constraints in modifying the code, viscosity and reactions were turned off. This, naturally, invalidates the need for building a boundary layer into the grid, however, the decision not to turn on viscosity came well after the grid had been built. This does, however, leave the grid in good shape for a follow-up project in which viscosity is accounted for.

The main program *hyp*, itself, does very little of the actual work in the code. Instead, *hyp* calls subroutines to carry out the complex calculations. The author was involved in modifications to only three of the subroutines called in *hyp*: *exchange_ghostcells*, *exchange*, and *geometry*. Explanations of how these subroutines operate and the changes implemented by the author will be presented in some of the following sections. Before these modifications can be discussed, two of the concepts used in *hyp* must be mentioned: MPI and ghost cells.

Concepts

MPI

When the code is operated on a multi-processor system, *hyp* uses the portable Message Passing Interface (MPI) library[11] to send the files for each block to a separate processor and have each of these processors run the calculation for only their assigned block, while transferring data at the boundaries of each block when necessary.

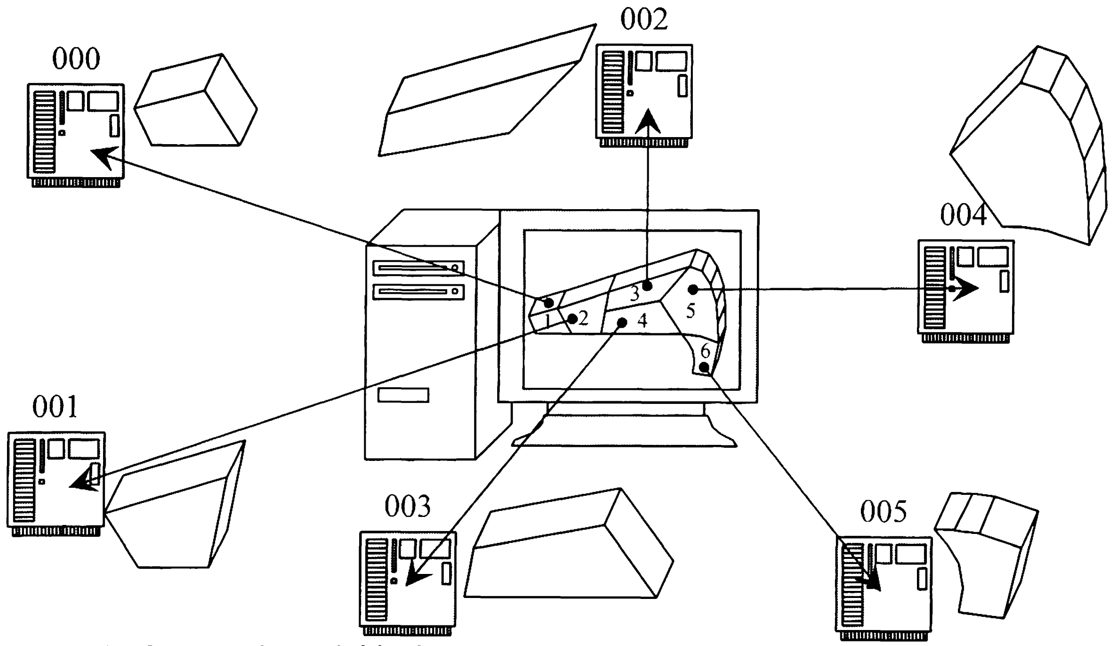


Figure 19: MPI sends each block to a separate processor

As MPI is used in *hyp*, for each side of each block, each face on that side which communicates (shares a common communication boundary) with another block transfers data to and receives data from the corresponding face of the communication partner (the block which the face talks to) by separate send and receive commands.

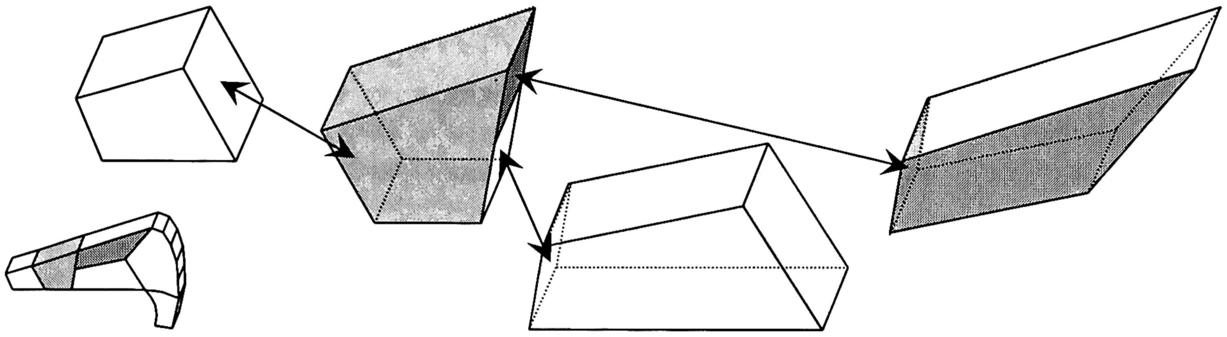


Figure 20: Communications operations to/from a single block

Five basic MPI commands are used in the code: *mpi_irecv, mpi_isend, mpi_waitall, mpi_allreduce, and mpi_finalize*:

- MPI_IRECV is used to send out an empty array to be filled.
- MPI_ISEND is used to send a filled array to be used to fill in the arrays sent with mpi_irecv.
- MPI_WAITALL instructs the current process to hold until all of the receive buffers (arrays) sent by the process have been filled.
- MPI_ALLREDUCE performs a basic MPI operation using a specific piece of data sent by all processes and returns the result to all processes. The process used in connection with mpi_allreduce in *hyp* is mpi_min. This function searches for the minimum value of all values sent with mpi_allreduce.
- MPI_FINALIZE releases memory allocated by the MPI functions. This command must be called after all other MPI operations are complete.

The above five commands have the following syntax respectively:

- call mpi_irecv(recv_buffer(start_position:end_position), points, MPI_REAL, sideCP(face), identifier, mpi_comm_world, req(count), ierr)

- call mpi_isend(send_buffer(start_position:end_position), points, MPI_REAL, sideCP(face), identifier, mpi_comm_world, req(count), ierr)
- call mpi_waitall(count,req(1:count), status, ierr)
- call mpi_allreduce (operand, result, count, mpi_real, operator, mpi_comm_world, ierr)
- call MPI_Finalize (ierr)

The variables used in the above commands are summarized below:

- "recv_buffer" and "send_buffer" are real one dimensional arrays sized to allow all necessary data to be transferred to or from the process. These arrays are sent with the send and receive buffers.
- "start_position" and "end_position" are integer values which indicate the positions within the send and receive buffers pertinent to the requested operation.
- "points" is an integer value specifying the number of data points to be transferred in the requested operation.
- "MPI_REAL" specifies that the type of data being transferred is mpi_real.
- "sideCP(face)" specifies which process the data is either being requested from (mpi_irecv) or being sent to (mi_isend). This parameter will appear, in the code, as one of the following: licp(face), ljcp(face), lkcp(face), uicp(face), ujcp(face), ukcp(face). This means that a separate request is sent for each face on each side of the block.
- "identifier" is an integer value that can be used to clarify the specific face to/from which data is being transmitted/received. Where no two blocks share more than one common face, the value of identifier can be set as a constant value for all operations. Where

multiple faces on a single side communicate between two blocks, identifier is required to clarify which face is talking with which.

- "mpi_comm_world" specifies a communication group that includes all of the processes that were running when the program starts.
- "req(count)" specifies the operation request number. "count" is an integer value which is incremented by 1 every time an irecv or isend operation is carried out.
    - Note that for mpi_allreduce, "count" specifies the number of memory locations in the datatype. For mpi_allreduce, as it is used in *hyp*, count is always 1.
- "ierr" stores any error codes generated in the MPI operation.
- "operand" specifies the variables to be worked with using mpi_allreduce. In *hyp*, this operand will be the individual process' minimum time step.
- "result" gives the variable to which the result of the MPI operation is saved in the process.
- "operator" specifies the MPI operation that mpi_allreduce applies to all of the data exchanged in the mpi_allreduce call. As used in *hyp*, operator is mpi_min, which finds the minimum value.

Ghost cells

In order to calculate the fluxes in the boundary cells of a block, the conditions in the cells adjacent to that block's boundary cells, in the adjacent blocks, must be known.

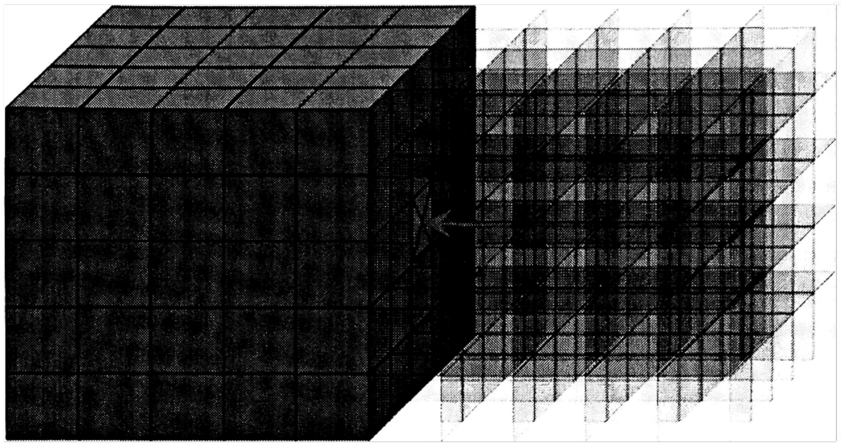


Figure 21: Flow conditions in adjoining cells of adjacent blocks must be known

For communication boundaries (boundaries between two adjacent blocks through which flow may pass), this means that the values in the Q (flow conditions) array must be passed to the block in question. In order to do so, the first step is to provide a set of new locations to store these flow conditions. The new locations take the form of "ghost" cells.

For non-communication boundaries, ghost cells provide a location to store flow conditions at the boundaries. This can be used to implement the boundary conditions. As an example, for a no-slip solid wall boundary, the flow rates in the ghost cells would be set equal to and opposite the flow rates on the corresponding boundary cells, so that the average values at the surface are equal to zero.

The subroutine *grid.f90* extrapolates ghost cells from the existing boundary cells by copying the first row of connectors running normal to the faces, adding the dx, dy, and dz to the existing boundary coordinates to make a new row of coordinates at x = 0, x = xmax +1, y = 0, y = ymax +1, z = 0, and z = zmax +1.

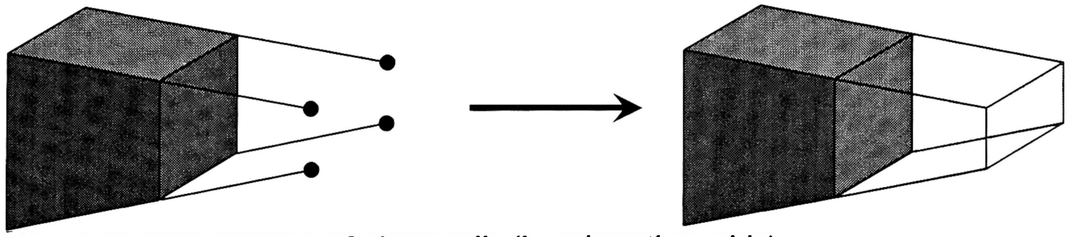


Figure 22: Extrapolation of ghost cells (in subroutine grids)

Where any two cells meet along a common face, and flow is permitted between the two blocks, the common face is termed a "communication boundary". In the input files, the face will have a boundary condition equal to zero. When a face is found to be a communication boundary, the ghost cells extrapolated in *grid* are replaced with the actual adjoining row of cells in the communication partner. This replacement takes place in subroutine *exchange_ghostcells.f90.*

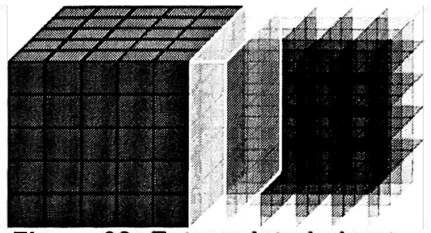


Figure 23: Extrapolated ghost cells replaced w/ adjoining row of cells from CP

When ghost cells are extrapolated in *grid*, if the boundary cell has a high enough aspect ratio, and the sides are not parallel, the extrapolated grid lines can intersect, resulting in zero or negative volumes in the ghost cells.

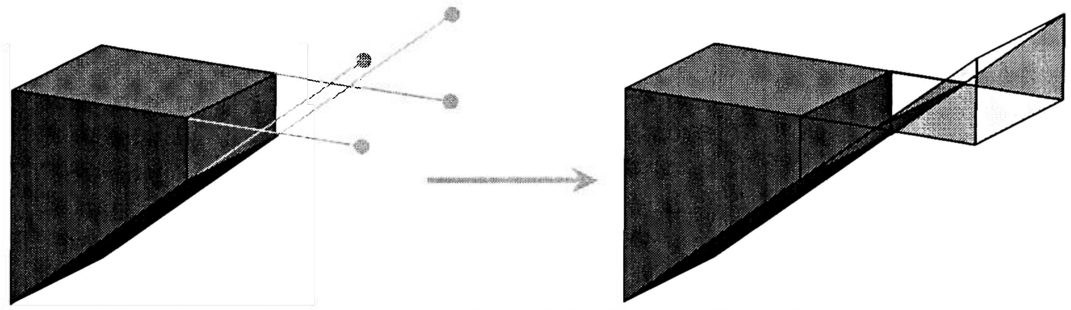


Figure 24: Extrapolation of ghost cell resulting in negative volume

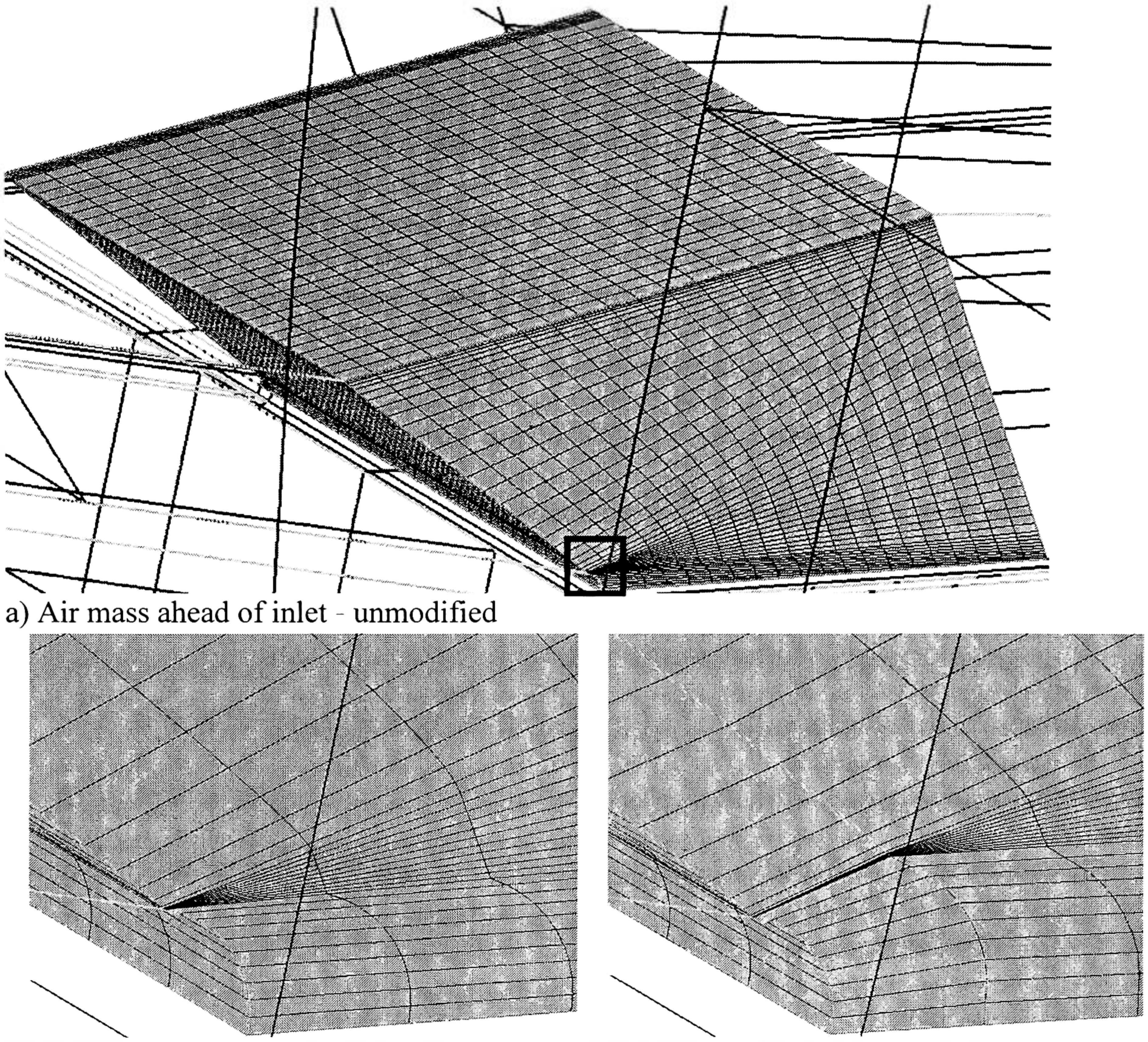
There are four ways to deal with this problem:

1. Increase the subdivision of the block.
2. Reduce the grid spacing close to the offending boundary reducing the aspect ratio of the boundary cells.
3. Adjust the boundaries of the block to make the sides of the boundary cells more parallel.
4. Further subdivide the boundary cells of the block using an external utility.

In many cases, it is feasible to simply increase the number of cells that comprise the block, reducing the aspect ratio of any given cell so that the grid lines do not cross when a ghost cell is extrapolated. In some complex geometries, this may not be possible. Where the offending block is bounded on several sides by other blocks, increasing the number of grid points in one direction on the offending block can require a similar increase in the dimensions of adjoining blocks. This may prove to be undesirable from the standpoint of load balancing, as it will increase the dimensions of one or more blocks.

If the grid is generated in Gridgen, it is often possible to adjust the grid point distribution of any given connector by entering the beginning spacing, the ending spacing, and the distribution function. While this method is extremely useful in specifying the exact shape of the cells within a block, it should be used with care. For some complex geometries, particularly when grid point distribution has already been altered for other connectors, further alterations to avoid generating zero volumes in the ghost cells may result in highly skewed or even negative volume cells inside the block or other adjacent blocks.

Where the boundaries of the block are not necessarily fixed, the boundaries may be adjusted to bring the sides closer to being parallel, thus ensuring that the lines of the cells within the blocks do not converge when the ghost cells are extrapolated. This may be feasible if one or more of the faces adjoining the face at which ghost cells converge are communication boundaries.



a) Air mass ahead of inlet - unmodified

b) Grid lines converge at solid surface

c) Grid lines adjusted to prevent convergence in ghost cells

Figure 25: Repairing ghost cells by altering the boundaries of the blocks

Where the geometry cannot be altered as mentioned above, an external utility, *grid_subdiv*, can be used to further subdivide any specified block in one or more of the dimensions i, j, and k.

Modified and new code

Utility (program) grid_subdiv – new code

If MPI is used, the external utility *grid_subdiv* can be used to subdivide cells within a block, outside of the grid generation program itself. This subdivision is carried out after the grid has been generated and exported and after the utilities *translator* and *extractor* are run to break the master grid file *grid.grd* and the master boundary condition file *BC.inp* into separate files *xxx/hyp.xxx.g* and *xxx/hyp.xxx.inp* for each process. These new files will contain only the grid coordinates and boundary conditions for the specific process.** If *grid_subdiv* is to be used, it must be run before the main program, *hyp*.

Grid_subdiv reads from an input file *grid_subdiv.inp* which contains a list of blocks to be divided as well as the instructions for how to divide the blocks. The program loops through the instructions, reading the instructions for each block. *Grid_subdiv* opens the grid file and reads the coordinates. The data in the existing grid file is backed up to a new file identified *as xxx/hyp.xxx.g.old.* New coordinates and maximum dimensions are calculated and the new grid file is written out with the original grid file name (*xxx/hyp.xxx.g*). Because the boundary condition file contains the i/j/k coordinates defining the position of each face on the block, and some of those coordinates will change when new points are added ahead of them, the boundary condition file is also read. A backup of the boundary condition file is

** (Note that the notation *xxx* used above, and frequently throughout the rest of this document, is intended to stand in for the three digit number assigned to an individual process. For example, the files for process 3 would be 003/hyp.003.g and 003/hyp.003.inp.)

written, new coordinates are calculated, and the boundary condition file is rewritten with the amended coordinates.

Grid_subdiv is not designed to introduce discontinuities within a block; hence, when instructed to subdivide a block over a range of i values, for example, the same subdivision will apply for all cells within this range of i coordinates, for the entire range j = 1 to jmax and k = 1 to kmax. Additionally, as *grid_subdiv.f90* and the subroutines *exchange _ghostcells_m1.f90* and *exchange_interior_m1.f90* are written, it is not possible to specify more than one subdivision region along any given axis for subdivision. Hence, it is not possible to specify that only the first and last rows of cells in the k direction be divided, without dividing the cells in between. This is a compromise between complexity of the instructions and code versus flexibility of the utility.

The grid subdivision input file

The input file *grid_subdiv.inp* lists the process number (block number minus 1, using a 3 digit format), the degree of division in the i, j, and k directions, and the start and ending cell numbers for division. The degree of division in the i, j, and k directions is given as number of cells after division per pre-division cell. The format is as follows:

proc_id, div_i, div_j, div_k, start_i, end_i, start_j, end_j, start_k, end_k

Example input:

```
000, 1, 2, 3,  0,  0, 2, 4, 1,  1
271, 3, 1, 4, 25, 25, 0, 0, 2, 13
```

Translation:

Two blocks will be divided. In Gridgen, these blocks would be identified as block 1 and 272. In *hyp*, *proc_id = block# - 1*, hence processes 0 and 271 are modified.

Block 1:

- In the i direction, do not divide cells.
    - Since no division takes place in the i direction, the values entered for start_i and end_i are significant only in that values must be supplied. To avoid confusion, it is recommended that zeros be given when no cell division is to take place.
- In the j direction, divide cells in half.
    - Division starts with the 2nd row of cells in the j direction and ends with the 4th row. Since this is a cell number rather than j coordinate, the starting j coordinate is 2 and the ending j coordinate is 5.
- In the k direction, divide cells into thirds.
    - Only the first row of cells in the k direction is divided.

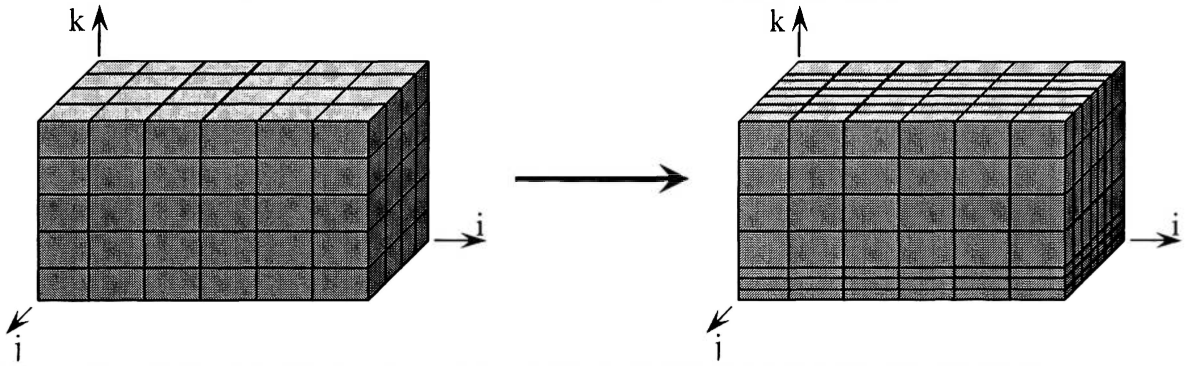


Figure 26: Grid subdivided using grid_subdiv (subdivision per example, block 1)

The existing *grid_subdiv.inp* file contains 13 lines of instructions at the top of the file which are intended purely to help the user understand precisely how to fill out the subdivision information. If no existing *grid_subdiv.inp* file is available, one can be created. Simply insure that block division instructions start on the 14th line of the file.

Brief look at the grid file

Because the program centers around modifying the grid file, it is worth noting the format that the grid files are written in. The grid files are written as Plot3D unformatted files. As the files are unformatted, the user cannot open the file and read it, but the data is stored as follows:

nblocks

imax(1), jmax(1), kmax(1), imax(2), jmax(2), kmax(2),..., imax(nblocks),

jmax(nblocks), kmax(nblocks)

x (i=1:imax, j=1:jmax, k=1:kmax)

y (i=1:imax, j=1:jmax, k=1:kmax)

z (i=1:imax, j=1:jmax, k=1:kmax)

In this notation:

- nblocks: number of blocks in the file (for the per process grid files *xxx/hyp.xxx.g* files, this will always be 1)
- imax: total number of grid points in the i direction (# cells in the i direction + 1)
- jmax: total number of grid points in the j direction (# cells in the j direction + 1)
- kmax: total number of grid points in the k direction (# cells in the k direction + 1)
- x: an array containing the x coordinate for each grid point
- y: an array containing the y coordinate for each grid point
- z: an array containing the z coordinate for each grid point

Calculations

Where the subdivision instructions call for new points to be created in the grid, the x, y, and z coordinates of the new points are calculated by interpolation. For division in the i direction (used as an example), calculation is as follows:

$$x_{new}(i_{new}, j, k) = x(i, j, k) + \{x(i+1, j, k) - x(i, j, k)\} \times \left\{\frac{i_{new} - i}{div\_i}\right\}$$

$$y_{new}(i_{new}, j, k) = y(i, j, k) + \{y(i+1, j, k) - y(i, j, k)\} \times \left\{\frac{i_{new} - i}{div\_i}\right\}$$

$$z_{new}(i_{new}, j, k) = z(i, j, k) + \{z(i+1, j, k) - z(i, j, k)\} \times \left\{\frac{i_{new} - i}{div\_i}\right\}$$

Since *grid_subdiv* increases the number of cells inside a block, the maximum dimensions along each axis are changed. For example:

$$i\max = i\max + (div\_i - 1) \times (end\_i + 1 - start\_i)$$

As noted earlier, some, potentially most, of the coordinates in the boundary condition file which define the bounds of the faces for the block cease to be accurate when subdivision is applied. These coordinates must be recalculated. How these values change is dependent upon whether they occur before, inside, or after the subdivision region.

- If the coordinate occurs before the subdivision region, i.e., the value is less than the start coordinate for division along the pertinent axis, no change is made.
- If the coordinate occurs inside the subdivision region, the value must be increased by the number of grid points added between the start of subdivision and the location of the coordinate:

  $$coord = coord + (coord - start) \times (div - 1)$$

- If the coordinate occurs after the subdivision region, i.e., the value of the coordinate is greater than the end coordinate for division along the axis, the change is identical to that of the max dimension:

  $$coord = coord + (div - 1) \times (end + 1 - start)$$

Some basic if-then statements are included in the code to insure that the coordinates in *xxx/hyp.xxx.inp* are positive values while this calculation is carried out, and are reverted to their prior sign upon completion.

Summary of the procedure

Since the boundary condition files must be rewritten, the quickest means of writing the files uses the same namelists that *hyp* uses to read the boundary condition files. The two namelists, *run* and *boundary_conditions*, were copied directly from *hyp*. A number of variables included in these namelist, but not otherwise required in *grid_subdiv* were also copied in.

After declaring variables and namelists, the subdivision input file is opened. The first thirteen lines, being instructions to the user, are skipped over. After this, the code executes a Do loop to read through the instructions for each block, acting on the instructions for that block before reading the next instruction. This loop is exited either when an error is detected in the instructions or when the end of file is read.

Inside the loop, after error checking of the instructions is completed, the input file *xxx/hyp.xxx.inp* is opened. The namelists *run* and *boundary_conditions* are read. This file is closed and a backup copy is written out to *xxx/hyp.xxx.inp.old.*

The grid file *xxx/hyp.xxx.g* is opened. The maximum i, j, and k dimensions (imax, jmax, and kmax) are read. The x, y, and z arrays are allocated with dimensions (imax, jmax, kmax). The x, y, and z coordinates are read from the file into the corresponding arrays and the file is closed. A backup copy is written to *xxx/hyp.xxx.g.old.*

New maximum dimension newimax, newjmax, and newkmax are calculated based upon the block division instructions. Arrays xnew, ynew, and znew are allocated with dimensions (newimax, newjmax, newkmax). These arrays are set equal to their original coordinate counterparts: xnew = x, ynew = y, znew =z. This leaves the new arrays only partially filled.

Rather than attempting to interpolate all of the new points in a single pass through the grid, interpolation of the new grid points is carried out first for division in the i direction, then for division in the j direction, and finally for division in the k direction.

For division in the i direction, filling of the arrays xnew, ynew, and znew takes place in three stages. In each stage, coordinates are looped through via three nested loops. In each stage, the j and k loops run through their full ranges while the i loop covers only a portion of its range in each stage. In the first stage, the i loop runs though the region before subdivision. In the second stage, the i loop runs through the subdivision region. In the final stage, the i loop runs from the end coordinate of the subdivision region to the final i coordinate.

In the first and third stages, coordinates are copied directly to the xnew, ynew, and znew arrays, though the i indices in the third stage need to be adjusted to reflect the number of points added inside the subdivision region. The second stage requires that for each cell all required subdivision coordinates be interpolated and added to the xnew, ynew, and znew arrays, adding in, also, the cell end coordinates. The i indices again need to properly account for the number of grid points added in ahead of the point being stored to the arrays.

The same basic procedure is used for the other two axes. For these two axes, the first stage is not repeated as the coordinates will not change from those already filled in. Before subdivision of the j axis, the x, y, and z arrays are deallocated and reallocated to the new maximum dimension. These arrays are set equal to their new value counterparts. The x, y, and z arrays are again set equal to their counterparts before subdividing in the k direction.

Upon completion of the above process, the xnew, ynew, and znew arrays should be completely filled. The grid file *xxx/hyp.xxx.g* can be rewritten with the new maximum dimensions and new coordinates.

For each side of the block, a loop is carried out to examine each face and recalculate the face defining coordinates. The input file *xxx/hyp.xxx.inp* is rewritten using the namelists *run* and *boundary_conditions*.

This completes the procedure used for each block. The do loop ends here. The input file is closed after the end of the loop.

Subroutine exchange_ghostcells – modified

The subroutine *exchange_ghostcells* is required when MPI is used. Where any two blocks communicate, *exchange_ghostcells* is used to replace the ghost cell coordinates calculated in subroutine *grid* with the coordinates of the first set of grid points behind the common face of the adjoining block. See the "Ghost cells" section for an explanation of ghost cells.

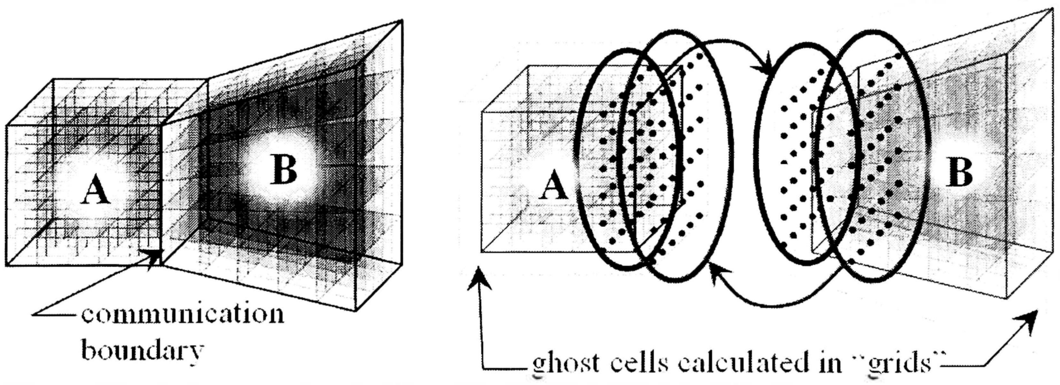


Figure 27: Exchange_ghostcells – transferring the coordinates

Unmodified subroutine

The unmodified *exchange_ghostcells* operates according to the following basic procedure:

- Total buffer size is determined based upon the dimensions of the block. Based upon this dimension, the receive buffer is allocated.

- The empty receive buffers are posted.
- The send buffer is allocated using the same buffer size determined for the receive buffer.
- The coordinates of the first set of cells behind the face are packed into the send buffer and the buffer is sent.
- MPI_WAITALL is called, ensuring that no process continues through the code until all of their receive buffers are filled.
- The ghost cell coordinates are filled from the receive buffers.

Posting the empty receive buffers is a fairly simple process which is repeated for each of the six sides of the block. For each side of the block, the code loops through all of the faces. Inside this loop, looking at one face at a time, a check is made to determine if the face is a communication boundary. (The boundary condition will have a value of zero.) If the face is a communication boundary, the following are carried out:

- The number of points to be sent is calculated based upon the dimensions of the face.
- An identifier for the face is assigned.
- The empty receive buffer is sent.

Packing the send buffer (sending the coordinates for the first row of grid points behind the face) is a slightly more complex procedure. For each side of the block, the code loops through all of the faces. Inside this loop, looking at one face at a time, a check is made to determine if the face is a communication boundary. If the face is a communication boundary, coordinates need to be sent.

The order in which coordinates for communication boundaries are saved to the original grid file varies based upon the orientation of the block. Because the receive buffer is one dimensional, the same order that was used in writing the grid file must be used when saving ghost cell coordinates to the send buffer and pulling them from the receive buffer. The x, y, and z coordinates are sent, and can be retrieved, using a pair of nested loops. The order can be determined from the values of the i/j/k coordinates that define the bounds of the face. One of the two pairs of high/low coordinates will be positive, while the other will be negative. The outer loop must step through the axis whose bounds are defined by the high/low coordinates with the negative values.

Once it is determined that coordinates need to be sent to the buffer for a given face, the above convention is used to determine the order in which coordinates need to be sent. At the same time, the negative i/j/k coordinates must be converted to positive values for the remainder of the send operation.

The i/j/k coordinates defining the bounds of the face are not constrained to increase in value. It is possible, for example, for the value of ljli(face) to be either greater or less than the value of ljui(face). To accommodate this, a value of -1 or +1 is assigned to the increment variable for each of the two axes that can vary across the face.

As with posting the receives, the number of points to be sent must be calculated based upon the face dimensions and a face identifier must be assigned. Variables used to track the current position within the buffer are initialized and the nested loops, discussed above, are used to save the x, y, and z coordinates for the first row of grid points behind the face to the send buffer.

After the nested loops, the send buffer is sent using the command MPI_ISEND (see the MPI section for more details).

Unpacking the receive buffers (assigning the coordinates stored in the buffer to the corresponding ghost cell coordinates) follows a process very similar to that used to pack the send buffer. Again, for each side of the block, the code loops through all of the faces. Coordinates need to be pulled from the receive buffer only if the face is a communication boundary.

As with packing the send buffer, the ordering of the nested loops must be determined. The i/j/k coordinates defining the bounds of the face are converted to positive values, and increments for each axis are defined. The number of points to be retrieved is calculated. The values in the receive buffer are copied to the x, y, and z arrays using the pair of nested loops. Because the buffer is one dimensional, the x, y, and z coordinates are saved sequentially, hence, after copying each of the x, y, and z coordinates, the position within the buffer must be updated.

Modified subroutine

It was discovered that the original code was unable to deal with a specific block topology which was used in the model. Modifications were required to accommodate this condition. Additionally, the use of grid subdivision necessitated some fairly significant changes to *exchange_ghostcells*.

Topology issue

A problem was found in transferring ghost cells when multiple faces on a single side of one block communicate with adjoining faces on 2 or more adjoining sides of a single communication partner.

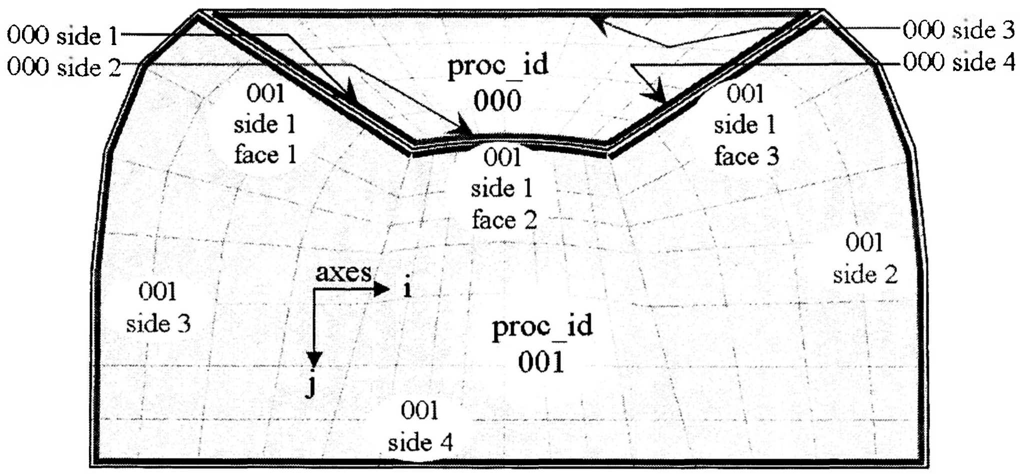


Figure 28: Exchange_ghostcells – topology issue

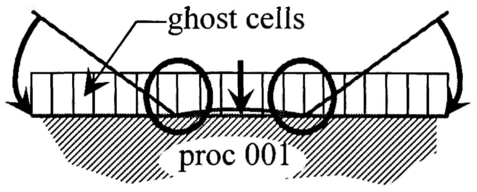


Figure 29: Exchange_ghostcells – topology issue: side straightened

To help visualize the nature of the problem, imagine that side 1 of proc 001 (fig. 28) is flattened out into to a straight line (fig. 29). Flattened out, it is clear that there must be one ghost cell on either side of the connection between each face (fig. 30a).

When transferring the grid points from the existing cells in proc 000 to the lower j ghost cells in proc 001, only two coordinates exist to define the three end coordinates needed for the outer bound of the two ghost cells at the junction between faces (fig. 30b). Because the coordinate axes switch between faces, the points being copied also switch. As a result, when copying the coordinates from the receive buffer to the x, y, and z arrays, the last coordinate recorded for each of the affected sides (except the final side) are over written. This can result in cells with only 5 sides or even result in negative volumes depending upon the geometries of both blocks (fig. 30c).

To rectify this geometry issue, the following two tasks must be carried out:

- Identify the specific grid point(s) for which the problem exists.
- Replace the coordinates for these points with coordinates calculated from the grid point before and after the affected point, which insure that the two cells in way of the junction between faces are valid cells.

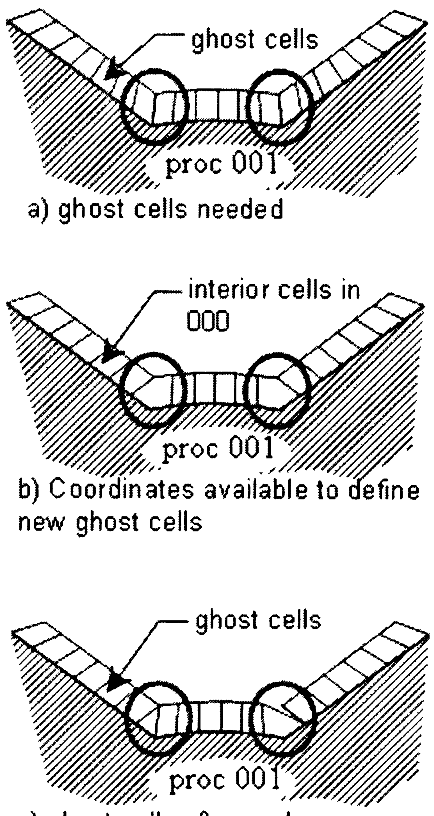


Figure 30: Exchange_ghostcells topology issue: affected cells

Identifying affected cells

The above problem will occur whenever two adjoining faces on one side of the block come from two different sides of the same communication partner. This fact is used to identify the affected grid points. Initially it was assumed that it would be necessary to capture both the first and second versions of the overwritten coordinate. This made it necessary to build the detection of affected cells into the unpacking buffers section of the code after the unpacking process has taken place.

Inside of the main unpacking loop, after the rest of the code inside the loop is completed, a second loop is used to compare all possible pairs of faces on a given side.

As with everything else in *exchange_ghostcells*, nothing needs to be done with a face unless it is a communication boundary. Additionally, for this portion of the code, only when both faces talk to the same communication partner do we need to do anything. Both of these conditions are checked before proceeding with the current pair of faces.

At this point, there are only two more conditions that the pair of faces needs to meet for it to be necessary to fix coordinates:

1. The faces must communicate with different sides of the communication partner.
2. The faces much touch each other at more than one point.

To find which side of the communication partner a face talks with, the pairs of high/low communication partner coordinates must be compared. Where the upper and lower coordinates for a specified axis on the communication partner are equal, that axis must be constant for the face. Since adjoining faces on a block must communicate with adjoining faces on the communication partner(s), if they talk to a single communication partner, they must talk to the same or adjoining sides; hence, it is sufficient to establish the axis that the communication partner face coordinates are constant on. This check must be carried out for both faces. If these two checks indicate that the faces talk to the same side on the communication partner (the communication partner faces are constant on the same axis), then there is no reason to continue.

To identify where two faces, which have passed the above tests, touch, if they touch at all, first the high and low i/j/k coordinates for the faces are sorted numerically and compared, looking for a region where the faces meet on one axis and overlap on the other.

If it is found that the faces touch at more than one point, a loop is executed, running through the range of interaction in the varying axis. For each coordinate within this range of interaction, a value indicating the axis of interface is stored to the array *coordpairing*.

Replacing affected coordinates

It was necessary to develop a means of redefining the affected grid point to insure that, as nearly as possible, no matter what the geometry of the two communicating blocks was like, the new coordinate would result in valid, usable cells. This means each new cell must have six sides, with intersections between sides taking place only at the edges (no negative or zero volumes permitted). Ideally, the aspect ratios of the cells should be kept close to one. Initially it was assumed that the new coordinate would simply be the average of the overwritten and over-writing coordinates. This, as it became apparent, would not always produce valid cells. Averaging the coordinates to either side of the interface axis was also considered, but, again, this would not always produce valid cells.

It was noted that, in all cases, there must be at least one row of coordinates before and one row of coordinates after the interface axis. (This corresponds to having at least one row of cells on either side of the interface between faces.) The solution was to adjust the coordinates before and after the interface axis first, before assigning the affected point the position halfway between the coordinates to either side of the interface axis.

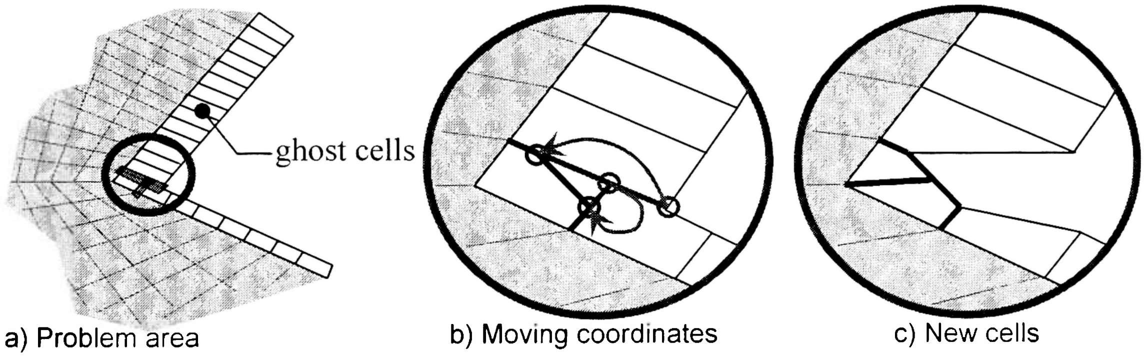


Figure 31: Exchange_ghostcells – topology issue: procedure for correcting coordinates

For each side of the block, all of the ghost cell grid points are looped through. For each grid point, a check is made to see if the *coordpairing* array has a value for an axis of interface, indicating that the coordinate needs to be repaired. The axis value also indicates the precise method by which the repair is to take place.

For the coordinates on either side of the overwritten coordinate, the distances between the boundary and ghost cell coordinates are calculated. The shorter distance is halved and the ghost cell coordinate is relocated accordingly. The length between the other ghost cell coordinate and its corresponding boundary coordinate is scaled back to the same length as the other, shortened distance, and the coordinate is relocated accordingly. The overwritten coordinate is then reassigned to a position halfway between the two redefined coordinates.

Accounting for subdivision

As noted in the "Grid subdivision" section, the program *grid_subdiv* can be used to split cells within a block after the grid has been exported from the grid generation program. While this is useful in allowing the code to deal with more complex geometries, it impacts the subroutine *exchange_ghostcells* in two main ways.

1. The coordinates that define the faces on any given side are corrected by *grid_subdiv* to reflect the new cells added; however, it was not practical to have this change reflected in the communication partner coordinates in the input files for all of the communication partners of a subdivided block. Some translation of the coordinates is thus required.
2. Because the communication partner may or may not be subdivided, and if the partner is subdivided, the subdivision may or may not match up with block sending coordinates, grid subdivision will generally result in non-contiguous interfaces between blocks. To accommodate this non-contiguity, only the original (non-subdivision) coordinates can be sent with the send buffer. On the other hand, for a subdivided block, when the receive buffer is being unpacked, there will be no coordinates in the buffer to be used to fill the subdivision coordinates. Not only are these coordinates missing, but the receive buffer does not contain instructions specifying where the points go. If the coordinates were added directly to the x, y, and z arrays, the subdivision ghost cell coordinates would be filled along with ghost cell coordinates corresponding to preexisting coordinates, skewing

the cells created while leaving other ghost cell coordinates, extrapolated in the *grid* subroutine, unmodified.

Recalculation of the face defining coordinates and the maximum dimension of the block is required to deal with the first item. Dealing with the second point requires:

1. While sending coordinates for grid points for a subdivided cell to the send buffer, only the original grid points are sent.
2. While unpacking coordinates for a subdivided cell from the receive buffer, since only the original coordinates are available, the subdivision coordinates must be interpolated.
3. To ensure that the right value is assigned to the correct grid point, the i/j/k indices must be carefully tracked and corrected for subdivision points added ahead of the index.

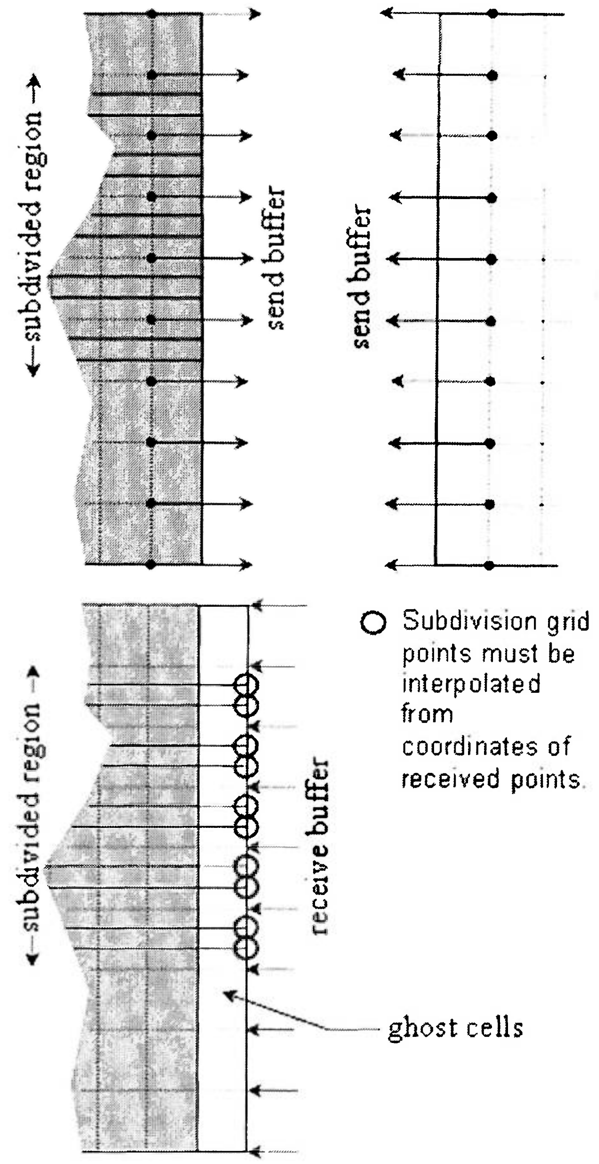


Figure 32: Exchange_ghostcells – grid subdivision

Basic procedure

Preliminary:

The block subdivision instructions that were used by *grid_subdiv* need to be read in from the input file *grid_subdiv.inp* so that they can be used to determine which of the coordinates are original and which are the product of subdivision.

In order to simplify some of the subsequent operations, the face defining coordinates and maximum dimensions are reset to the original (pre-subdivision) values.

Posting receives:

Since only original coordinates (none of the coordinates added in grid subdivision) will be sent to the send buffer, the size of the send and receive buffers do not change. Since the face coordinates and maximum dimensions are reset to pre-subdivision values, no change is required in posting receives.

Posting sends:

The basic procedure used in the unmodified *exchange_ghostcells* still applies, however, it is necessary to include a good deal of logic to ensure that only the original coordinates are sent.

The original set of nested loops, modified only to reflect the correct send coordinates, is retained intact, but is included inside another if/then/else block which checks to see if any of the grid points on the face are the result of subdivision. If there are no subdivision grid points on the face, the original procedure can be used, but if some grid points are not original, a modified version of this procedure must be used.

A set of nested loops going through both of the varying axes is again used. The loop through the second axis will, however, take place only if the index for the first axis corresponds to a series of original grid points. An if/then/else block contained within each of the loops checks to see where the loop index is with respect to the subdivision region.

In the outer loop, if the axis index is outside the subdivision region (before or after the region or at the start coordinate) cells have not been divided along that axis in way of that position along the axis, hence the inner loop can be run directly. If the index is inside the subdivision region, the inner loop executes only if the outer axis index corresponds to an original grid point.

In the inner loop, coordinates are saved to the send buffer only if the inner axis index is outside the subdivision region or corresponds to an original grid point within the subdivision region.

Unpacking receives:

Again, the basic procedure for unpacking receives still applies, but must be supplemented with logic to ensure that the coordinates pulled from the buffer go to the right ghost cell grid point coordinates. Additional calculations must be included to interpolate the values for the subdivision ghost cell coordinates. The subdivision grid points need to be input into the x, y, and z arrays in the correct order. This is complicated by the fact that both axes can be incrementing positively or negatively.

The original set of nested loops, modified only to reflect the correct send coordinates, is retained intact, included inside another if/then/else block which checks to see if any of the grid points on the face are the result of subdivision. If there are no subdivision grid points on the face, the original procedure can be used, but if some grid points are not original, a modified version of this procedure must be used.

Another set of nested loops for the two varying axes is used; however, because different procedures are used depending upon where the grid point is with respect to the subdivision regions along each axis, the loop for the second axis must be repeated for each case in the if/then/else block contained within the loop for the first axis.

As with the send operations, an if/then/else block contained within each of the loops checks to see where the grid point is with respect to the subdivision region along the axis in question.

The basic logic for the loops is essentially the same as that used in the send operations. For the outer loop, if the axis index is outside the subdivision region, the second loop can be run directly. The outer loop index is used inside the inner loops to specify where coordinates will be stored in the coordinate arrays. Where the index follows the subdivision region, the index must be corrected to reflect the number of point added in the subdivision region before the index can be used to specify where coordinates are to be stored in the coordinate arrays.

If the outer loop index is within the subdivision region, for each index value, the inner loop will be accessed first using just the current outer loop index. After this takes place, new coordinates corresponding to the subdivision along the outer axis are interpolated. The inner loop is accessed for each subdivision of the outer loop.

The inner loop normally pulls coordinates from the receive buffer, storing them to the coordinate arrays when the inner loop index is not within the subdivision region. When the outer loop provides interpolated coordinates, these are used by the inner loop instead of coordinates pulled from the receive buffer. Within the subdivision region, the inner loop uses logic similar to that used in the outer loop. For each inner loop index, the first coordinate received is stored to the coordinate arrays. Subdivision coordinates for the cell are then interpolated and stored to the arrays before moving on to the next index.

Finishing code:

The face defining coordinates and maximums dimension must be reverted to the original (post subdivision) coordinates as these values are used in other procedures.

Calculations

Correcting coordinates for subdivision:

Frequent mention is made above to correcting an i/j/k index for subdivision. This means adding in the number of grid points that have been added between the start of subdivision and the referenced grid point. Using index *ilo* as an example:

If $ilo < start\_i$:no change is required. $ilo = ilo$

If $ilo > end\_i$: $ilo = ilo + (end\_i - start\_i + 1) \times (div\_i - 1)$

Otherwise: $ilo = ilo + (ilo - start\_i) \times (div\_i - 1)$

Reversing subdivision correction:

Near the beginning of the subroutine, indices already take into account subdivision, and this needs to be reversed. Applying a little algebra to the above equations, the following conversions are used to reverse the subdivision correction (again, *ilo* is used as an example):

If $ilo < start\_i$:no change is required. $ilo = ilo$

If $ilo > end\_i$: $ilo = ilo - (end\_i - start\_i + 1) \times (div\_i - 1)$

Otherwise: $ilo = (ilo + start\_i \times (div\_i \quad 1)) / div\_i$

Grid point interpolation:

Aside from details of locating the data to interpolate within the buffer, this is just normal interpolation. If, for division in the j direction, looking at the original cell i,3,k, assuming div_j = 5, the next original grid point would be i,8,k. The coordinates for grid point i,j,k would be:

$$x(i,5,k) = x(i,3,k) + (x(i,8,k) - x(i,3,k)) \times (5-3)/div\_j$$
$$y(i,5,k) = y(i,3,k) + (y(i,8,k) - y(i,3,k)) \times (5-3)/div\_j$$
$$z(i,5,k) = z(i,3,k) + (z(i,8,k) - z(i,3,k)) \times (5-3)/div\_j$$

Subroutine exchange

The subroutine *exchange* is also required when MPI is used. Where any two blocks communicate, *exchange* is used to transfer flow information from the first row of cells behind the common communication boundaries to the corresponding ghost cells of the communication partners. See the "Ghost cells" section for an explanation of ghost cells.

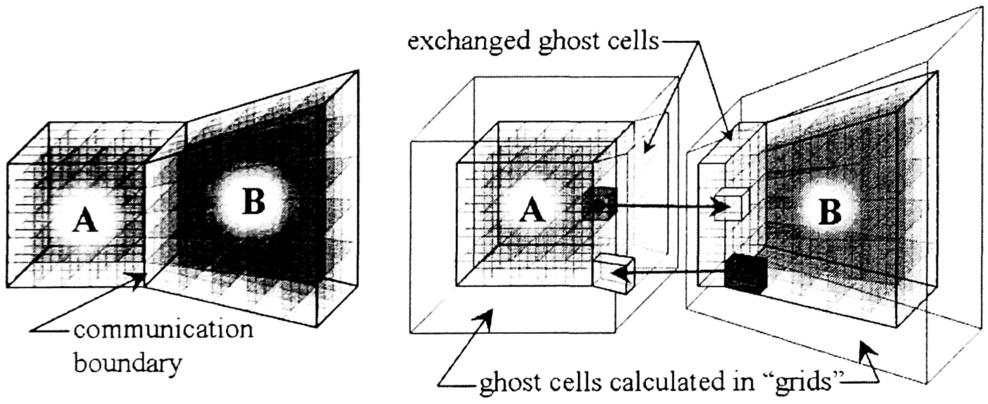


Figure 33: Exchange – transferring Qs

Unmodified subroutine

The unmodified *exchange* subroutine operates according to the following basic procedure:

- Total buffer size is determined based upon the dimensions of the block and the number of values (*nv*) to be included in the Q (flow conditions) array for each cell. Based upon this dimension, the receive buffer is allocated.
- The empty receive buffers are posted.

- The send buffer is allocated using the same buffer size determined for the receive buffer.
- The Qs for the first row of cells behind the face are packed into the send buffer and sent.
- MPI_WAITALL is called, ensuring that no process continues through the code until all of its receive buffers are filled.
- The values in the Q array for the ghost cells are overwritten with the data from the receive buffer.

As can be seen from this abbreviated procedure, the subroutine follows a procedure very similar to that used in the original version of *exchange_ghostcells*. The send and receive buffer sizes, in this subroutine, are dependent upon not only the dimensions of the block, but also the number of values stored in the Q array for each cell. Aside from this difference in buffer size, the MPI_IRECV operations are identical to those used in *exchange_ghostcells*.

In addition to the change in buffer size, the MPI_ISEND and unpacking operations vary from those used in *exchange_ghostcells* in two ways:

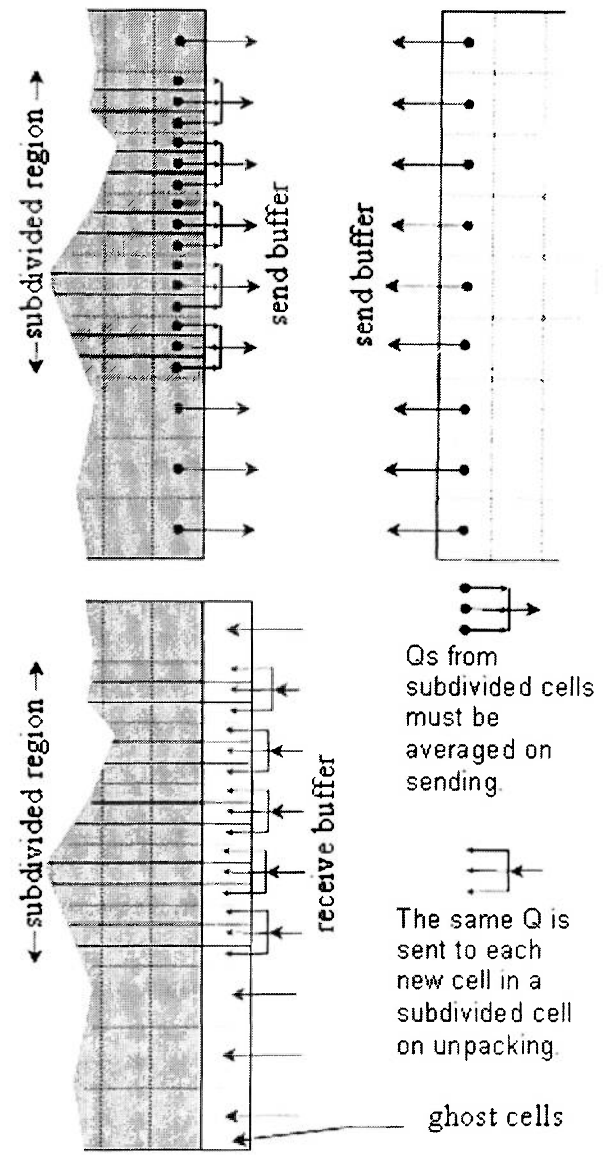
1. Because the subroutine is concerned with the contents of cells rather than grid points, the loops run from one to the axis maximum dimension minus one (rather one to the maximum dimension).
2. A larger number values are transferred to and from a single array, Q, as opposed to transferring only one value from each of the three arrays x, y, and z.

Aside from these two changes, the sending and unpacking procedures are essentially the same as those used in *exchange_ghostcells*.

Modifications to subroutine exchange

While *exchange* does not suffer from the geometry issue that affected *exchange_ghostcells*, it does share the same problems with regards to non-contiguous interfaces between blocks when *grid_subdiv* is used to subdivide cells within a block. In the case of *exchange*:

1. In the MPI_ISEND operations, the value for each flow condition variable in each boundary subdivision cell for a given original cell must be averaged, with only the single set of averaged flow condition variables being stored to the send buffer.
2. In the unpacking operations, since only one set of flow condition variables is available for the original cell, this set of values is assigned to all of the sub-cells for the one original cell.


←subdivided region→
send buffer
send buffer
←subdivided region→
receive buffer
Qs from subdivided cells must be averaged on sending.
The same Q is sent to each new cell in a subdivided cell on unpacking.
ghost cells
Figure 34: Exchange - grid subdivision

Basic procedure

As with *exchange_ghostcells*, the block subdivision instructions that were used by *grid_subdiv* must be read in so that they can be used to determine which cells are original and which are the product of subdivision. In order to simplify some of the subsequent operations, the face coordinates and maximum dimension are then reset to the original (pre-subdivision) values.

For both the send and unpacking operations, the original procedures are again used whenever there are no subdivision cells on the face. When there are subdivision cells on the face, modified versions of the sets of nested loops are required.

Modifications to the nested loops for MPI_ISEND operations:

Because different procedures are used depending upon whether the cell is an original cell or a subdivision cell, the loop for the second axis must be repeated for both cases in the if/then/else block contained within the outer loop. The if/then/else block contained within each of the loops checks to see if the grid point is inside or outside of the subdivision region with respect to the axis in question.

Where the outer loop index is not within the subdivision region, the inner loop is called directly, correcting the outer loop index first if it follows the subdivision region. If the index for the inner loop is not within the subdivision region, the index is corrected if necessary and the Q values are copied to the buffer. If the inner loop index is within the subdivision region, for each index value, the Qs for each of the corresponding subdivision cells are summed and the average set of values are sent to the buffer.

Where the outer loop index falls within the subdivision region, an extra loop is incorporated within the inner loop, summing up the subdivision cells along the outer loop axis as well as, when the inner loop index falls within the subdivision region, along the inner loop axis.

Modifications to the nested loops for unpacking operations:

Because the only change required for this section of the subroutine is ensuring that all of the subdivision cells for a given original cell receive the same set of Qs, the modifications to this section are much simpler.

As with the modifications in the other sections, unless the loop (outer or inner) index is located ahead of the subdivision region with respect to the corresponding axis, the index must be corrected to reflect the number of points added ahead of it. That noted, if the outer loop index is outside the subdivision region, the inner loop can be called. If the inner loop index is also not within the subdivision region, Qs are copied directly from the buffer to the Q array. Otherwise, the inner loop will run through all of the subdivision cells for the original cell corresponding to the current index, copying the current set of Qs to the Q array for each cell before updating the position within the buffer and continuing the inner loop.

If the outer loop index is within the subdivision region, a modified inner loop is used, ensuring that whether the inner loop index is within the subdivision region or not, all of the ghost cells derived from the original cell referenced by the current inner and outer indices receive the same set of Qs.

6. Results of the CFD Analysis

The full model is comprised of 292 separate blocks, totaling 11,864,020 cells. The modifications to *hyp* enabled this very large model to be analyzed. After approximately 1300 iterations, expansions in the nozzles and at the nozzle exits began to develop, with the concurrent decrease in temperature and density. In some of the cells within the expansion regions, the change in temperature and/or pressure in a single iteration was too large for the calculated time step, resulting in a negative temperature or density in the cell.

In order to get around this problem, a program *renumberblocks* was created to transfer the grid, input, and, when requested, restart files for selected blocks (identified in a separated input file, *renumberblocks.inp*) from the current run directory to a smaller run directory on the same level. In doing so, the files are renumbered and the communication partners listed in the input files (*xxx/hyp.xxx.inp*) are changed to reflect the new numbering. A second new program, *returnblocks*, was developed to transfer the restart files back from the smaller run directory to the original run directory, to allow the full analysis to take advantage of the analysis performed on the smaller subset of blocks.

Using the *renumberblocks* utility, the forward 188 blocks of the model were run separately, so that the flow ahead of the nozzles could be fully resolved. This was done in the hopes that, with the decreased flow speed behind the shocks, the effects of expansions within and after the nozzle would be decreased, enabling calculations to be completed with the full model once the flow was resolved heading into the combustor.

Analysis of this subset of blocks was carried out for almost 30,000 iterations before the allocated computer time on the super computer cluster Lemieux ran out. No problems developed during this time period, even with increased Courant-Friedrich-Lewy (CFL) number, corresponding to an increased time step. Due to the scale of the model, even this number of iterations only partially resolved the flow.

While, due to time constraints, a fully resolved 3D analysis could not be completed, with some reengineering, the code proved capable of analyzing a very large three dimensional model. In the process, a number of useful external utilities were produced, further increasing the versatility of the code.

Results of the CFD analysis were examined using the 3-D plotting program Tecplot. Screen shots of the results of the analysis of the forward 188 blocks are provided in Appendix 3.

7. Recommendations for Future Work

The following need to be implemented before the CFD analysis of the design can be considered complete:

Viscosity must be turned on. This will result in a boundary layer being generated. As noted previously, the grid is already set up to accommodate this. Full chemical kinetic modeling of the flow must be implemented so that the effects of dissociation can be accommodated. This should result in some deformation of the shock waves. With these two modifications, the flow path analysis must be completed, and the inlet must be repositioned to insure the new inlet shock is placed precisely on the lip of the combustor cowl.

Diffusion modeling must be implemented and a second free stream condition, for the fuel injection at the porous wall, must be added and activated. Global reaction rate combustion modeling must be implemented to model the propane combustion. With these sections of the code activated, the model must be run again. The positions of the ramps may need to be adjusted slightly, due to the effects of dissociation and fuel injection.

If it is found that a significant degree of combustion is taking place ahead of the combustion chamber, the fuel injection point will need to be adjusted. Due to modifications made to the code since the grid was initially developed, a separate block will not be required to apply this new fuel injection point. A selected region of an existing surface domain can be split off as a separate face, with a separated inlet condition just for that face.

If it is found that combustion is not substantially completed within the combustion chamber, one of the alternate means of initiating combustion, discussed in the ignition section of the Preliminary Design section, will have to be used. It is recommended that both methods be examined for their potential impacts on the craft's performance.

Once it is ensured that combustion is taking place correctly, the pressure distribution along the craft will need to be analyzed to calculate the net thrust generated by the craft. The velocity distribution along the craft may also be used to approximate frictional drag.

Up to this point, the analysis will only have been carried out at a single flight condition: Scramjet mode flying at Mach 5 at an altitude of 24.8 km. As the craft is expected to be able to travel at speeds significantly higher than Mach 5, the above analysis should be carried out at a variety of Mach numbers and altitudes to obtain a more accurate picture of its potential operating regime. Additionally, an analysis should be carried out for the craft operating in rocket propulsion mode at zero atmospheric pressure.

If a subsonic / low supersonic propulsion system can be integrated into the craft, this design should also be examined. If rocket propulsion is required to accelerate the craft to a speed at which scramjet propulsion can take over, analysis of the rocket propulsion system at the pertinent flight conditions should also take place.

Based upon the flow conditions observed inside the nozzles in the above tests, it must be determined where the aft end of the cowl can be truncated without significantly reducing thrust throughout the craft's flight profile. Once the model is modified accordingly, further CFD analyses in scramjet and rocket propulsion modes should be carried out to confirm the design decision.

References

[1]Perrell, E. R., Erickson, W.D., Candler, G.V., "Numerical Simulation of Nonequilibrium Condensation in a Hypersonic Wind Tunnel," *J. Thermophysics and Heat Transfer*, Vol. 10, No. 2, April-June 1996, pp. 277-283.

[2]Boudreau, Albert H., "Status of the US Air Force HyTECH Program," *12th AIAA International Space Planes and Hypersonic Systems Technologies*, AIAA 2003-6947, AIAA, December 15-19, 2003.

[3] Kanda, Takeshi and Kudo, Kenji, "Conceptual Study of a Combined-Cycle Engine for an Aerospace Plane," *Journal of Propulsion and Power*, Vol. 19, No. 5, September-October 2003, pp. 859-867.

[4] Edwards, Tim and Meyer, Michael L., "Propellant Requirements for Future Aerospace Propulsion Systems," *38th AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, AIAA 2002-3870, AIAA, July 7-10, 2002.

[5] Wang, T.-S., "Thermophysics Characterization of Kerosene Combustion," *Journal of Thermophysics and Heat Transfer*, Vol. 15, No. 2, April 2001, pp. 140-147.

[6] Elliott, L. et al., "A Novel Approach To The Reduction Of An Aviation Fuel/Air Reaction Mechanism Using A Genetic Algorithm," *ASME Turbo Expo 2004: Land, Sea And Air*, GT2004-53053, Vol. ?, ASME, June 14-17, 2004, pp. ?, submitted for publication.

[7] Guinan, Daniel P., Alan Drake, Dean Andreadis, Stephen A. Beckel, United States Patent Application 2004006508, "Variable geometry inlet design for scram jet engine", Serial No. 264172, Series Code: 10, filed October 2, 2002.

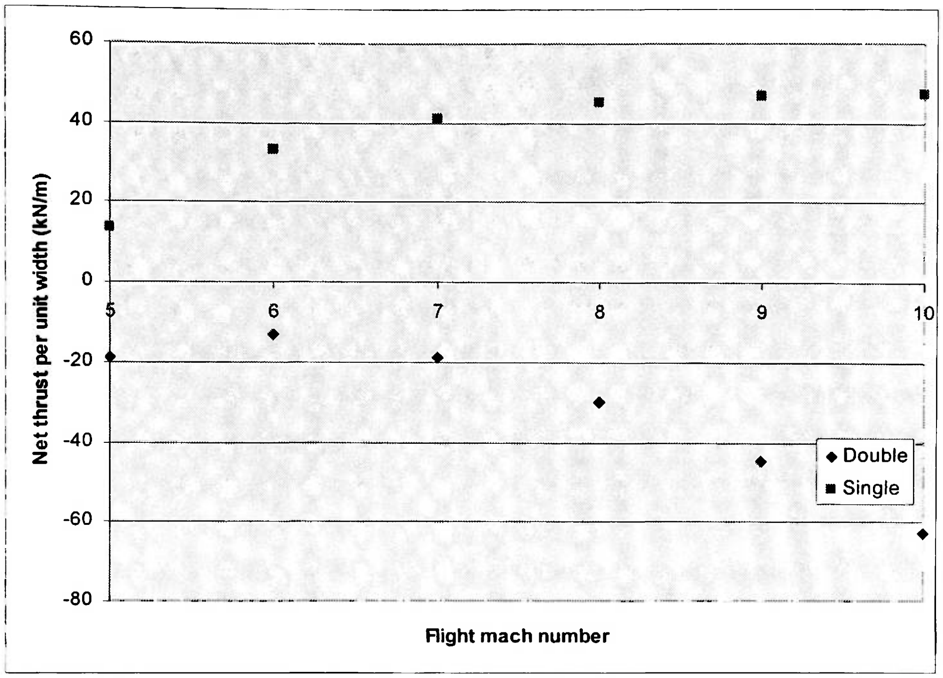
[8]Anderson, John D., *Modern Compressible Flow, with historical perspectve*, 3rd ed., McGraw-Hill, New York, NY, 2003, Chaps. 2-4.

[9] Turns, Stephen R., *An Introduction to Combustion, Concepts and Applications*, 2nd ed., McGraw-Hill, New York, NY, 2000 , Chaps. 2 & 5.

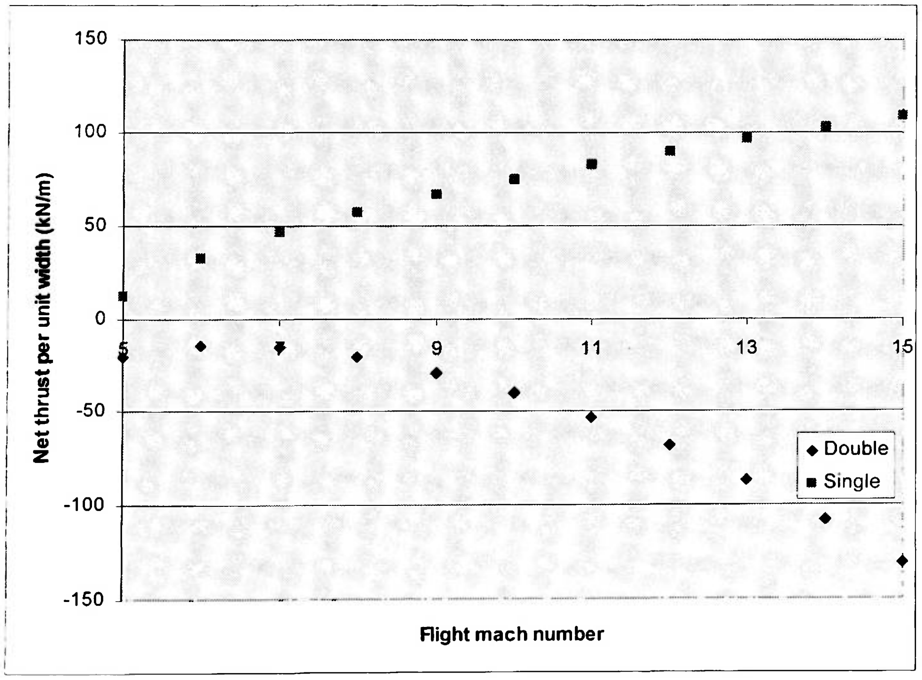
[10]Schlichting, Hermann, *Boundary Layer Theory*, 7th ed., McGraw-Hill, New York, NY, 1979, page 336

[11]Pacheco, Peter S., *Parallel Programming with MPI*, Morgan Kaufmann Publishers Inc., San Francisco, California, 1997 , Chaps. 3, 5, & 13.

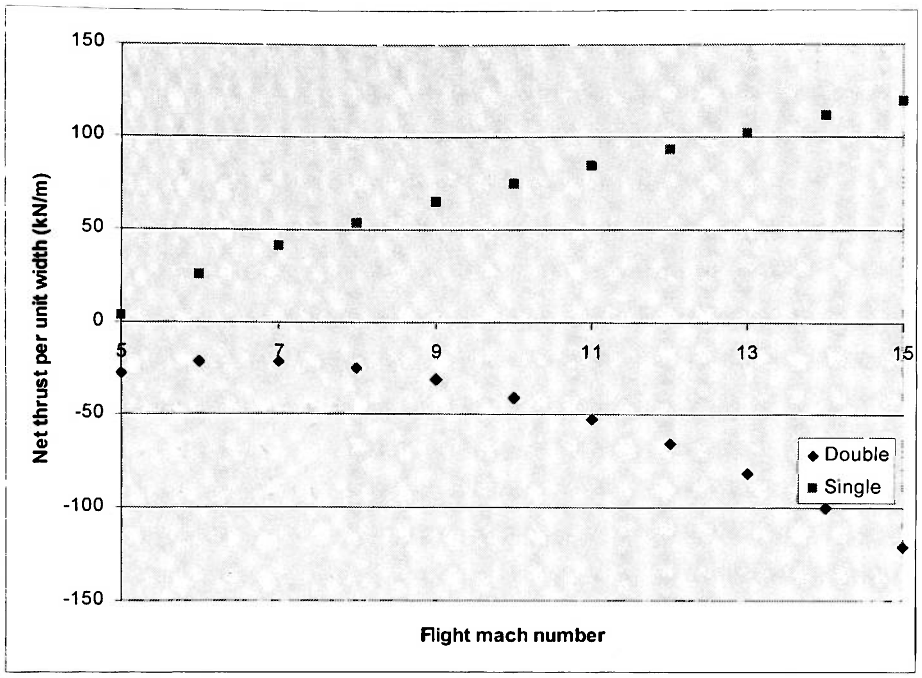
Appendix 1: 2 D analysis - Net thrust / unit width vs flight Mach number, select mods



Mod 1

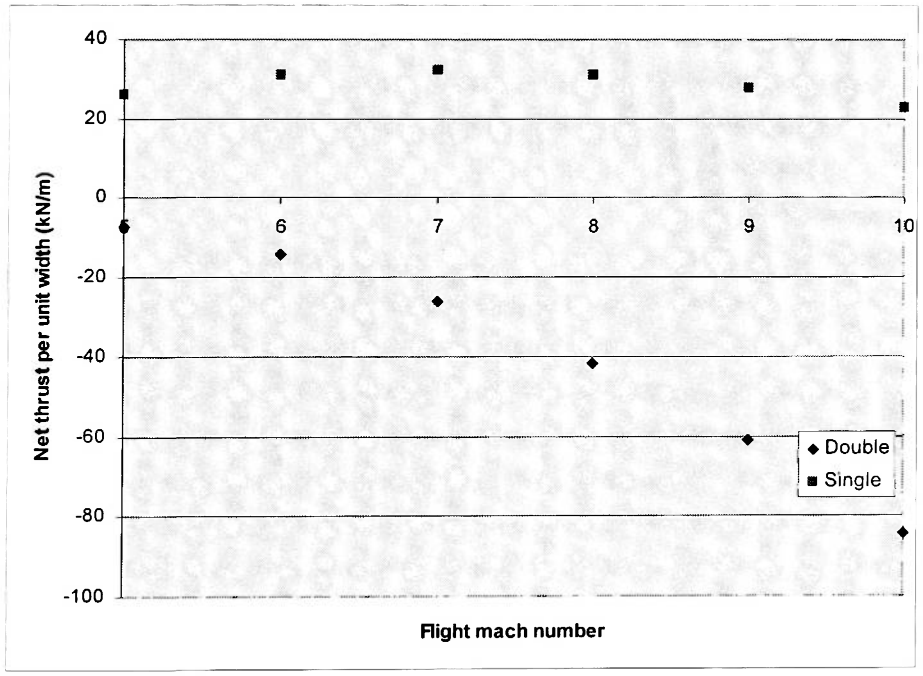


Mod 3



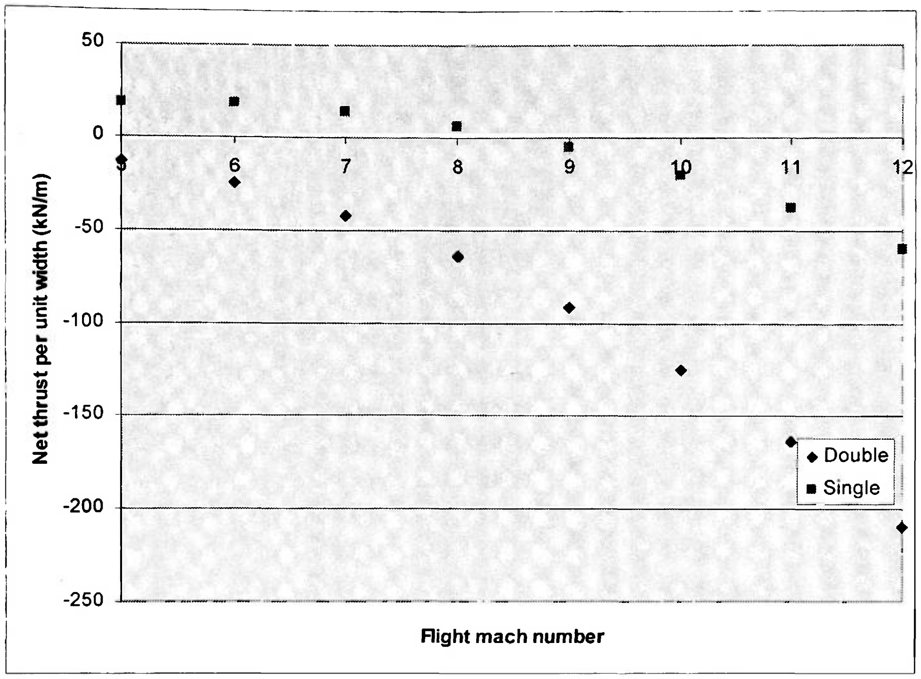
150
100
50
0
-50
-100
-150
5
7
9
11
13
15
Net thrust per unit width (kN/m)
Flight mach number
◆ Double
■ Single

Mod 5

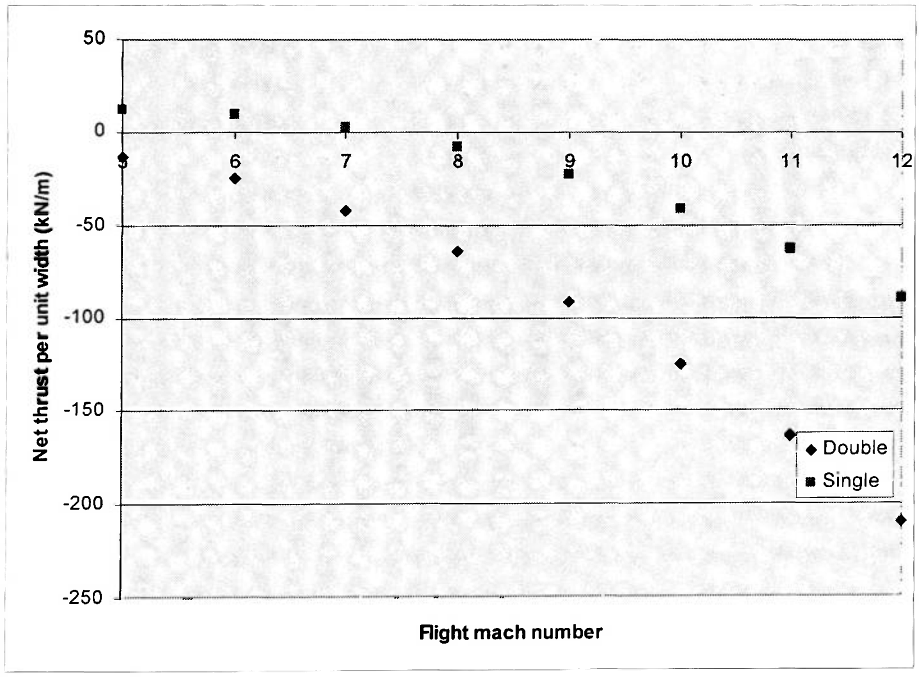


40
20
0
-20
-40
-60
-80
-100
5
6
7
8
9
10
Net thrust per unit width (kN/m)
Flight mach number
◆ Double
■ Single

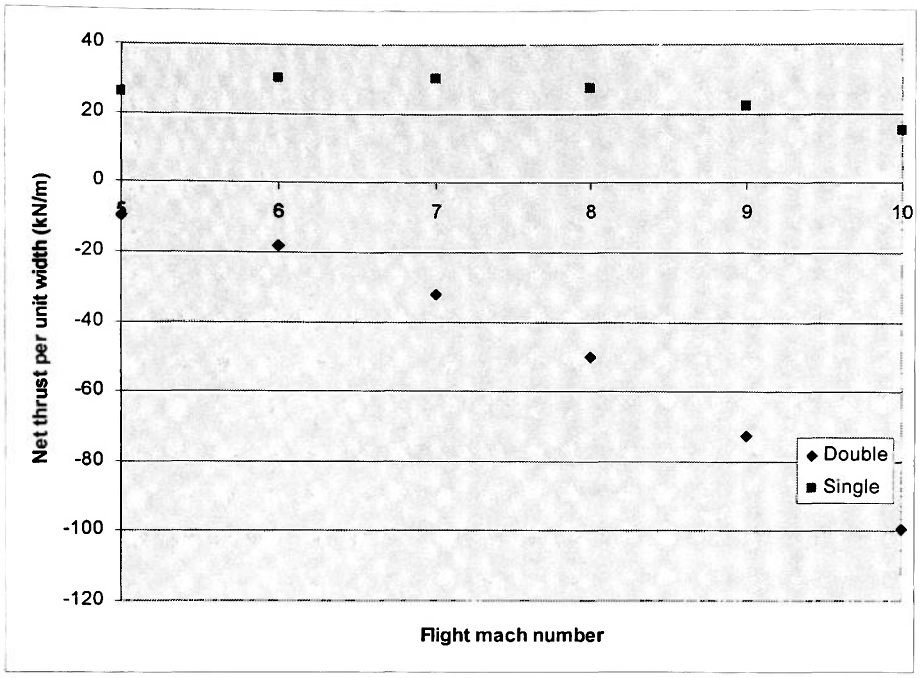
Mod 6


50
0
-50
-100
-150
-200
-250
5
6
7
8
9
10
11
12
Net thrust per unit width (kN/m)
Flight mach number
◆ Double
■ Single

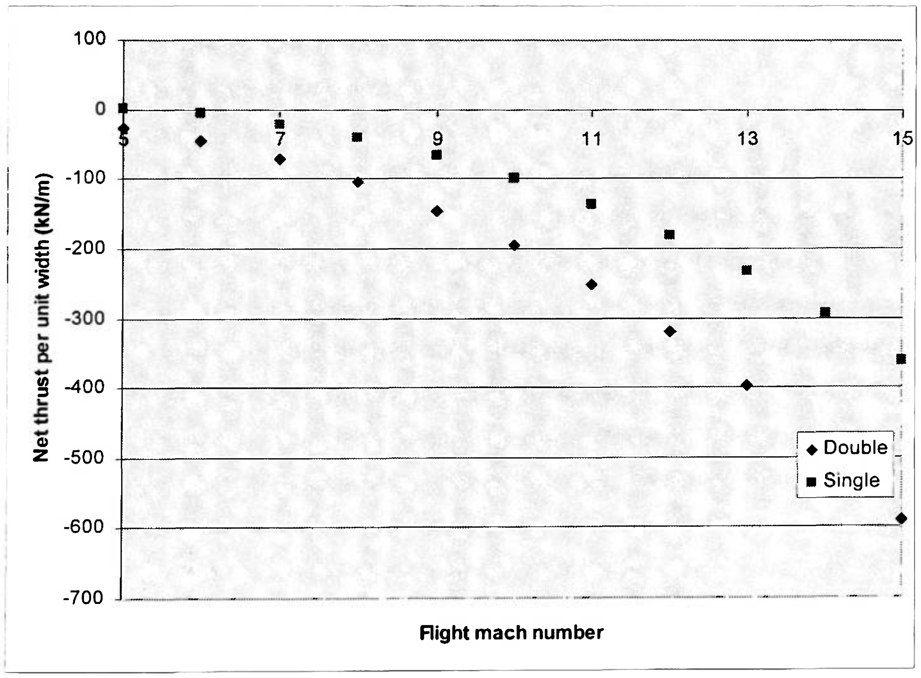
Mod 8


50
0
-50
-100
-150
-200
-250
5
6
7
8
9
10
11
12
Net thrust per unit width (kN/m)
Flight mach number
◆ Double
■ Single

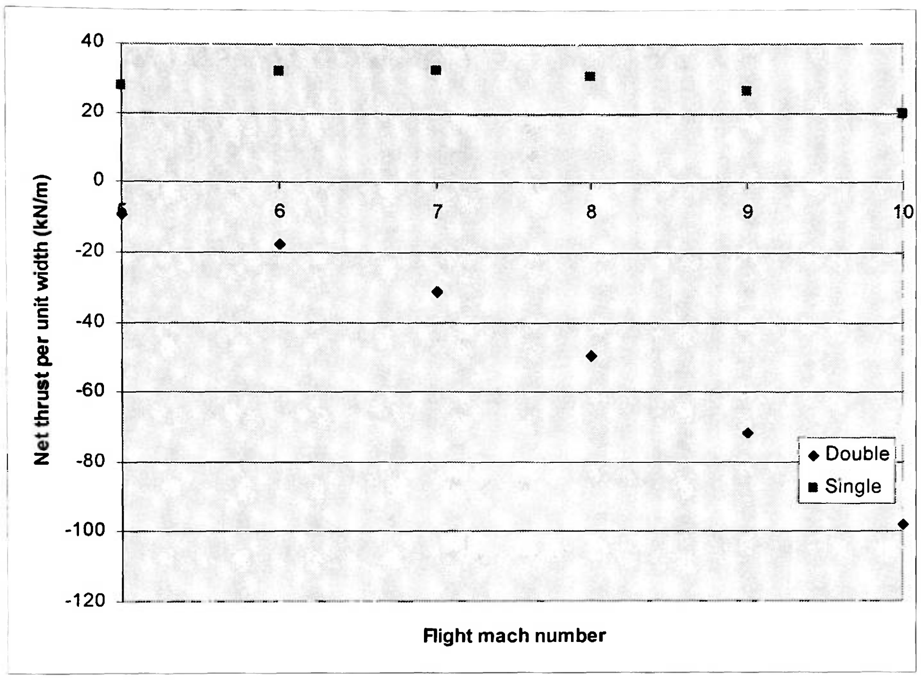
Mod 10


40
20
0
-20
-40
-60
-80
-100
-120
5
6
7
8
9
10
Net thrust per unit width (kN/m)
Flight mach number
Double
Single

Mod 12

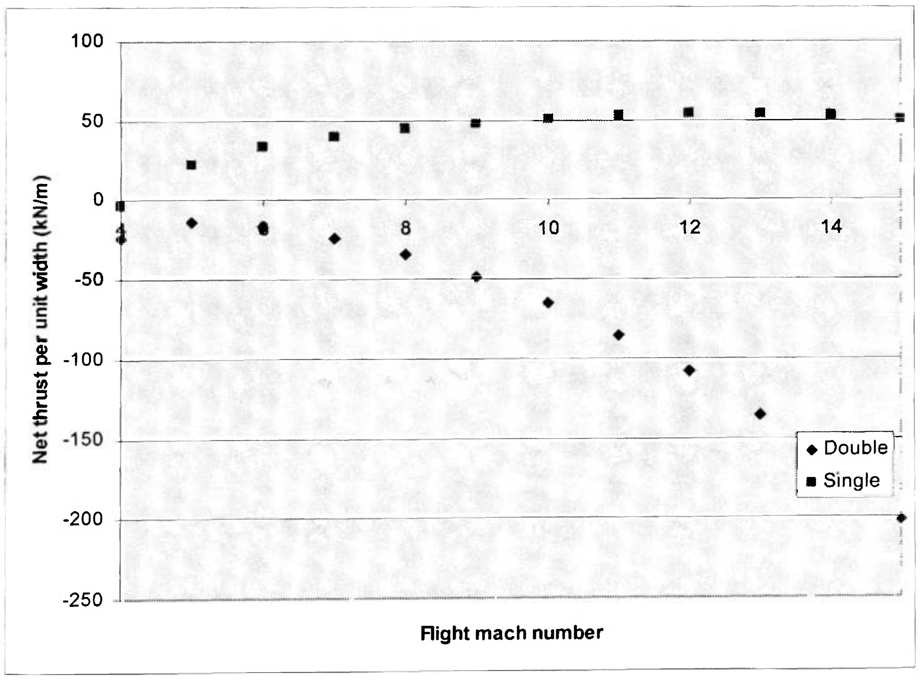

100
0
-100
-200
-300
-400
-500
-600
-700
5
7
9
11
13
15
Net thrust per unit width (kN/m)
Flight mach number
Double
Single

Mod 13



40
20
0
-20
-40
-60
-80
-100
-120
5
6
7
8
9
10
Net thrust per unit width (kN/m)
Flight mach number
Double
Single

Mod 18



100
50
0
-50
-100
-150
-200
-250
4
8
10
12
14
Net thrust per unit width (kN/m)
Flight mach number
Double
Single

Mod 26

Appendix 2: List of blocks in final grid

	Dimensions			Total	Contact	
Block	**Side 1**	**Side 2**	**Side 3**	**Cells**	**w/ hull**	**Description**
1	31	26	51	37,500	Y	right combustion chamber, aft half
2	31	26	51	37,500	Y	left combustion chamber, aft half
3	24	51	31	34,500	Y	inlet region
4	71	11	26	17,500	Y	inlet region
5	71	11	36	24,500	Y	inlet region
6	71	106	6	36,750	Y	pre-injection plane
7	71	106	6	36,750	Y	fuel injection plane
8	6	51	11	2,500	Y	fillet to the right of the bow
9	11	126	11	12,500	Y	air forward of the bow fillet and fillet to right of bow
10	31	26	56	41,250	N	forward most block, upper, right of craft
11	31	26	56	41,250	N	forward most block, lower, in line with craft
12	23	32	61	40,920	N	air to side of craft, lower
13	43	35	31	42,840	N	airmass above block 85
14	56	26	31	41,250	N	airmass below block 151
15	26	31	51	37,500	N	lower aft most corner airblock
16	31	26	51	37,500	N	upper aft most corner airblock
17	6	6	51	1,250	Y	air to side of nozzle upper fillet
18	6	6	51	1,250	Y	air to side of nozzle lower fillet
19	21	51	43	42,000	Y	air to right of craft, above blocks 21 and 23
20	42	21	51	41,000	Y	air to right of bow
21	51	36	16	26,250	Y	air to right of inlet craft iwo inlet region
22	36	51	16	26,250	Y	air to right of inlet craft iwo inlet region
23	24	51	36	40,250	Y	air to right of inlet craft iwo inlet region
24	26	51	35	42,500	Y	air to right of craft, iwo combustor region
25	31	26	51	37,500	Y	right combustion chamber, forward half
26	31	26	51	37,500	Y	left combustion chamber, forward half
27	24	51	31	34,500	Y	inlet region
28	24	51	31	34,500	N	inlet region
29	25	51	31	36,000	Y	inlet region
30	24	51	31	34,500	Y	inlet region
31	25	51	31	36,000	Y	inlet region
32	27	31	51	39,000	Y	left nozzle
33	27	31	51	39,000	Y	right nozzle
34	31	27	51	39,000	Y	left nozzle
35	31	27	51	39,000	Y	right nozzle
36	31	27	51	39,000	Y	left nozzle
37	31	27	51	39,000	Y	right nozzle
38	31	27	51	39,000	Y	left nozzle
39	31	27	51	39,000	Y	right nozzle
40	31	27	51	39,000	Y	left nozzle
41	31	27	51	39,000	Y	right nozzle
42	31	26	56	41,250	N	forward most block, upper, in line with craft
43	31	26	56	41,250	N	forward most block, upper, in line with craft
44	31	26	56	41,250	N	forward most block, upper, right of craft
45	31	26	56	41,250	N	forward most block, upper, right of craft

46	31	26	56	41,250	N	forward most block, lower, in line with craft
47	31	26	56	41,250	N	forward most block, lower, in line with craft
48	31	26	56	41,250	N	forward most block, lower, right of craft
49	31	26	56	41,250	N	forward most block, lower, right of craft
50	23	32	61	40,920	Y	air below lower bow plating and to right of craft
51	23	32	61	40,920	Y	air below lower bow plating
52	23	33	61	42,240	Y	air below lower bow plating
53	24	33	61	44,160	Y	air below lower bow plating
54	24	33	61	44,160	Y	air below lower bow plating
55	24	33	61	44,160	Y	air below lower bow plating
56	24	33	61	44,160	Y	air below lower bow plating
57	24	33	61	44,160	Y	air below lower bow plating
58	24	33	61	44,160	Y	air below lower bow plating
59	24	33	61	44,160	Y	air below lower bow plating
60	23	33	61	42,240	Y	air below lower bow plating
61	24	32	61	42,780	Y	air below lower bow plating
62	24	32	61	42,780	Y	air below lower bow plating
63	24	32	61	42,780	Y	air below lower bow plating
64	24	32	61	42,780	Y	air below lower bow plating
65	24	32	61	42,780	Y	air below lower bow plating
66	24	32	61	42,780	Y	air below lower bow plating
67	24	32	61	42,780	Y	air below lower bow plating
68	23	32	61	40,920	Y	air below lower bow plating
69	24	32	61	42,780	Y	air below lower bow plating and to right of craft
70	24	32	61	42,780	Y	air below lower bow plating and to right of craft
71	24	32	61	42,780	Y	air below lower bow plating and to right of craft
72	24	32	61	42,780	Y	air below lower bow plating and to right of craft
73	24	32	61	42,780	Y	air below lower bow plating and to right of craft
74	24	32	61	42,780	Y	air below lower bow plating and to right of craft
75	24	32	61	42,780	Y	air below lower bow plating and to right of craft
76	23	32	61	40,920	Y	air below lower bow plating and to right of craft
77	24	32	61	42,780	N	air to side of craft, lower
78	24	32	61	42,780	N	air to side of craft, lower
79	24	32	61	42,780	N	air to side of craft, lower
80	24	32	61	42,780	N	air to side of craft, lower
81	24	32	61	42,780	N	air to side of craft, lower
82	24	32	61	42,780	N	air to side of craft, lower
83	24	32	61	42,780	N	air to side of craft, lower
84	23	32	61	40,920	N	air to side of craft, lower
85	43	35	31	42,840	Y	air above upper bow plating
86	43	35	31	42,840	Y	air above upper bow plating
87	43	35	31	42,840	Y	air above upper bow plating
88	43	35	31	42,840	Y	air above upper bow plating
89	43	35	31	42,840	Y	air above upper bow plating
90	43	36	31	44,100	Y	air above upper bow plating
91	43	35	31	42,840	N	airmass above block 86
92	43	35	31	42,840	N	airmass above block 87
93	43	35	31	42,840	N	airmass above block 88
94	43	35	31	42,840	N	airmass above block 89
95	43	36	31	44,100	N	airmass above block 90

96	43	36	31	44,100	Y	air above upper bow plating
97	43	36	31	44,100	Y	air above upper bow plating
98	43	36	31	44,100	Y	air above upper bow plating
99	43	36	31	44,100	Y	air above upper bow plating
100	43	36	31	44,100	Y	air above upper bow plating
101	43	36	31	44,100	N	airmass above block 96
102	43	36	31	44,100	N	airmass above block 97
103	43	36	31	44,100	N	airmass above block 98
104	43	36	31	44,100	N	airmass above block 99
105	43	36	31	44,100	N	airmass above block 100
106	43	35	31	42,840	N	airmass above block 108
107	42	35	31	41,820	N	airmass above block 109
108	43	35	31	42,840	Y	air above upper bow plating, extends to right of craft
109	42	35	31	41,820	N	airmass to right of block 108
110	43	35	31	42,840	Y	air above upper bow plating, extends to right of craft
111	42	35	31	41,820	N	airmass to right of block 110
112	43	35	31	42,840	Y	air above upper bow plating, extends to right of craft
113	42	35	31	41,820	N	airmass to right of block 112
114	43	35	31	42,840	Y	air above upper bow plating, extends to right of craft
115	42	35	31	41,820	N	airmass to right of block 114
116	43	35	31	42,840	Y	air above upper bow plating, extends to right of craft
117	42	35	31	41,820	N	airmass to right of block 116
118	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
119	42	36	31	43,050	N	airmass to right of block 118
120	43	35	31	42,840	N	airmass above of block 110
121	42	35	31	41,820	N	airmass above of block 111
122	43	35	31	42,840	N	airmass above of block 112
123	42	35	31	41,820	N	airmass above of block 113
124	43	35	31	42,840	N	airmass above of block 114
125	42	35	31	41,820	N	airmass above of block 115
126	43	35	31	42,840	N	airmass above of block 116
127	42	35	31	41,820	N	airmass above of block 117
128	43	36	31	44,100	N	airmass above of block 118
129	42	36	31	43,050	N	airmass above of block 119
130	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
131	42	36	31	43,050	N	airmass to right of block 130
132	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
133	42	36	31	43,050	N	airmass to right of block 132
134	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
135	42	36	31	43,050	N	airmass to right of block 134
136	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
137	42	36	31	43,050	N	airmass to right of block 136
138	43	36	31	44,100	Y	air above upper bow plating, extends to right of craft
139	42	36	31	43,050	N	airmass to right of block 138
140	43	36	31	44,100	N	airmass above of block 130
141	42	36	31	43,050	N	airmass above of block 131
142	43	36	31	44,100	N	airmass above of block 132
143	42	36	31	43,050	N	airmass above of block 133
144	43	36	31	44,100	N	airmass above of block 134
145	42	36	31	43,050	N	airmass above of block 135

146	43	36	31	44,100	N	aırmass above of block 136
147	42	36	31	43,050	N	aırmass above of block 137
148	43	36	31	44,100	N	aırmass above of block 138
149	42	36	31	43,050	N	aırmass above of block 139
150	32	23	61	40,920	Y	aır below nozzle
151	56	26	31	41,250	N	outsıde aır block below blocks 22 and 192
152	56	26	31	41,250	N	aırmass below block 152
153	56	26	31	41,250	N	outsıde aır block below blocks 21 and 192
154	56	26	31	41,250	N	aırmass below block 158
155	56	26	31	41,250	N	aırmass below block 159
156	56	26	31	41,250	N	aırmass below block 160
157	56	26	31	41,250	N	aırmass below block 161
158	56	26	31	41,250	N	ınsıde aır block below blocks 22 and 192
159	56	26	31	41,250	Y	aırmass below ınlet regıon
160	56	26	31	41,250	Y	aırmass below ınlet regıon
161	56	26	31	41,250	Y	aırmass below ınlet regıon
162	56	26	31	41,250	N	aırmass below block 166
163	56	26	31	41,250	N	aırmass below block 167
164	56	26	31	41,250	N	aırmass below block 168
165	56	26	31	41,250	N	aırmass below block 169
166	56	26	31	41,250	N	ınsıde aır block below blocks 21 and 192
167	56	26	31	41,250	Y	aırmass below ınlet regıon
168	56	26	31	41,250	N	aırmass below ınlet regıon
169	56	26	31	41,250	N	aırmass below ınlet regıon
170	26	31	51	37,500	N	aırmass to rıght of block 171
171	26	31	51	37,500	N	aırmass aft/below nozzles
172	26	31	51	37,500	N	aırmass aft/below nozzles
173	26	31	51	37,500	N	aırmass aft/below nozzles
174	33	41	31	38,400	N	aırmass to rıght of block 175
175	41	32	31	37,200	Y	aırmass behınd lower half of nozzle, extends past craft
176	41	32	31	37,200	Y	aırmass behınd lower half of nozzle
177	41	32	31	37,200	Y	aırmass behınd upper half of nozzle
178	41	32	31	37,200	Y	aırmass behınd lower half of nozzle
179	41	32	31	37,200	Y	aırmass behınd upper half of nozzle
180	31	26	51	37,500	N	aırmass to rıght of block 181
181	31	26	51	37,500	N	aırmass aft/above nozzles
182	31	26	51	37,500	N	aırmass aft/above nozzles
183	31	26	51	37,500	N	aırmass aft/above nozzles
184	41	19	61	43,200	Y	aırmass above craft's horızonal upper surface
185	21	51	43	42,000	Y	aır to rıght of craft upper body
186	21	51	43	42,000	Y	aır to rıght of craft upper body
187	21	51	42	41,000	Y	aır to rıght of craft upper body
188	42	21	51	41,000	Y	aır to rıght of bow
189	42	21	51	41,000	Y	aır to rıght of bow
190	42	21	51	41,000	Y	aır to rıght of bow
191	21	51	42	41,000	Y	aır to rıght of bow
192	36	51	16	26,250	Y	aır to rıght of ınlet craft ıwo ınlet regıon
193	24	51	36	40,250	Y	aır to rıght of ınlet craft ıwo ınlet regıon
194	25	51	36	42,000	Y	aır to rıght of ınlet craft ıwo ınlet regıon
195	26	51	35	42,500	N	aırmass to rıght of block 24

196	26	51	35	42,500	Y	air to right of nozzle
197	26	51	35	42,500	Y	air to right of nozzle
198	26	51	35	42,500	Y	air to right of nozzle
199	26	51	35	42,500	Y	air to right of nozzle
200	26	51	35	42,500	N	airmass to right of block 196
201	26	51	35	42,500	N	airmass to right of block 197
202	26	51	35	42,500	N	airmass to right of block 198
203	26	51	35	42,500	N	airmass to right of block 199
204	32	23	61	40,920	Y	air below nozzle
205	32	23	61	40,920	Y	air below nozzle, extends past the craft to the right
206	33	23	61	42,240	N	airmass to the right of block 205
207	32	24	61	42,780	Y	air below nozzle
208	32	24	61	42,780	Y	air below nozzle
209	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
210	33	24	61	44,160	N	airmass to the right of block 205
211	32	24	61	42,780	Y	air below nozzle
212	32	24	61	42,780	Y	air below nozzle
213	32	24	61	42,780	Y	air below nozzle
214	32	24	61	42,780	Y	air below nozzle
215	32	24	61	42,780	Y	air below nozzle
216	32	24	61	42,780	Y	air below nozzle
217	32	23	61	40,920	Y	air below combustion chambers
218	32	24	61	42,780	Y	air below nozzle
219	32	24	61	42,780	Y	air below nozzle
220	32	24	61	42,780	Y	air below nozzle
221	32	24	61	42,780	Y	air below nozzle
222	32	24	61	42,780	Y	air below nozzle
223	32	24	61	42,780	Y	air below nozzle
224	32	23	61	40,920	Y	air below combustion chambers
225	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
226	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
227	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
228	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
229	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
230	32	24	61	42,780	Y	air below nozzle, extends past the craft to the right
231	32	23	61	40,920	Y	air below combustion chambers, extends past craft
232	33	24	61	44,160	N	airmass to right of block 225
233	33	24	61	44,160	N	airmass to right of block 226
234	33	24	61	44,160	N	airmass to right of block 227
235	33	24	61	44,160	N	airmass to right of block 228
236	33	24	61	44,160	N	airmass to right of block 229
237	33	24	61	44,160	N	airmass to right of block 230
238	33	23	61	42,240	N	airmass to right of block 231
239	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
240	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
241	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
242	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
243	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
244	41	19	61	43,200	N	inner airmass right of region above upper horiz surface
245	41	61	18	40,800	N	outer airmass right of region above upper horiz surface

246	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
247	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
248	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
249	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
250	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
251	41	19	61	43,200	Y	airmass above craft's horizonal upper surface
252	41	19	50	35,280	N	inner airmass right of region above upper horiz surface
253	31	33	41	38,400	N	airmass to right of block 259
254	36	61	21	42,000	Y	inlet region
255	36	61	21	42,000	Y	inlet region
256	36	26	38	32,375	Y	inlet region
257	36	26	36	30,625	Y	inlet region
258	12	26	16	4,125	Y	left nozzle
259	41	31	32	37,200	Y	aft of nozzles, top
260	16	12	26	4,125	Y	right nozzle
261	61	11	51	30,000	Y	sheet to right of nozzle exit
262	41	19	61	43,200	Y	airmass above craft's horizonal upper surface, extends past craft to the right
263	41	19	61	43,200	Y	airmass above craft's horizonal upper surface, extends past craft to the right
264	372	3	21	14,840	Y	air to right of craft upper body
265	39	49	21	36,480	N	airmass to right of block 264
266	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
267	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
268	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
269	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
270	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
271	31	51	21	30,000	Y	air to right of craft upper body
272	36	26	51	43,750	Y	air to right of nozzle
273	36	26	51	43,750	N	airmass to right of block 272
274	141	25	12	36,960	N	airmass above/right of craft's horizonal upper surface
275	141	28	12	41,580	N	airmass above/right of craft's horizonal upper surface
276	141	25	12	36,960	N	airmass above/right of craft's horizonal upper surface
277	12	18	41	7,480	N	airmass above/right of craft's horizonal upper surface
278	19	12	41	7,920	N	airmass above/right of craft's horizonal upper surface
279	51	42	21	41,000	Y	air to right of craft upper body
280	21	51	42	41,000	Y	air to right of craft upper body
281	19	61	41	43,200	Y	airmass above craft's horizonal upper surface, extends past craft to the right
282	19	50	41	35,280	N	inner airmass right of region above upper horiz surface
283	50	18	41	33,320	N	outer airmass right of region above upper horiz surface
284	18	50	41	33,320	N	outer airmass right of region above upper horiz surface
285	31	29	26	21,000	Y	left nozzle
286	12	26	16	4,125	Y	right nozzle
287	29	26	31	21,000	Y	left nozzle
288	16	12	26	4,125	Y	right nozzle
289	12	26	16	4,125	Y	left nozzle
290	16	12	26	4,125	Y	right nozzle
291	12	26	16	4,125	Y	left nozzle
292	16	12	26	4,125	Y	right nozzle

Appendix 3: CFD analysis – screen-shots

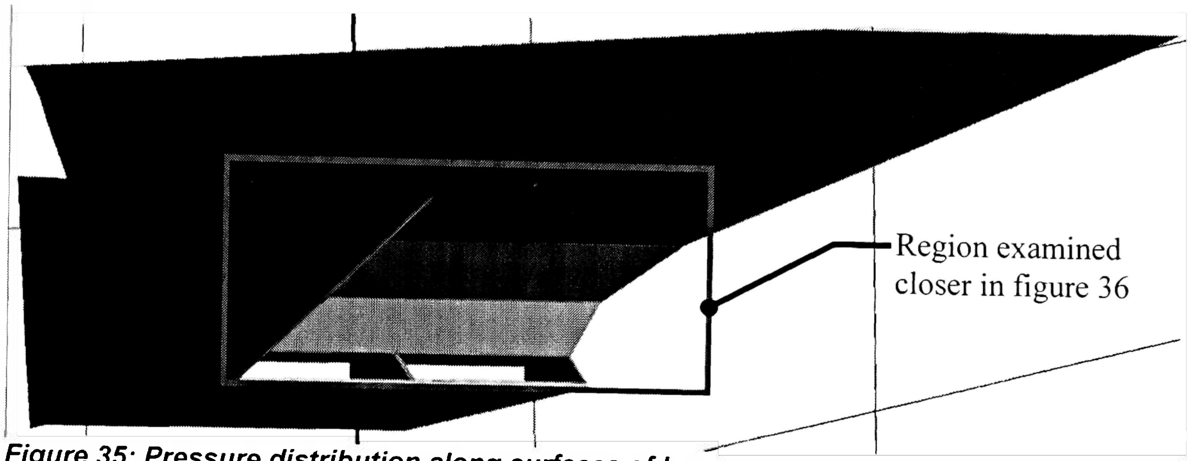


Figure 35: Pressure distribution along surfaces of bow

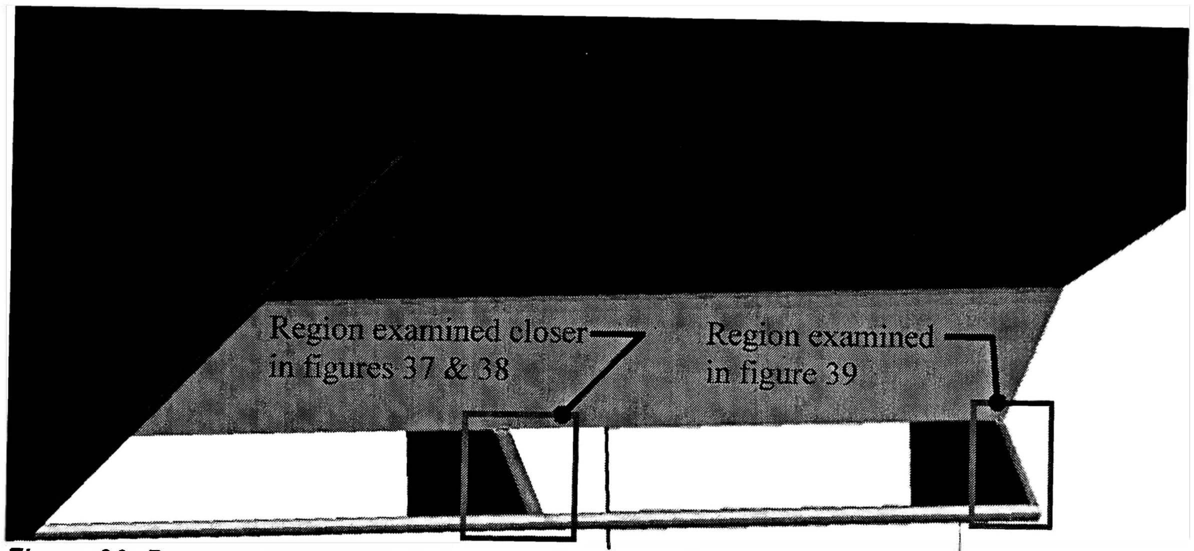


Figure 36: Pressure distribution along surfaces of inlet region

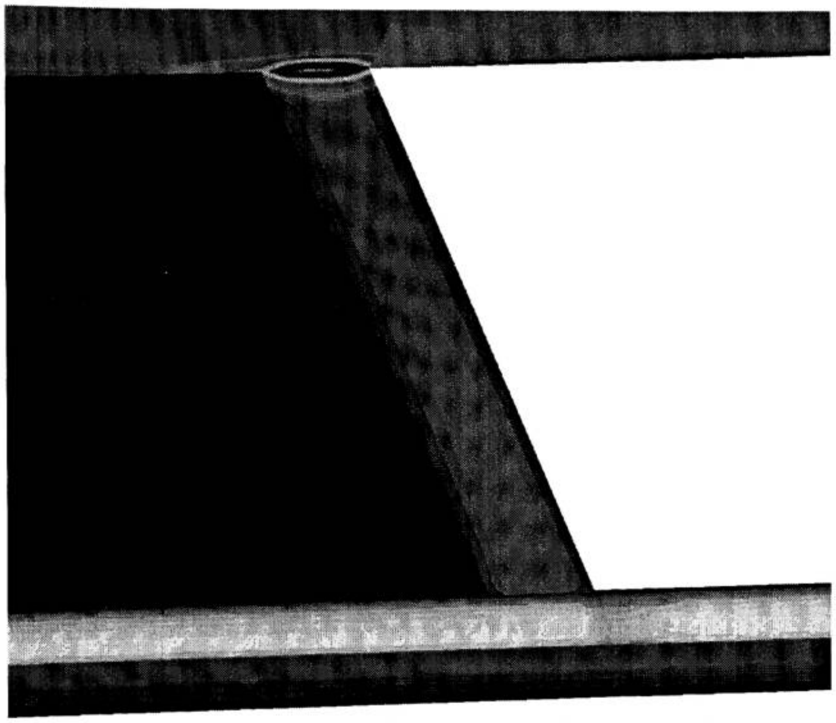


Figure 37: Pressure distribution along surfaces - leading edge of inlet baffle

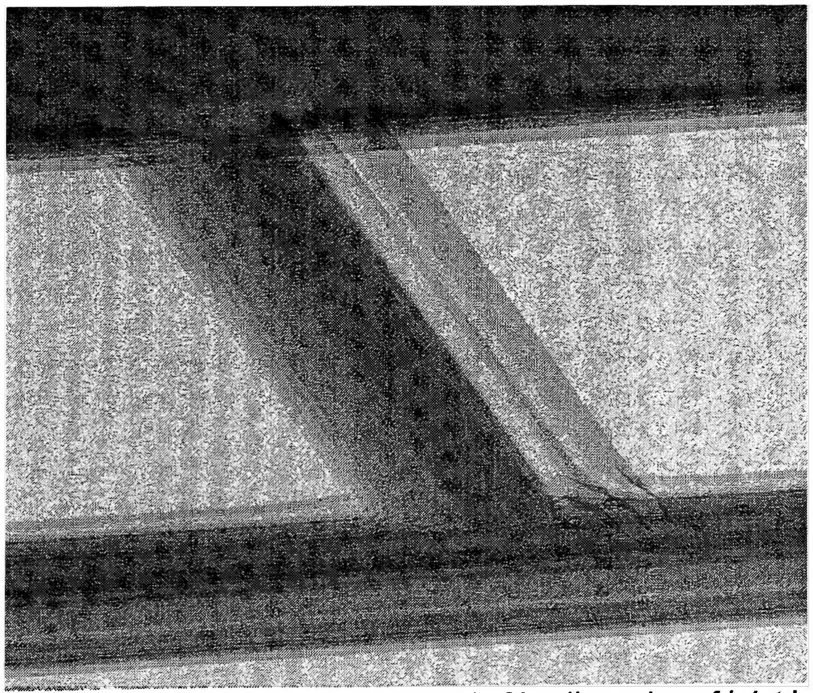


Figure 38: Pressure contours ahead of leading edge of inlet baffle

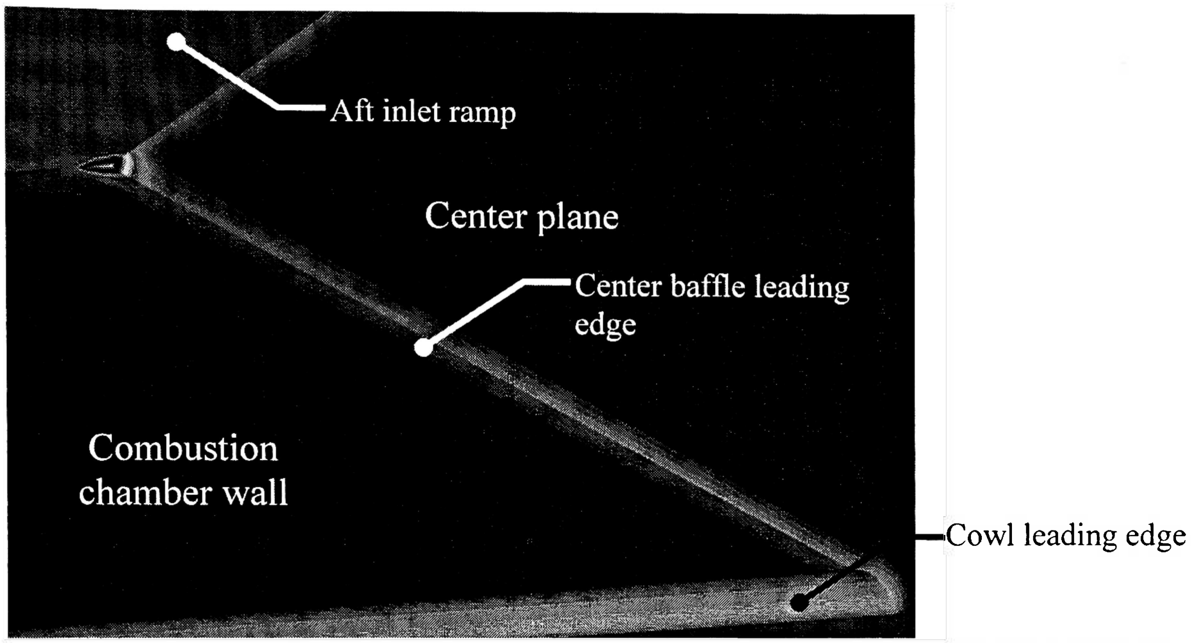


Figure 39: Pressure distribution along surfaces and center plane @ leading edge of center baffle

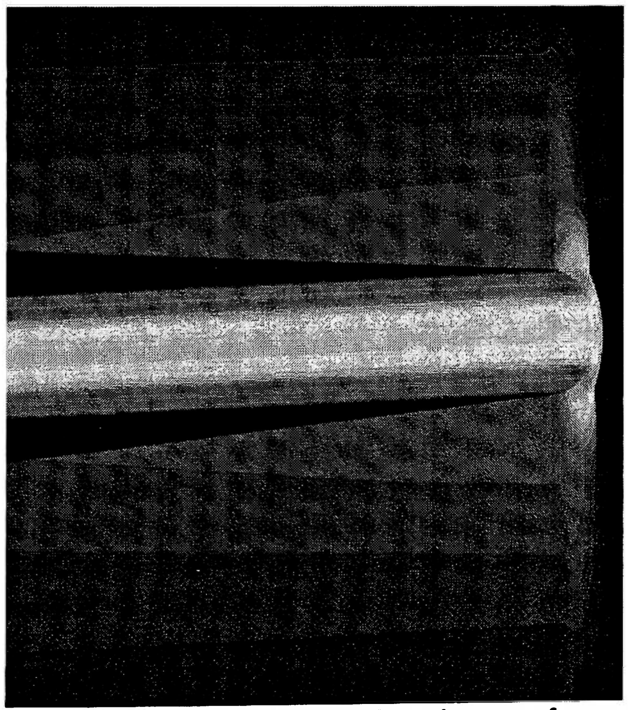


Figure 40: Pressure distribution along surfaces and center plane @ bow leading edge

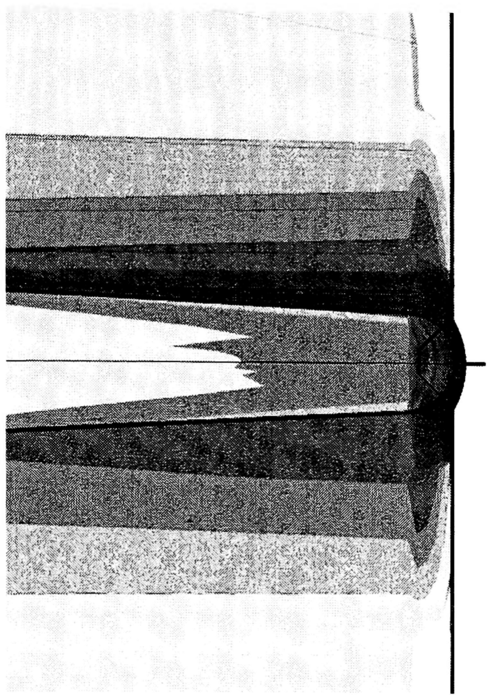


Figure 41: Pressure contours in way of bow leading edge

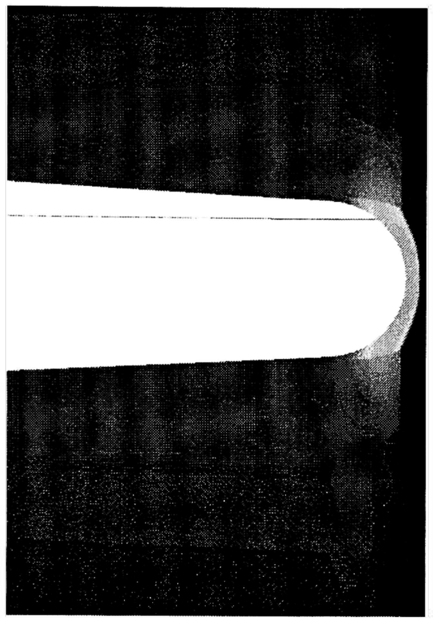


Figure 42: Pressure distribution along center plane iwo bow leading edge

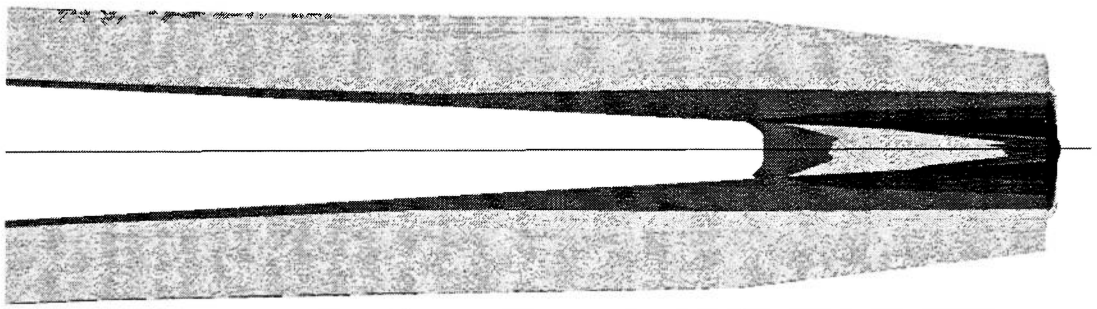


Figure 43: Pressure contours in way of bow – shock to side of bow is visible

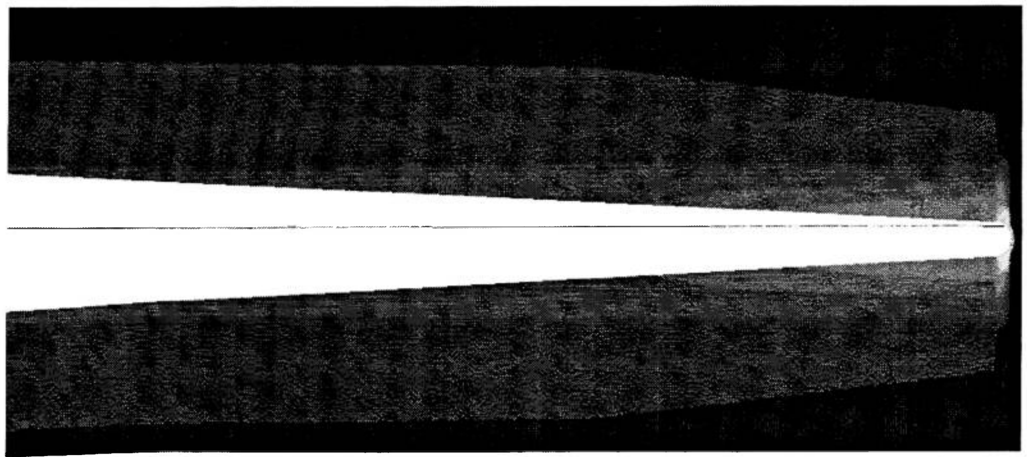


Figure 44: Pressure distribution along center plane in way of bow