COMPUTATION OF HIGH SPEED CHEMICALLY REACTING VISCOUS FLOWS WITH CARTESIAN MESH ON A GPU BASED PARALLEL SYSTEM

A Thesis submitted

in partial fulfillment for the Degree of

Doctor of Philosophy

by

V.ASHOK

Department of Aerospace Engineering

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

MARCH, 2013

COMPUTATION OF HIGH SPEED CHEMICALLY REACTING VISCOUS FLOWS WITH CARTESIAN MESH ON A GPU BASED PARALLEL SYSTEM

A Thesis submitted

in partial fulfillment for the Degree of

Doctor of Philosophy

by

V.ASHOK

Department of Aerospace Engineering

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

MARCH 2013

CERTIFICATE

This is to certify that the thesis entitled **Computation of High Speed Chemically Reacting Viscous Flows with Cartesian Mesh on a GPU Based Parallel System** submitted by **V.Ashok** to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Doctor of Philosophy** is a *bona fide* record of research work carried out by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

Dr. V.Adimurthy
Supervisor
Dean, R&D, IIST

Dr. George Joseph
Co-Supervisor
Prof. Brahmprakash Professor
Vikram Sarabhai Space Centre

Thiruvananthapuram
March, 2013

Counter signature of HOD with seal

DECLARATION

I declare that this thesis entitled **Computation of High Speed Chemically Reacting Viscous Flows with Cartesian Mesh on a GPU Based Parallel System** submitted in partial fulfillment of the degree of **Doctor of Philosophy** is a record of original work carried out by me under the supervision of **Dr. V.Adimurthy and Dr. George Joseph** and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgements have been made wherever the findings of others have been cited.

V.Ashok
SC08D001

Thiruvananthapuram
March-2013

ACKNOWLEDGEMENTS

At the outset, I would like to express my sincere gratitude to Dr.V.Adimurthy, my supervisor at Indian Institute of Space Science and Technology, for playing a very vital role by guidance, constant encouragement, and constructive review of my doctoral work. I had the good fortune of working with him for more than two decades as my superior in the Aerodynamics group of Vikram Sarabhai Space Centre and now as my supervisor at the Indian Institute of Space Science and Technology. His in depth and distilled understanding of the subject with eye for details has gone a long way in enhancing the quality of this work. In spite of his very busy schedule, he always took special interest in the review of my work for which I am greatly indebted.

My sincere gratitude to Dr.George Joseph my co-supervisor at Vikram Sarabhai Space Centre, for giving me constant encouragement, useful reviews and also for regular monitoring of the progress of my work. Next, I would like to express my sincere gratitude to Shri. Thomas.C.Babu, Prof. Brahmprakash Scientist at Vikram Sarabhai Space Centre, my long time teacher in high performance computing, and the chief architect of GPU based SAGA supercomputer which was extensively used for this work. It is indeed a privilege to have been officially associated with him for more than one and a half decades which provided me a good learning opportunity on parallel computing. His deep knowledge in the subject of high performance GPU based computing and programming aspects has really made GPU computing part of this work possible. I also take this opportunity to sincerely thank Dr.B.N.Suresh, founder Director of IIST who gave encouragement and a good opportunity to pursue research work at IIST. My sincere thanks to Professor N.Balakrishnan of IISc Bangalore and Professor T. Sundararajan of IIT Madras who were my doctoral committee members for their constructive review and very useful suggestions. I also sincerely thank Professor Kurien Issac, Head of Department of Aerospace Engineering, IIST and my doctoral committee member for constructive suggestions. My thanks also to Prof. C.S.Narayanamurthy, Head, Department of Physics, IIST for his contribution as doctoral committee member.

This is also the opportunity to thank Dr.K.S.Dasgupta, Director of IIST for enabling me to pursue doctoral work at IIST.

My special thanks to Shri. M.V.Harichand of Aero parallel computing facility for useful discussions and help provided on GPU programming aspects. I thank Shri. Amitkumar Singh for rendering help in post processing of some of the CFD outputs. My sincere appreciation to Dr.S.L.N Desikan for help provided during the thesis preparation.

My special thanks to Dr.T.Jayachandran and Shri. Hemant Jha for very useful discussions on CFD aspects. I also thank, Dr.Dipankar Das, Shri.Lazar Chittilappily and Shri. Gnanasekar for useful discussions related to Scramjet engine simulations. I also take this opportunity to thank Shri. R.Swaminathan, former head Aerodynamics R&D Division, Dr.Pradeep Kumar, former Group Director, Aerodynamics and Aerothermal Group and former Deputy Directors of Aeronautics entity, Shri.S.V.Sharma, Dr.K.Sivan, Dr.S.Swaminathan and the present Deputy Director Shri.S.Pandian whose support I received while pursuing my doctoral program. My special thanks are also to Shri. P.S.Veeraraghavan, former Director VSSC for allowing me to pursue doctoral studies along with my official duties. I wish to place on record my special gratitude to Dr.K.Radhakrishnan, Chairman of ISRO who motivated me to pursue doctoral studies when he was the Director of VSSC.

Since most of the doctoral work had to be done beyond regular office hours, support from my family members was very essential. In this regard, I am deeply indebted to my parents, my wife Uma and my two daughters, Pooja and Nandini. They allowed me to pursue my doctoral work on a regular basis at house without many complaints.

ABSTRACT

The computation of high speed chemically reacting flows during reentry of a vehicle from outer atmosphere and Scramjet propulsion involving high speed turbulent combustion of hydrogen and air are some of the important technologies for low-cost access to space. The solutions to such problems using Cartesian mesh framework have a tremendous advantage in terms of very fast turnaround time from geometry to solution because of completely automated grid generation. However the Cartesian mesh has a very serious limitation in terms of handling the near wall viscous resolution and hence requires some special treatment near the wall. With regard to solving complex flow problems pertaining to Scramjet combustion the turn- around time can be reduced in a very cost effective way through parallel computing with latest high performance computing technology engaging cluster of multi core processors with Graphic Processing Unit called GPU which accelerates the computation. The present work addresses all the above three issues namely, the near wall resolution of hypersonic viscous flows with a Cartesian mesh based system and computations of finite rate chemically reacting turbulent flow in Scramjet engines with Cartesian mesh and performing the Scramjet computations on a GPU based parallel system. Thus the work carried out has three main objectives. The first objective is to obtain the solution of high speed laminar viscous flows both non-reacting and reacting for reentry type problems with a hybrid approach of unstructured prism layer near the wall and Cartesian mesh way from the wall The second objective is to develop a turbulent finite rate chemically reacting code with hydrogen air combustion for Scramjet computations involving complex geometries with Cartesian mesh from an existing perfect gas Cartesian mesh turbulent flow code which uses a wall function approach. The third objective is to develop parallel computing algorithms and necessary code for GPU based parallel computing to perform tip-to-tail simulation for a typical Scramjet vehicle with combustion on a cluster of machines with GPU accelerators.

The Cartesian mesh based viscous laminar flow solution is achieved by creating an unstructured prism layer near the wall by the normal projection of Cartesian mesh panels and stitching with the outer Cartesian mesh and performing a hybrid solution having a combination of unstructured prism layer solution near the wall and Cartesian mesh solution away from the wall. As for the numerical scheme, the inviscid fluxes are computed using Advective Upstream Splitting Method and linear reconstruction of primitive variables with limiter is employed. The viscous fluxes are evaluated from gradients estimated using standard Green-Gauss procedure. The solution is fully explicit and marched using backward Euler time marching mode with local time stepping for convergence acceleration. The developed code is first validated for perfect gas conditions against available experimental results for typical sphere-cone-cylinder-flare geometry at hypersonic Mach number for zero angle of attack. For three dimensional cases with angle of attack, the prism layers extruded from the Cartesian mesh from surface panels which are of 3 sides to 6 sides are not stitched with the outer Cartesian

mesh. Hence in this approach, first an Euler solution is obtained for the Cartesian mesh and this solution is mapped on to the hybrid unstructured prism layer near the wall and the laminar Navier-Stokes solution carried out for the unstructured prism layer alone with the Euler solution data as the boundary condition for the outermost unstructured prism layer. This solution procedure was validated against available experimental heat flux data at angle of attack for hypersonic Mach number. For hypersonic chemically reacting flows the hybrid solution methodology with 7 species finite rate air chemistry model (Park-87) is used. Results of species mass fractions, temperature profiles, wall heat flux and shock stand-off distance from the present code are validated for standard test cases like chemically reacting flow over wedge and Lobb sphere by comparing with the reported results of other CFD code solutions with structured mesh and with limited experimental results.

The solution of turbulent flow in Scramjet engines with finite rate Hydrogen-air chemistry was achieved by developing a code starting from an existing Cartesian mesh perfect gas turbulent code with wall function. 7 species 7 reaction ONERA chemistry model was used to obtain the species production rates from Hydrogen-air reactions. The developed code was validated against available experimental data on pressure and total temperature from ground test results of a typical Scramjet combustor in connected pipe mode conditions. Since the ground test conditions are not the same as flight conditions, numerical experiments were performed to bring out the effects of inlet pressure and vitiation on the Scramjet combustor performance.

The finite-rate chemically reacting flow in Scramjet engines involve highly compute intensive operations on a very large mesh which typically demands use of high performance computing platforms. In this regard, the utility of latest GPU based computing platforms has been explored for such applications. To obtain good performance from GPU accelerators with Cartesian mesh solvers was a real challenge as the rectangular adaptive Cartesian mesh with hanging node is not inherently data parallel. To achieve good parallel computing performance with GPU accelerators, suitable data parallel algorithms as applicable to adaptive Cartesian mesh and good memory management techniques were developed. Data parallelism was achieved by grouping the Cartesian mesh cells into eight different cell groups with each group having almost identical computational flow and group-wise computation is launched in the GPU kernel one after another. Parallel computing performance on cluster of GPU machines and factors affecting the performance are brought out.

Tip-to-tail computation with combustion for a representative Scramjet vehicle with a cone cylinder fore body and two Scramjet engines mounted was carried out with the developed Cartesian mesh solver on a cluster of GPU machines. The performance of the vehicle in terms of pressure, combustion efficiency and thrust was evaluated for a typical flight condition for two air fuel equivalence ratios. The parallel computing performance on the GPU cluster for such a large size problem is also brought out.

In this thesis, Cartesian mesh based solution to laminar hypersonic flows both non-reacting and chemically reacting flow as in re-entry type vehicles and Scramjet engine turbulent flows with Hydrogen-air combustion which are the two critical technologies for low-cost access to space have been addressed Also development of suitable data parallel computing algorithms has been done to enable the use of adaptive

Cartesian mesh solver on a cluster of GPU based machines to reduce the turnaround time for Scramjet engine solution which is very essential for a faster design cycle.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AGARD	Advisory Group for Aerospace Research and Development
AIAA	American Institute of Aeronautics and Astronautics
ALU	Arithmetic Logic Unit
AUSM	Advective Upstream Splitting Method
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy, Time step constraint for numerical scheme
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DRAM	Dynamic Random Access Memory
DRDBD	Distributed Relocated Block Device
ER	Equivalence Ratio
EURANUS	European Aerodynamic Numerical Simulator
FLOPS	Floating Point Operations Per Second
GFLOPS	Giga Floating Point Operations Per Second (10^9 FLOPS)
GPU	Graphic Processing Unit
HIEST	High Enthalpy Shock Tunnel
HPC	High Performance Computing
ISRO	Indian Space Research Organisation
JAXA	Japanese Space Exploration Agency
LAURA	Langley Aerothermodynamic Upwind Relaxation Algorithm
LES	Large Eddy Simulation

LINPACK	Linear Algebra Package
MIMD	Multiple Instruction Multiple Data
MPI	Message Passing Interface
NASCART-GT	Numerical Aerodynamic Simulation via CARTesian Grid Techniques, Solution adaptive Cartesian grid flow solver
NFS	Network File System
OS	Operating System
PetaFLOP	Peta Floating Point Operations Per Second (10^{15} FLOPS)
PSLV	Polar Satellite Launch Vehicle
RAM	Random Access Memory
RANS	Reynolds Averaged Navier-Stokes
SAGA	Supercomputer for Aerospace with GPU Architecture
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Thread
SMP	Symmetric Multi Processor
SRE	Space Recovery Experiment
TFLOPS	Tera Floating Point Operations Per Second (10^{12} FLOPS)
TPS	Thermal Protection System
TSTO	Two Stage To Orbit

NOTATIONS

$\lvert M \rvert$	Absolute value of M
Pr	Prandtl number
Sc	Schmidt number
Le	Lewis number

NOMENCLATURE

a	speed of sound
C_p	coefficient of specific heat at constant pressure
D_i	diffusion coefficient of species i
e	internal energy
E	total energy
F_c	vector of convective fluxes
F_v	vector of viscous fluxes
H	total enthalpy
h	static enthalpy
K	coefficient of thermal conductivity, reaction rate coefficient, parameter in Venkatakrishnan limiter
M	Mach number
MW_i	molecular weight of species i
n_x, n_y, n_z	components of unit normal vector in x, y and z direction
p	static pressure
P	production of turbulent kinetic energy
Pr_T	turbulent Prandtl number
q	heat flux
R	characteristic gas constant, residue

R_U	universal gas constant
dS	surface element
S_x, S_y, S_z	cartesian components of the face vector
T	temperature
t	time, third body coefficients
Δt	time step
u	velocity in x direction
U	vector of conserved variables
v	velocity in y direction
V	contravariant velocity
w	velocity in z direction
$\dot{w}_i$	production rate of species i
W	source vector
X_i	mole fraction of species i
Z_i	mass fraction of species i
α	angle of attack, under-relaxation parameter, stoichiometric coefficient of reactant
β	stoichiometric coefficient of product
κ	Kinetic energy of turbulence
ε	rate of dissipation of turbulent kinetic energy, parameter in limiter
ρ	density
$d\Omega$	volume element

τ_{ij} viscous stress on the i plane in j direction

τ_{ii}^{κ} normal turbulent viscous stress related to κ

τ_{ii}^{ε} normal turbulent viscous stress related to ε

μ_l molecular viscosity

μ_T turbulent viscosity

φ equivalence ratio

ψ limiter, chemical symbol, flow refinement criteria

γ ratio of specific heat

σ CFL number

$\Lambda_C^x, \Lambda_C^y, \Lambda_C^z$ convective spectral radius in x,y and z directions

$\Lambda_v^x, \Lambda_v^y, \Lambda_v^z$ viscous spectral radius in x,y and z directions

Subscripts

∞,inf free stream

diff diffusion

ref reference

$i+\frac{1}{2}$ cell interface

L left

R right

C convective, constant in estimation of reaction rate coefficient

v viscous

fr	forward
br	backward
s ,i	species s ,species i , cell i
r	reaction r
0	Stagnation

superscript

c	convective
p	pressure

Chemical nomenclature

N_2	molecular nitrogen
O_2	molecular oxygen
NO	nitric oxide
O	atomic oxygen
N	atomic nitrogen
NO^+	nitric oxide ion
e	electron
H_2	molecular hydrogen
H_2O	water vapour
OH	hydroxyl
H	atomic hydrogen
CO_2	carbon-dioxide

CHAPTER-1
INTRODUCTION

Low-cost access to space has been an important area of focus for the launch vehicle community. In this regard, recoverability and reusability play a dominant role and most of the space faring nations are exploring newer and better methods to achieve this. Also the urge for the scientific community to know more about other planets and their satellites has led to a number of planetary entry missions and payload capsule recovery experiments. Mars Pathfinder, Huygens Probe, Beagle-2, Stardust, Galileo and Space Recovery Experiment are some of the very exciting missions undertaken by various space agencies. All this has led to a gamut of activities in the area of Aerothermodynamics and considerable progress has been made in experimental as well as computational methods. In the experimental methods, shock tunnels and plasma wind tunnels are mainly used to get data for high speed reentry conditions where chemical reactions associated with high temperature conditions are important. However, shock tunnels and plasma tunnels do not simulate all the important flight parameters like the Mach number, Reynolds number, Damkohler number and stagnation enthalpy conditions. Under such circumstances, the computational fluid dynamics (CFD) becomes a valuable tool wherein the codes are first validated against shock tunnel or plasma wind tunnel results and thereafter used for predicting for the actual flight conditions.

Apart from the aerothermodynamic aspect, low-cost access to space also depends on efficient propulsion systems. In this context, considerable studies are being carried out in the area of advanced propulsion systems like the air-breathing propulsion because of its high specific impulse. This has led to active research in the area of Ramjet and Scramjet engines. In order to arrive at a good design of an air-breathing engine, a number of configurations in the design space need to be evaluated. In this regard, using high fidelity CFD tools in the initial design phase will

be highly beneficial as one can arrive at a good design in the initial stage itself. This would help in avoiding costly configuration changes later in the future which otherwise would have occurred by resorting to low fidelity tools during design phase. Using high fidelity CFD tools for air-breathing engine design would mean solution of turbulent chemically reacting flows with air fuel chemistry for a large number of candidate configurations in a very reasonable time frame. In this regard, it is well known that CFD employing Cartesian mesh has considerable advantage over other types of meshes for the simulation of complex geometries in terms of very less turnaround time for mesh generation. Owing to this, considerable effort is being undertaken by the hypersonic CFD community to solve such finite-rate chemically reacting flows during hypersonic reentry and Scramjet combustion with Cartesian mesh based solvers. Also since such problems demand large computational time due to the solution of large number of equations with chemical reactions on a large mesh size, high performance computing plays a very important role. With the advent of Graphic Processing Unit (GPU) based parallel computing, the CFD solvers are getting adapted for such kind of hardware. Hence the work focusing on chemically reacting flow with air-chemistry and turbulent flow with hydrogen air combustion on Cartesian meshes coupled with high performance computing using cluster of GPU based computing platforms will be very beneficial for the efficient design and analysis of space systems that would be promising candidates for low-cost access to space.

1.1 Reentry Hypersonic Flow

A space capsule that has to be recovered from the orbit has to be deboosted to reenter the earth's atmosphere. During the course of reentry of a space capsule from rarefied atmosphere of about 120 km to the touch down, the capsule encounters free molecular, transitional, and continuum flow regimes. In the case of free molecular flow, the molecules are so far apart that the distance traveled by the molecules before collision with other molecules, called mean free path, is much larger than the

characteristic dimension of the body. This phenomenon is characterized by the Knudsen number which is the ratio of mean free path to the characteristic dimension of the capsule. Flows with Knudsen number greater than 10 are generally treated as free molecular. For Knudsen numbers between 0.01-10 the flow is considered transitional and less than 0.01 the flow can be considered continuum. In the continuum flow regime there are different classes of flows, namely, (i) thermal non-equilibrium and chemical non-equilibrium flow, (ii) thermal equilibrium and chemical non-equilibrium, (iii) thermal equilibrium and chemical equilibrium and (iv) perfect gas flow conditions. Figure 1.1 shows the various flow regimes encountered in the stagnation region of sphere of radius 0.305m as it reenters the earth atmosphere as given by Gupta et al. (1990). The ballistic reentry from the orbit will result in large velocities of the order of thousands of meters per second and such high velocities in the continuum flow regime result in a strong shock in front of the vehicle. Behind the shock there will be sharp rise in the static temperature which can give rise to chemical reactions of the constituent species of air. Molecules behind the shock which will have energy content in translational, rotational, vibrational and electronic modes and will also exchange energy between these various modes. It is to be noted that there are time scales associated with various phenomenon associated with the hypersonic chemically reacting flow over such reentry bodies. First one is the fluid-dynamic time scale which is the time taken for the flow to cross the characteristic length dimension of the reentry body and the second one is the reaction time scale which is the time taken for the chemical reactions to occur. The third time scale is the relaxation time scale which is the time taken for the energy transfer between various internal energy modes namely translational, rotational, vibrational and electronic modes. Depending on the ratio of fluid dynamic time scale to chemical reaction time scale, called as the Damkohler number the high temperature hypersonic flows are classified as Frozen, Chemical Equilibrium and Chemical Non-equilibrium flows which are described below.

a) Frozen flows

If the reaction time scale is much larger than the fluid-dynamic time scale, the flow can be treated as frozen. In this case, the reactions take a large time to complete and by this time the fluid would have already crossed the characteristic length dimension which means that the composition of the fluid does not change in the domain of interest. Hence frozen flows can be treated as non-reacting flows with perfect gas equation of state to describe the thermodynamic state of gas.

b) Chemical Equilibrium flows

If the reaction time scale is much smaller than the fluid-dynamic time scale, the flow is considered to be in chemical equilibrium. In this case, when the gas flows from one point to another point, the local pressure and temperature changes and the reactions are so fast that the fluid reaches the local equilibrium state corresponding to the temperature and pressure at the point. This results in change in composition from point to point and the thermodynamic properties have to be evaluated from the equilibrium composition.

c) Chemical Non-equilibrium

The flow is said to be in chemical non-equilibrium when reaction time scale and fluid-dynamic time scale are of the same order. In this case the flow is not able to reach the local equilibrium value corresponding to the pressure and temperature at each point. In this case, the species continuity equations have to be solved to get the non-equilibrium composition at each point.

Just like the flows are classified based on the Damkohler number associated with chemical reactions, the flows can be also be classified into thermal equilibrium and thermal non-equilibrium based on the ratio of relaxation time scale (time taken for energy transfer between various internal energy modes) to fluid-dynamic time scale also called as Damkohler number associated with thermal non-equilibrium and is given below.

d) Thermal Non-equilibrium

If the relaxation time scale is of the order of the fluid-dynamic time scale, then the flow is said to be in thermal non-equilibrium. The relaxation time scale would

essentially depend on the number of collisions that would take place among the molecules which is a function of density and temperature of the gas. Of the internal energy modes, the translational and rotational modes quickly equilibrate as only very few collisions are needed to reach equilibrium between translational and rotational modes. This is not the case with vibrational modes and hence for thermal non-equilibrium flow, the vibrational energy equation is also to be solved along with the translational energy equation. Hence this will give rise to vibrational temperature associated with vibrational mode of energy apart from the usual translational temperature. In such cases, the energy transfer between vibrational and translational modes (V-T) and vibration vibration modes (V-V) have to be taken into account in the vibrational energy equation which would give rise to a two temperature model. In some cases, the electronic mode of energy would not have equilibrated with other modes and hence electronic energy equation also has to be solved from which the electronic temperature associated with electronic energy can be calculated which will give rise to a three temperature model.

e) Thermal equilibrium

If the relaxation time scale is much smaller than the fluid-dynamic time scale, then the energy transfer between the various internal energy modes would quickly equilibrate before the characteristic flow time and the flow is said to be thermal equilibrium. Such flows are characterized by single temperature model and only one energy equation is solved which corresponds to the translational temperature that has equilibrated with rotational, vibrational and electronic modes.

Figure 1.1 shows the various flow regimes for a particular combination of velocity and altitude occurring during reentry. At the start of the reentry when the altitudes are very high, the flow is in chemical and thermal non-equilibrium. As the altitude decreases, due to increase in density, the flow tends to chemical non-equilibrium but thermal equilibrium. This means that the vibrational relaxation times are very small or in other words the number of collision increases and hence vibrational energy exchanges are quickly equilibrated. As the altitude further reduces

during the reentry, the flow is characterized by thermal and chemical equilibrium. In this case both the vibrational relaxation time and the chemical reaction time are much smaller than the characteristic flow time.

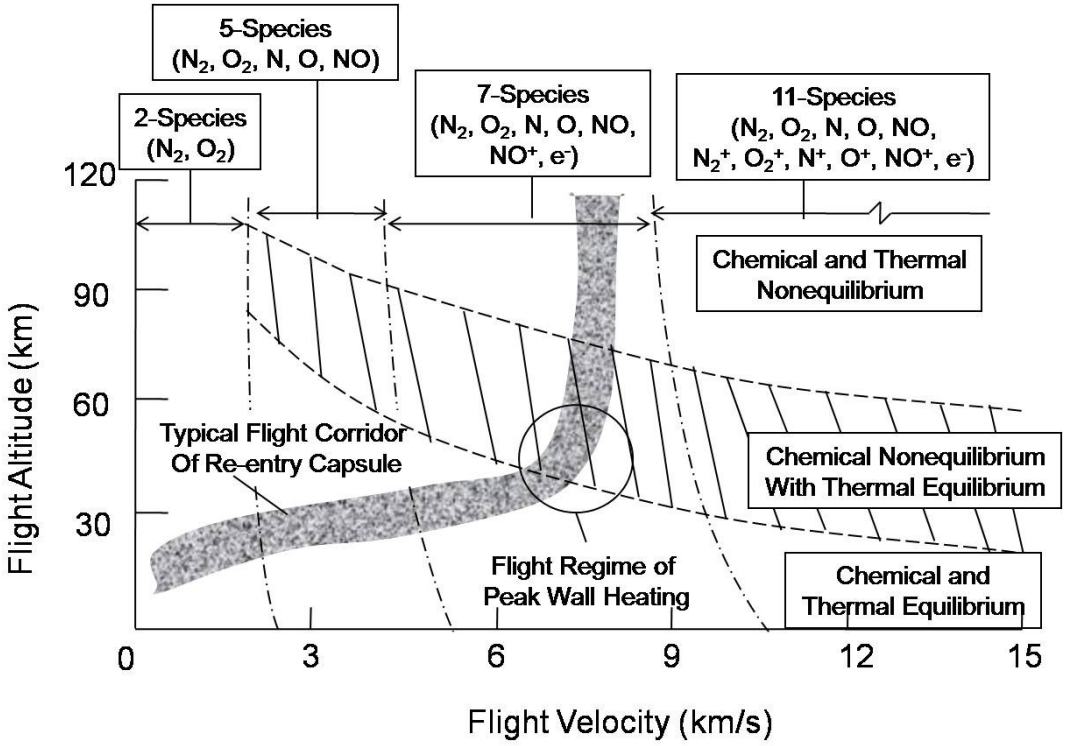


Figure 1.1 Flow regimes encountered at stagnation region of 0.305m radius sphere [Gupta et al. (1990)]

At further lower altitudes when sufficient deceleration of the reentry capsule has already taken place because of the drag, the velocities are not high enough to cause sufficient rise in temperature to cause chemical reactions. Hence the high temperature effects are not present and thus perfect gas computations would be sufficient to compute the flow. All these aspects are shown in Figure 1.1 and also number of species equations to be solved for various altitude velocity combinations is also shown in the figure. It is seen that 7 species air chemistry model is adequate for

typical ballistic reentry vehicles and the peak heating mostly occurs in the regime when the flow is in chemical non-equilibrium and thermal equilibrium.

High temperature effects for air at different temperature conditions under sea level pressure are described by Anderson (1989). At temperatures less than 800 K the gas stays calorifically perfect. Only translational and rotational internal energy modes are fully excited while the excitations of vibration mode are negligible and chemical reactions are not present. In this regime the specific heats are essentially constant and this corresponds to Mach number less than 3 at sea level condition. For temperatures between 800 K and 2000 K, the vibrational mode of energy becomes an important portion along with translational and rotational modes. In this regime the specific heat is a function of temperature and hence the gas is thermally perfect. At temperatures between 2000 K and 2500 K, the vibrational modes are fully excited and the molecular Oxygen starts dissociating. Around 4000 K the molecular Oxygen is completely dissociated and also the molecular Nitrogen starts to dissociate. At 9000 K, the molecular Nitrogen is almost completely dissociated. At 12000 K, all the gases are completely dissociated and sufficient ionization has taken place to have good amount of free charges.

The reentry conditions impose severe thermal load on the recovery module and hence the thermal protection system should be able to take into account of this thermal load. The thermal protection system (TPS) like Carbon Phenolic are of the ablative type which were the ones that was used for the first manned earth reentry missions of Apollo. One of the disadvantages of the ablative type TPS is the shape change due to ablation which would make the aerodynamics not so much predictable resulting in deviations on the touch down point. In the case of non ablative TPS like silica tiles, the shape change due to reentry heating is not present and hence the landing point can be more precisely predicted as compared to ablative type TPS. Heat flux prediction during entire reentry trajectory is one of the important inputs for

TPS design. The total thermal load would decide the thickness of the TPS and the maximum heat flux will decide the type of material for TPS to be used.

The wall heat flux due to hypersonic aerothermodynamics depends on the type of flow regime and the wall characteristics. The high temperature behind the shock at hypersonic flow will trigger chemical reactions of the constituents of air, like Oxygen and Nitrogen dissociation and the extent of dissociation is mainly a function of temperature. The dissociation reactions are essentially endothermic and hence the temperature reduces. However at the wall, the atomic Nitrogen and Oxygen formed due to dissociation can recombine and the heat of recombination is given to the wall. This would depend on the material of the wall which could provide active sites for recombination reaction. The wall which aids the recombination process is termed as catalytic wall. Thus the catalytic wall will have in addition to convective heat flux due to the temperature gradient at the wall, a diffusive heat flux component due to heat of recombination. In the case of a non-catalytic wall the diffusive component of heat flux is absent as the gradient of the species mass fraction at the wall is zero.

Silica tiles with Borosilicate glass coating prevent recombination at the wall and hence are non-catalytic coatings. The above mentioned aspects of catalytic wall are for the chemical non-equilibrium regimes. In the case of equilibrium flow, the reaction time is much shorter than the flow time and hence the recombination process takes place in the boundary layer itself which will deliver the heat of recombination in the boundary layer making the boundary layer hotter. This would result in a large convective heat flux. Usually the equilibrium heat flux and the non equilibrium fully catalytic heat flux are of the same order. Figure 1.2 shows capsule which was part of Space Recovery Experiment (SRE) in which the reentry capsule was put in orbit by the Polar Satellite Launch Vehicle (PSLV) of Indian Space Research Organisation (ISRO) on January 20, 2007 and recovered at Bay of Bengal on January 22, 2007. The nose cap which experienced very high heat flux of the order of 200 W/sq.cm was

made of Carbon Phenolic and the cone and flare portion TPS was made of Silica Tiles with Borosilicate Glass coating to make the wall non-catalytic.

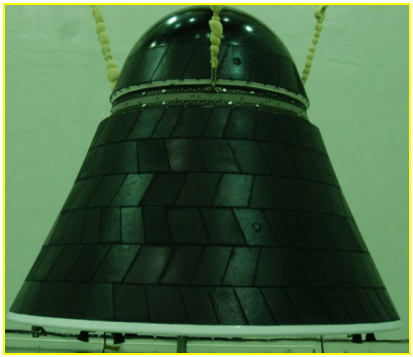


Figure 1.2 Space Capsule of SRE mission (www.ISRO.org)

Another important aerothermodynamic phenomenon associated with reentry is the communication black out. If the post-shock temperatures during reentry are large enough to cause ionization, the free electrons present can severely hamper the communication signals from the spacecraft to ground stations depending on the electron number density of the plasma around the capsule. Hence it is essential to correctly predict the period of communication blackout during reentry so that the data can be stored on board during this period and later transmitted to ground once the blackout period is over.

Shock layer radiation during reentry becomes important when the reentry velocities exceed 10 km/s which would give rise to shock layer temperatures of the order of 20000 K, and is typical of earth reentry like Apollo missions. For the Galileo spacecraft that undertook the planetary entry to Jupiter, the heat flux at stagnation was almost entirely due to radiation as reported by Gnoffo (1999) since the spacecraft entry velocity to Jovian atmosphere was as high as 56 km/s.

1.2 Reacting Flow in Scramjet Engines

The air-breathing engines have a very high specific impulse as compared to the conventional solid, liquid and cryogenic engines and are good candidates for low-cost access to space. This is because, the Oxidizer weight in a rocket motor is more than 70% of the total weight and if this oxygen can be drawn from atmosphere during the atmospheric phase of flight, the total weight would substantially reduce resulting in substantial increase of specific impulse. Figure 1.3 shows the various propulsion system options as a function of Mach number as given by Fry Ronald (2004)

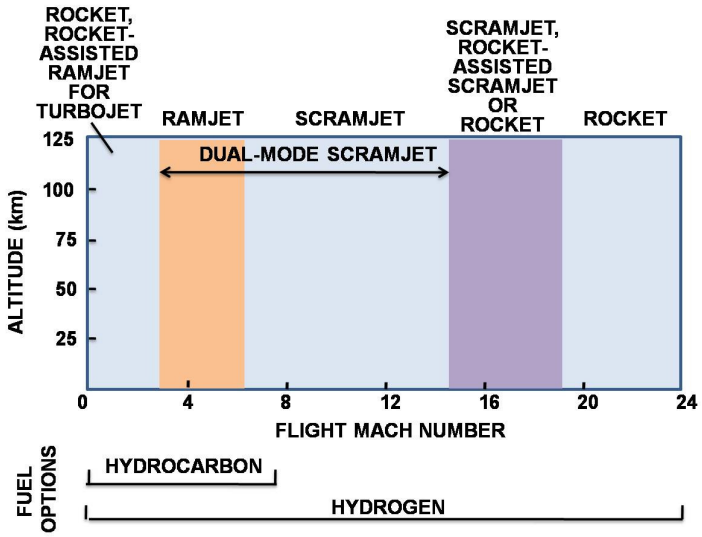


Figure 1.3 Various Propulsion options as a function of Mach number from Fry Ronald(2004)

At lower speeds, the rocket assisted turbojets have to be used and when the Mach number exceeds 3, the air-breathing in the Ramjet mode of operation can be exercised up to Mach number around 6 and beyond Mach number 6 and up to 14 the Scramjet (Supersonic Combustion Ramjet) mode of air-breathing flight is essential. In order to reduce the weight and complexities of having multiple propulsion systems, a dual mode ramjet scramjet is often proposed which will operate in Ramjet mode of operation from Mach number 3 to 6 and in Scramjet mode from 6 to 14. For the space access application, there are many advantages in applying the Scramjet as the propulsion system for the second stage of a Two-Stage-to-Orbit (TSTO),

hydrocarbon fuelled aerospace plane as shown by Townend (2001). The turbo jet and Ramjet mode of flight operation is relatively well known as compared to the Scramjet. Hence lot of research is taking place in this area of Scramjets by first having technology demonstration flights. The successful flight of X-43A described by Voland et al. (2006) and the first flight test of X-51A on 26th May 2010 is a case in point. Figure 1.4 shows the schematic of a typical Scramjet engine.

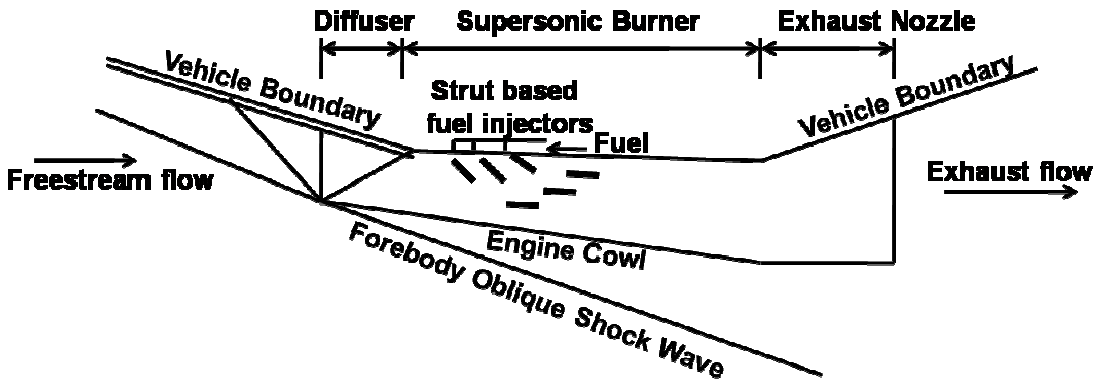


Figure 1.4 Schematic of a typical Scramjet engine

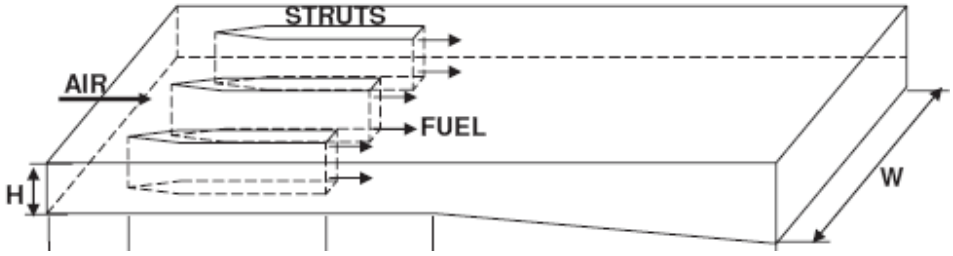


Figure 1.5 Schematic of a typical Scramjet combustor

In the supersonic combustion Ramjet, the ignition of fuel which is either hydrogen or hydrocarbon based has to take place at supersonic speeds. The

combustion and the flame stabilization under supersonic speeds is quite complex and is likened to as “lighting a candle in a hurricane”. The real challenge in the Scramjet engine operation is to burn as much fuel as possible in the combustion chamber without causing the unstart of air-intake. The three key components of the Scramjet engine is the hypersonic air intake, the geometry of the struts for the strut based fuel injection and the nozzle. Additionally the structures have to withstand the extremes of temperature during hypersonic flight combined with additional temperature due to combustion. The intake and the strut together should have as much mass recovery as possible with minimum total pressure loss and at the same time achieve the desired combustion entry Mach number for supersonic combustion. The strut design should be such that, with minimum blockage one should be able to have maximum injection and mixing.

Figure 1.5 shows a typical Scramjet combustor with strut-based injection. The strut helps to introduce stream wise vorticity and thus enhance mixing of fuel with the incoming air. The chemical reactions involving the fuel and the air are very complex processes in the presence of turbulence and involve multi-step reactions. Also sometimes, the turbulence chemistry interaction would further complicate the process. However if the turbulent time scales are much smaller than the reaction time scales then the flow is mixing dominated and the interaction terms would not be dominant. Before Scramjet engines are incorporated in the full-scale flights, the engines are first characterized by conducting technology demonstrator missions in a scale down mode that would give a total picture of the flight performance of the entire system including the individual system performance.

1.3 Survey of Work Done on Hypersonic Flow with Air Chemistry Using Cartesian Mesh

The Cartesian mesh has tremendous advantage in terms of completely automated mesh generation of very complex geometries and since the mesh generation is the most time consuming process for a complex geometry, the

turnaround time from geometry to solution will be much smaller as compared to other CFD solution techniques with structured mesh. However the Cartesian mesh has limitations in handling viscous flows because of its inability to have near wall viscous resolution due to lack of uniformly fine mesh near the wall. A number of research activities exist on Euler equations with Cartesian mesh and are reported by Gaffney et al. (1987), Berger and LeVeque (1989), De. Zeeuw and Powell (1991), Chiang et al. (1992), Epstein et al. (1992), Melton et al. (1995), Yang et al. (1997a,1997b). Many of these researchers have applied it to very complex geometries. Also work has been carried out on Navier-Stokes solution with Cartesian Mesh and reported by Frymier et al. (1988), Corier (1994), Karman Jr (1995), Wang (1996,1998), Xiangying Chen and Zha (2009), Ya'eer Kidron et al. (2010) which are either pure Cartesian Mesh or hybrid mesh with Cartesian mesh away from the wall and Prismatic cells near the wall. There are also other approaches like grid stitching approach, reported by Partha Mondal et al. (2007). In this work, a stretched Cartesian mesh is employed over the streamline bodies like aerofoil and certain Cartesian grid points are moved towards the wall which would avoid small cut cells. Such types of grids are called Cartesian-like grids and have given good solutions to Navier-Stokes equation for relatively low Reynolds numbers. Other Cartesian grid based approach reported by Munikrishna and Balakrishnan (2011) is a meshless approach wherein the mesh points are obtained from the Cartesian mesh and meshless solution of Navier-Stokes equation is carried out. In this procedure, implementing a positive viscous discretisation procedure is shown to be the most crucial part. The solution for viscous turbulent flow solution has also been very successfully obtained with a combination of structured grid near the wall and Cartesian mesh away from the wall with a point cloud in between the two as reported by Katz and Jameson (2009). This work has brought in new concepts like multi- cloud and meshless interface which would compliment the grid based approaches. Another method to solve the Navier-Stokes equation with Cartesian grids is by the immersed boundary method which was originally devised by Peskin (1997) for heart valve modeling using Navier-Stokes equation in two dimensions. In this method, the cells that contain the surface have a body force added to their momentum

equations which represent the reactive force that the body is applying to the fluid in response to the fluid pressure and shear stress and which will eventually make the velocity of the fluid at the wall equal to zero. Subsequently, some of the work with this method is reported by Georgi Kalitzin and Iaccarino (2003), Rajat Mittal and Iaccarino (2005), M.D.Tullio et al. (2007). Computations of turbulent viscous flows have also been carried out by Jae-doo Lee (2006) with Cartesian grids with immersed boundary approach with ghost cell boundary conditions so as to increase the accuracy and minimize the unrealistic fluctuation of flow properties. The turbulence model in this work used was the standard $\kappa - \varepsilon$ model of Launder and Spalding with a new wall function approach for unstructured Cartesian grid solver. There is also yet another modified wall function approach applied to a Rectangular Adaptive Cartesian grid solver for turbulent viscous flows reported by Hagemann et al. (1996) and successfully applied to plug nozzle cluster problems. However all the above methods namely the hybrid mesh methods, pure Cartesian mesh, immersed boundary methods, hybrid Cartesian and meshless methods and Cartesian mesh methods with suitable wall function have so far not been extended to hypersonic flows and in particular to reentry type flows so as to obtain near wall quantities like heat flux. As for the work on chemically reacting hypersonic flows, considerable work is reported in literature with structured mesh and the work reported by Candler (1989,1991), Gnoffo (1989), Alavilli (1997), and Ghislain et al. (2008) are a few of them. However, very little work is reported on computation of chemically reacting hypersonic flow with Cartesian mesh.

The work on chemically reacting hypersonic flow with Cartesian mesh was possibly first reported by Shuang-Zhang Tu and Ruffin (2002). In this work, the inviscid flow with thermochemical non-equilibrium was carried out for certain 2D geometries with rectangular adaptive Cartesian mesh. Subsequently, using an existing 3-D Cartesian grid based solver (NASCART –GT), Jin Wook Lee (2007) and Jin Wook Lee et al. (2010) extended it to handle 3-D inviscid flows with thermal and chemical non-equilibrium with grid adaptation for complex geometries and also

with the capability to compute on a large cluster of machines in a parallel mode using a Space-Filling-Curve (SFC) based domain decomposition technique. This analysis was applied to non-equilibrium hypersonic flow analysis of Ballutes. In this work, since the computations were inviscid, the important quantities like convective heat flux on the wall due to high speed chemically reacting flow had to be done through approximate method of coupling inviscid analysis with an integral boundary layer method. Based on the survey of work done it is found that chemically reacting hypersonic viscous flow computations with a Cartesian mesh based approach is still a topic to be explored so as to obtain near wall quantities like convective heat flux.

1.4 Survey of Work Done on Scramjet Turbulent Flow Computation with Combustion Using Cartesian Mesh

The Scramjet technology being one of the forefront technologies in the area of air-breathing propulsion, considerable amount of work has taken place during the past two decades. Curran and Murthy (2000) describe the design and technical challenges encountered in Scramjet propulsion. Considerable amount of work with structured and unstructured mesh is reported in literature for computational simulation of Scramjet components like intake, combustion chamber and nozzle. Work on air intake has been reported by Tani.K et al. (2001,2006), Krause et al. (2006), Evgeny et al. (2008), Krause and Ballmann (2007). Extensive work has been reported by Gerlinger et al. (1994,1998,2001,2005,2008,2010) which are on supersonic mixing and combustion, strut based injection and mixing, implicit multigrid method to compute turbulent combustion, studies on lobe strut injectors to improve mixing by generating streamwise vortices and factors affecting supersonic combustion in terms of reaction rate mechanism, grid spacing and initial conditions. Gerlinger (2012) has also very recently brought out a low diffusion version of multi-dimensional limiting process (MLP) of Kim et al. (2005) and applied it to turbulent combustion process which proved to be an efficient and simple method to extend conventional second order schemes to higher accuracies and also improve convergence which is especially

attractive for unsteady flows. Mixing and combustion has also been reported by Jeong-Yeol Choi et al. (2005), Sebastian Karl et al. (2006), Rainer et al. (2010), Michael Emory et al. (2011), Soumyajit Saha and Debasis Chakraborty (2011) , Huang Wei et al. (2011) , Rahul Ingle and Debasis Chakraborty (2012). Kindler et al. (2011) reports on TASCOM3D which is a scientific code capable of performing wide range of chemically reacting flows coupled with high performance computing. In the case of Scramjet engine, the tip to tail simulation is more meaningful as the coupling effect of intake and combustor like intake un-start if any due to pressure rise from combustion will come out of the simulation. Some work is also reported with tip-to-tail simulation of Scramjet engine with intake, combustor and nozzle wherein the performance of the Scramjet engine in terms of thrust produced could be obtained. Kodera et al. (2003) has reported one such simulation wherein the entire Scramjet engine from intake to nozzle was simulated with unstructured mesh on the Japanese supercomputer and the simulations were compared with the experimental results obtained from HIEST shock tunnel. Recently Gaitonde et al. (2009) showed the integrated analysis of Scramjet flow path with three intake configurations by employing high performance computing. The three types of inlets considered in this work were of traditional rectangular cross-section configuration and two were of streamlined shaped inlets. The combustor that followed the inlet was a cavity based flame holding combustor. The simulations were performed to analyze the important parameters like inlet distortion, fuel air mixing, ignition and thrust generation at free stream Mach numbers between 6 and 8. Other aspects investigated in this paper are the effect of fidelity in chemistry models from frozen to finite-rate chemistry models of increasing complexity and also fidelity in turbulence closure models from Reynolds averaged to Large Eddy Simulation models (LES). It was found that small scales resolved with superior LES method were essential to understand the shock dynamics and ignition delay time. All the above computational work on tip-to-tail simulations are from structured or conventional unstructured mesh. Considering the advantages of Cartesian mesh for completely automated grid generation from CAD model; evolving a suitable design of a Scramjet engine mounted on a test vehicle for

a desired performance with a number of simulations by varying the design parameters of intake, combustor and nozzle would be very fast with latest high performance computing platforms. Most of the high performance computations reported in literature for such large scale problems are mainly the conventional CPU based computing. With the advent of latest hardware like GPU accelerators, the computational cost for the same computing power as CPU would be substantially lower in terms of lesser hardware cost and lower power consumption. Based on the literature survey carried out, it is noted that such tip-to-tail simulations of Scramjet engine with Cartesian mesh and using the latest hardware platforms like cluster of CPU compute nodes with GPU accelerators is not very much explored and hence needs more attention.

1.5 Survey of work done in CFD with GPU Computing

The Graphic Processing Unit (GPU) is fast becoming a very cost effective tool for high performance computing and its application is now encompassing wide range of applications including CFD. Recently there has been a spurt of activity on the use of GPU accelerators with CFD and are reported by many researchers. Julien Thibault and Inanc Senocak (2009) give the implementation of Navier-Stokes equation for incompressible flow using desktop machines with multi GPU's and using NVIDIA's CUDA (Compute Unified Device Architecture) programming model. A speed up of 21X (21 times) was obtained when simulations were done on Intel Core 2 Duo 3 GHz processor with 2 GPU (NVIDIA S870 Tesla) accelerators as compared to single core 2.4 GHz AMD Opteron processor. Everett et al. (2010) describes the work carried out to simulate unsteady turbulent flow on a cluster of GPU's with double precision accuracy for the multi block turbulent flow solver. High performance could be obtained by optimizing the data layout on the GPU and optimizing data transfers and also using asynchronous memory copies so as to overlap GPU execution with communication. A speed up of 70X was quoted for a cluster of 8 Tesla 2050 "Fermi" GPU's as against the serial solver. Dana and Inanc Senocak (2011) present the dual

level parallel implementation of the Navier-Stokes equation to simulate buoyancy driven incompressible flows with parallel geometric Multigrid solver. Here CUDA programming model is used for fine-grain data parallel operations within each GPU, and MPI for coarse-grain parallelization across the cluster. Rey DeLeon and Inanc Senocak (2012) have demonstrated the use of GPU for the computation of incompressible turbulent channel flow with Large Eddy Simulation. Hai P Le and Cambier (2012) report on the implementation of reacting gas solver with detailed chemical kinetics on a GPU. A speed up factor of 40 was obtained for a 9 species gas mixture with 38 reactions. Based on the survey of work done on GPU applications to HPC in CFD, it is seen that solution of large scale flow problems on a cluster of GPU based machines using Rectangular Adaptive Cartesian Mesh and with Combustion is an area that needs to be explored. In an industrial perspective, such type of High Performance Computing with GPU accelerators for solving Scramjet type of flow with combustion using adaptive Cartesian mesh will be of great benefit in significantly reducing turnaround time for solution. This will enable the designers to explore a large number of candidate configurations so as to explore larger area in the design space to arrive at a near optimal design of a Scramjet engine configuration.

1.6 Motivation and Research Objectives of the Present Study

The computation of high speed chemically reacting flows during reentry of a vehicle from outer atmosphere and Scramjet propulsion involving high speed turbulent combustion of hydrogen are some of the important technologies for low-cost access to space. Because of completely automated grid generation, solutions of such problems using Cartesian mesh framework has a tremendous advantage in terms of very fast turnaround time from geometry to solution particularly when the configurations of interest are complex. Also the turnaround time can be further reduced in a very cost effective way through parallel computing. In recent times, the use of GPU accelerated multi cores, have further enhanced the utility of such parallel codes, both from view point of computational cost and speed up.

However the Cartesian mesh has a limitation in terms of handling the near wall viscous resolution and hence requires some special treatment near the wall to obtain wall quantities like heat flux. Often a near-wall resolution realizable from the use of suitable wall functions is considered adequate from the view point of obtaining solutions in a typical industrial framework. Nevertheless, there can be instances where resolving the wall layer adequately can be important for the problem. In this context, we have explored the feasibility of Cartesian mesh based solutions for laminar hypersonic flows and also reentry hypersonic chemically reacting flows as applicable to recovery modules.

Simulating a Scramjet engine flow with all its geometric complexities can be a challenge to the conventional CFD tools, but becomes an ideal candidate for Cartesian mesh based computations. This problem further becomes compute intensive because of the need to simulate Hydrogen-air combustion with finite-rate chemistry. It is noted that this complex problem has not been addressed so far within a Cartesian grid framework. The fact that good estimates of pressure distribution with turbulent combustion can very well be obtained with approximate wall functions suggest the use of full Cartesian mesh to resolve the geometry and flow. To achieve this, the existing Cartesian mesh based perfect gas turbulent flow code which has already been applied to many problems and reported by Ashok and Babu (1999), Chakraborty et al. (2003), Manokaran et al. (2003), and Singh et al. (2009) can be extended to handle finite-rate chemically reacting flow of Hydrogen-air combustion. This is one of the aims of the present work.

The computations involving finite-rate chemistry of Hydrogen-air combustion on three dimensional complex geometries inherently are compute intensive with large solution turnaround times. The use of High Performance Computing (HPC) capability becomes imperative in any meaningful use of this strategy in a typical design setting. Therefore, in this context, the HPC capability offered by GPU accelerators with substantial cost and power consumption advantages is a very natural choice for

effecting such computations. But the challenge would be to implement GPU based parallelism in a Cartesian grid based solver, typically allowing hanging nodes. Based on the survey of work carried out on GPU computing, it is found that such type of very large scale computations of tip-to-tail Scramjet simulations with finite-rate combustion using an adaptive Cartesian mesh on a GPU based parallel system, is an area that needs to be explored. Hence the present work has the following three objectives

a) Obtain the solution of non-reacting and finite-rate chemically reacting laminar hypersonic flow for reentry type problems, with a hybrid Cartesian approach involving unstructured prism layer solution for near wall resolution and Cartesian mesh solution in the outer region.
b) Develop a turbulent finite-rate chemically reacting code for Hydrogen-air combustion for Scramjet computations from existing Cartesian mesh perfect gas turbulent flow solver having a wall function approach.
c) Develop parallel computing algorithms for Cartesian grid solver exploiting GPU cores for chemically reacting turbulent flows and perform tip-to-tail simulations for a typical Scramjet vehicle with an objective of predicting its thrust.

1.7 Outline of the Thesis

Introduction and objectives of the present study are described in this first chapter. The second chapter describes the formulation of the chemical non-equilibrium flow with description about the transport properties and chemical reactions considered for air chemistry as well as Hydrogen-air chemistry. The third chapter deals with the numerical method used with details about the reconstruction scheme, inviscid flux computation, viscous flux computation, local time stepping, point implicit scheme and species under-relaxation of source terms and the time marching method. Fourth chapter describes the hybrid solution methodology with details about generation of prism layer from background Cartesian mesh panels and

validation cases for near-wall viscous resolution for hypersonic flow so as to obtain heat flux for non-reacting and reacting flows with Cartesian mesh. The chapter also describes the computation and validation of flows with Hydrogen-air combustion in Scramjet engines involving complex geometries with pure Cartesian mesh using a wall function approach. The effect of various flow parameters like vitiation and inlet pressure on the combustor performance is also brought out in this chapter. Fifth chapter deals with parallel computation with GPU accelerators wherein parallel computing algorithms developed for GPU computations for Cartesian mesh solvers with hanging node are described. The computational load distribution between CPU and GPU and type of cells that are allotted to CPU and GPU in the order of priority, communication methodology and parallel computing performance on a cluster of GPU based machines are given in this chapter. The sixth chapter describes the tip-to-tail simulation of a typical Scramjet vehicle with external flow and internal flow with combustion in Scramjet engine attached to the vehicle. Also the performance in terms of combustion efficiency and axial force due to combustion is highlighted. The parallel computing performance of this large scale combustion problem on a cluster of GPU machines is also brought out. Chapter seven gives important conclusions and future work needed for further improvements.

CHAPTER-2
FORMULATION FOR CHEMICAL NON-EQUILIBRIUM

During reentry of a spacecraft or a reusable launch vehicle, the high temperature effects start dominating which requires consideration of real or high temperature gas effects. The high temperature effects include chemical reactions and excitation of vibration and electronic modes. If the chemical reaction time and characteristic flow time are comparable then the flow is said to be in chemical non-equilibrium. In this chapter, the formulation for earth atmospheric reentry flows with chemical non-equilibrium and thermal equilibrium as well as finite-rate turbulent chemically reacting non-equilibrium flow of Hydrogen and air will be considered. The maximum heat flux during reentry flows for ballistic reentry normally occurs for flows in chemical non-equilibrium and thermal equilibrium as shown in Figure 1.1 and hence is a design driver case. Taking this into consideration, the flows considered for the studies are in thermal equilibrium and hence the single temperature model is used which means that the temperature term that appears in the energy equation is essentially the translation temperature that is in equilibrium with the vibrational and electronic temperature. In this formulation, the species conservation equations are to be solved for each of the constituent species of air. Typically in a 7-species model of air, N_2, O_2, NO, O, N, NO^+, and e species are considered. The production rates of species are governed by finite-rate chemistry models. In the case of Hydrogen-air combustion in Scramjets, the species considered in the formulation are H_2, O_2, H_2O, OH, H, O, and N_2. For the species conservation equations while the molecular diffusion due to concentration gradient which is the dominant part is modeled, the diffusion due to pressure and temperature gradients is neglected

2.1 Laminar Hypersonic Flow with Air Chemistry

The Navier-Stokes equation for the solution of laminar finite-rate chemically reacting flow of air is as given below in Equation (2.1).

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega + \oint (F_c - F_v) dS = \int_{\Omega} W d\Omega \tag{2.1}$$

Where U is vector of conserved variables and F_c and F_v are vector of convective and viscous fluxes and W is the source vector as given below.

$W = (0,0,0,0,0,\dot{w}_{N2},\dot{w}_{O2},\dot{w}_{NO},\dot{w}_{N},\dot{w}_{O},\dot{w}_{NO^+},\dot{w}_{e})$ are the vector of species production rates .

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \rho Z_{N2} \\ \rho Z_{O2} \\ \rho Z_{NO} \\ \rho Z_{N} \\ \rho Z_{O} \\ \rho Z_{NO^+} \\ \rho Z_{e} \end{bmatrix} \quad F_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V \\ \rho Z_{N_2} V \\ \rho Z_{O_2} V \\ \rho Z_{NO} V \\ \rho Z_{N} V \\ \rho Z_{O} V \\ \rho Z_{NO^+} V \\ \rho Z_{e} V \end{bmatrix} \quad F_v = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\theta_x + n_y\theta_y + n_z\theta_z + \sum_i \left(\rho_i h_i V_{diff\_i}\right) \\ \rho Z_{N_2} V_{diff\_N_2} \\ \rho Z_{O_2} V_{diff\_O_2} \\ \rho Z_{NO} V_{diff\_NO} \\ \rho Z_{N} V_{diff\_N} \\ \rho Z_{O} V_{diff\_O} \\ \rho Z_{NO^+} V_{diff\_NO^+} \\ \rho Z_{e} V_{diff\_e} \end{bmatrix} \tag{2.2}$$

Where Z_i is the mass fraction of species i and V is the contravariant velocity at the face of the cell defined as the scalar product of velocity vector and the unit normal vector at the surface. This can be expressed by Cartesian velocity components

u, v, w in x, y and z directions and components of the outward normal of the cell face which are n_x, n_y, n_z.

$$V = un_x + vn_y + wn_z \tag{2.3}$$

$$E = \sum_{i=1}^{N_s} Z_i e_i + (u^2 + v^2 + w^2)/2 \tag{2.4}$$

is the total energy per unit mass and e_i is the internal energy of species i

and total internal energy $e = \sum_{i=1}^{N_s} Z_i e_i$ (2.5)

And V_{diff_i} = Contravariant diffusion velocity of ith species at the face of the cell defined as the scalar product of diffusion velocity vector and the unit normal vector to the surface of a face and is given by

$$V_{diff\_i} = u_{diff\_i} n_x + v_{diff\_i} n_y + w_{diff\_i} n_z \tag{2.6}$$

Where $u_{diff\_i} = \frac{\rho}{\rho_i} D_i \frac{\partial X_i}{\partial x}$ where D_i, X_i and ρ_i are the diffusion coefficient, mole fraction and density of ith species respectively

$$H = e + \frac{u^2 + v^2 + w^2}{2} + \frac{p}{\rho} \tag{2.7}$$

$$\theta_x = u\tau_{xx} + v\tau_{xy} + w\tau_{xz} + K\frac{\partial T}{\partial x} \tag{2.8}$$

$$\theta_y = u\tau_{yx} + v\tau_{yy} + w\tau_{yz} + K\frac{\partial T}{\partial y} \tag{2.9}$$

$$\theta_z = u\tau_{zx} + v\tau_{zy} + w\tau_{zz} + K\frac{\partial T}{\partial z} \tag{2.10}$$

$$\tau_{xx} = 2\mu\left(\frac{\partial u}{\partial x} - \frac{1}{3}(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)\right) \tag{2.11}$$

$$\tau_{yy} = 2\mu\left(\frac{\partial v}{\partial y} - \frac{1}{3}(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)\right) \tag{2.12}$$

$$\tau_{zz} = 2\mu\left(\frac{\partial w}{\partial z} - \frac{1}{3}(\partial u/\partial x + \partial v/\partial y + \partial w/\partial z)\right) \tag{2.13}$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{2.14}$$

$$\tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \tag{2.15}$$

$$\tau_{yz} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{2.16}$$

2.2 Turbulent Flow with Hydrogen-air Combustion

The Navier-Stokes equation for the solution of finite-rate chemically reacting flow of Hydrogen and air is as given below in Equation (2.17)

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega + \oint (F_c - F_v) dS = \int_{\Omega} W d\Omega \tag{2.17}$$

Where U is vector of conserved variables and F_c and F_v are vector of convective and viscous fluxes and W is the source vector as given below.

$$U = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \\ \rho \kappa \\ \rho \varepsilon \\ \rho Z_{H2} \\ \rho Z_{O2} \\ \rho Z_{H_2O} \\ \rho Z_{OH} \\ \rho Z_{H} \\ \rho Z_{O} \\ \rho Z_{N_2} \\ \rho Z_{CO_2} \end{bmatrix} \quad F_c = \begin{bmatrix} \rho V \\ \rho u V + n_x p \\ \rho v V + n_y p \\ \rho w V + n_z p \\ \rho H V \\ \rho \kappa V \\ \rho \varepsilon V \\ \rho Z_{H_2} V \\ \rho Z_{O_2} V \\ \rho Z_{H_2O} V \\ \rho Z_{OH} V \\ \rho Z_{H} V \\ \rho Z_{O} V \\ \rho Z_{N_2} V \\ \rho Z_{CO_2} V \end{bmatrix} \quad F_v = \begin{bmatrix} 0 \\ n_x\tau_{xx} + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\tau_{yy} + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\tau_{zz} \\ n_x\theta_x + n_y\theta_y + n_z\theta_z + \sum_i \left(\rho_i h_i V_{diff\_i}\right) \\ n_x\tau^{\kappa}_{xx} + n_y\tau^{\kappa}_{yy} + n_z\tau^{\kappa}_{zz} \\ n_x\tau^{\varepsilon}_{xx} + n_y\tau^{\varepsilon}_{yy} + n_z\tau^{\varepsilon}_{zz} \\ \rho Z_{H_2} V_{diff\_H_2} \\ \rho Z_{O_2} V_{diff\_O_2} \\ \rho Z_{H_2O} V_{diff\_H_2O} \\ \rho Z_{OH} V_{diff\_OH} \\ \rho Z_{H} V_{diff\_H} \\ \rho Z_{O} V_{diff\_O} \\ \rho Z_{N_2} V_{diff\_N_2} \\ \rho Z_{CO_2} V_{diff\_CO_2} \end{bmatrix} \tag{2.18}$$

$W = (0,0,0,0,0,S_K, S_\varepsilon, \dot{w}_{H_2}, \dot{w}_{O_2}, \dot{w}_{H_2O}, \dot{w}_{OH}, \dot{w}_H, \dot{w}_O, \dot{w}_{N_2}, \dot{w}_{CO_2})^T$ is the vector of species production rates. CO_2, is also included as a species to take into account vitiated air used in the ground testing of Scramjet combustor.

n_x, n_z, n_z are the components of outward normal of each face of the cell and V is the contravariant velocity which is the velocity normal to each face given by

$V = un_x + vn_y + wn_z$ as given in Equation (2.3)

$$\tau_{xx}^{\kappa} = \left(\mu_l + \frac{\mu_T}{\sigma_K}\right)\frac{\partial \kappa}{\partial x}, \quad \tau_{yy}^{\kappa} = \left(\mu_l + \frac{\mu_T}{\sigma_K}\right)\frac{\partial \kappa}{\partial y}, \ \tau_{zz}^{\kappa} = \left(\mu_l + \frac{\mu_T}{\sigma_K}\right)\frac{\partial \kappa}{\partial z} \tag{2.19}$$

$$\tau_{xx}^{\varepsilon} = \left(\mu_l + \frac{\mu_T}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x}, \quad \tau_{yy}^{\varepsilon} = \left(\mu_l + \frac{\mu_T}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial y}, \ \tau_{zz}^{\varepsilon} = \left(\mu_l + \frac{\mu_T}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial z} \tag{2.20}$$

The production rates of turbulent kinetic energy κ and turbulent dissipation ε is given by

$$\begin{bmatrix} S_\kappa \\ S_\varepsilon \end{bmatrix} = \begin{bmatrix} P - \rho\varepsilon \\ \left(C_{\varepsilon 1}P - C_{\varepsilon 2}\rho\varepsilon\right)\dfrac{\varepsilon}{\kappa} \end{bmatrix} \tag{2.21}$$

The production of turbulent kinetic energy P is given by

$$P = \tau_{xx}^T \frac{\partial u}{\partial x} + \tau_{yy}^T \frac{\partial v}{\partial y} + \tau_{zz}^T \frac{\partial w}{\partial z} + \tau_{xy}^T \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + \tau_{xz}^T \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) + \tau_{yz}^T \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \tag{2.22}$$

The turbulent viscosity $\mu_T = \dfrac{C_\mu \rho \kappa^2}{\varepsilon}$ (2.23)

The closure coefficients used in $\kappa - \varepsilon$ equations and turbulent Prandtl number are as given below by Launder and Sharma (1974)

$C_\mu = 0.09$, $C_{\varepsilon 1} = 1.44$, $C_{\varepsilon 2} = 1.92$, $\sigma_K = 1.0$, $\sigma_\varepsilon = 1.3$, $\mathrm{Pr}_T = 0.9$

2.3 Thermodynamic Model and Transport Properties for Hypersonic Chemically Reacting Air

Internal energy of mixture of gases is the sum of internal energy of the individual species and is given by

$$e = \sum_{i=1}^{N_s} Z_i e_i \tag{2.24}$$

Similarly enthalpy of the mixture of gases is obtained as the sum of enthalpies of individual species

$$h = \sum_{i=1}^{N_s} Z_i h_i \tag{2.25}$$

The values for the enthalpy of individual species is given by polynomial curve fits from Moss (1974)

$$\frac{h_i}{R_U T} = a_1 + \frac{a_2 T}{2} + \frac{a_3 T^2}{3} + \frac{a_4 T^3}{4} + \frac{a_5 T^4}{5} + \frac{a_6}{T} \tag{2.26}$$

The value of a_1 to a_6 is given in the table below for various species used is given below in Table 2.1 and Table 2.2.

Table 2.1 Enthalpy curve fit coefficients for species used in air chemistry model for temperature range 150 K to 7250 K from Moss (1974)

Species	Coefficients					
	a_1	a_2	a_3	a_4	a_5	a_6
N	0.247732e1	0.82901e-4	-76069e-7	0.22462e-10	-0.15404e-14	0.56133e5
NO	0.31968e1	0.11769e-2	-0.38778e-6	0.55731e-10	-0.28806e-14	0.98652e4
N_2	0.32066e1	0.96095e-3	-0.26764e-6	0.33488e-10	-0.15440e-14	-0.99993e3
O	0.26848e1	-0.24358e-3	0.96464e-7	-0.13271e-10	0.65179e-15	0.29177e5
O_2	0.32206e1	0.13129e-2	-0.46651e-6	0.70960e-10	-0.38179e-14	-0.10143e4
NO^+	0.32070e1	0.95464e-3	-0.26312e-6	0.32660e-10	-0.14946e-14	0.11809e6
e^-	0.2500e1	0.0	0.0	0.0	0.0	-0.74537e3

Table 2.2 Enthalpy curve fit coefficients for species used in air chemistry model for temperature range 7250 K to 40000 K from Moss (1974)

Species	Coefficients					
	a_1	a_2	a_3	a_4	a_5	a_6
N	0.29580e1	0.72397e-4	-19304e-8	0.65165e-14	-0.85711e-21	0.52845e5
NO	0.44678e1	0.22696e-4	-0.14025e-9	0.16592e-14	0.63494e-22	0.86956e4
N_2	0.44517e1	0.21996e-4	-0.20502e-9	0.24036e-14	0.82543e-22	-0.23784e4
O	0.25750e1	0.22027e-4	0.50533e-10	-0.11758e-13	-0.13901e-20	0.28692e5
O_2	0.44763e1	0.33173e-4	-0.11257e-9	0.13687e-14	0.68543e-22	-0.20125e4
NO^+	0.44517e1	0.23791e-4	-0.20057e-9	0.23327e-14	0.73176e-22	0.11670e6
e^-	0.2500e1	0.0	0.0	0.0	0.0	-0.74537e3

Non uniform spatial distribution of velocity, temperature and species concentrations cause the molecule transport and in macroscopic dimensions these are well known phenomenon like viscosity, heat conduction, and diffusion.

The coefficient of viscosity for individual species is given by Blottner (1971)

$$\mu_i = e^{C} T^{(A \ln T + B)} / 10 \quad \text{(kg/m-s)} \tag{2.27}$$

For temperatures less than 1000 K i.e. temperatures below dissociation temperatures, the values of A, B, C are as follows

A=-0.1045186, B=1.9790489, C=-16.48024

For temperatures from 1000 K to 30000 K, the constants to estimate the coefficients of each of the species for Air-chemistry is given by Blottner (1971) and is showed in Table 2.3.

Table 2.3 Constants for calculation of species viscosity from Blottner (1971)

Species	A	B	C
N_2	0.0268142	0.3177836	-11.3155513
O_2	0.0440290	-0.0826158	-9.2019475
NO	0.0436378	-0.0335511	-9.5767430
N	0.0203144	-0.0826158	-11.6031403
O	0.1155720	0.3177836	-12.4327495
NO^+	0.3030141	-3.5039791	-3.7355157

Mixture viscosity is given by Wilke's relation

$$\mu_{mix} = \sum_{i=1}^{N_s} \frac{X_i \mu_i}{\sum_{j=1}^{N_s} X_j \varphi_{ij}} \tag{2.28}$$

$$\text{where } \phi_{ij} = \left[\frac{1 + \left(\frac{\mu_i}{\mu_j} \right)^{0.5} \left(\frac{MW_i}{MW_j} \right)^{0.25}}{\sqrt{8}\left[1 + \frac{MW_i}{MW_j} \right]^{0.5}} \right] \tag{2.29}$$

Thermal conductivity for monatomic gas and molecule are given by Svehla (1962)

For mono-atomic gas

$$K_i = \frac{15}{4}\left(\frac{\mu_i R}{MW_i} \right) \tag{2.30}$$

And for molecule

$$K_i = \left[\frac{15}{4} + \frac{1}{Sc}\left(\frac{C_{pi} MW_i}{R} - \frac{5}{2} \right) \right]\left(\frac{\mu_i R}{MW_i} \right) \tag{2.31}$$

$$Sc = \frac{\mu}{\rho D} = \frac{\Pr}{Le} \ , \ \Pr = \frac{\mu C_p}{K}, \ Le = \frac{\rho D C_p}{K}$$

Where Sc is the Schmidt number, $\Pr$ is the Prandtl number and Le is the Lewis number

The diffusion coefficient is calculated from specified Schmidt number.

$D = \frac{\mu}{\rho Sc}$ and the effective species diffusion coefficients are obtained from the equation

$$D_i = \frac{(MW_i / MW)(1 - \rho_i / MW_i) D}{1 - X_i} \tag{2.32}$$

In the above expression MW is the mixture molecular weight. The diffusion coefficient of ions is generally assumed to be twice as that of neutral species, due to linking of electron and ion diffusion in the presence of electric field. The effective

diffusion coefficient of electrons D_e is obtained by equating the diffusion velocity of ions with that of electrons.

2.4 Thermodynamic Model and Transport Properties for Hydrogen- Air Combustion

The enthalpy of each species formed during Hydrogen-air combustion is expressed in terms of polynomial curve fit given by Kee et al. (1992) and is given below.

$$\frac{h_i}{R_U T} = a_1 + \frac{a_2 T}{2} + \frac{a_3 T^2}{3} + \frac{a_4 T^3}{4} + \frac{a_5 T^4}{5} + \frac{a_6}{T} \tag{2.33}$$

Table 2.4 Table of enthalpy curve fit coefficients used for species in Hydrogen-air combustion for the temperature range 300-1000 K from Kee et al. (1992)

Species	Coefficients					
	a_1	a_2	a_3	a_4	a_5	a_6
H_2	0.03298e02	0.08249e-02	-0.08143e-05	-0.09475434e-09	.04134872e-11	-0.10125e04
O_2	0.03212e02	0.11274e-02	-0.05756e-05	0.13138773e-08	-0.08768554e-11	0.100524e04
H_2O	0.033868e02	0.0347498e-01	-0.06354e-04	0.06968581e-07	-0.02506588e-10	-.030208e06
OH	0.036372e02	0.018509e-02	-0.16761e-05	0.02387202e-07	-0.08431442e-11	0.03606e05
H	0.0250e02	0.0	0.0	0.0	0.0	0.025471e06
O	0.02946e02	-0.16381e-02	0.02421e-04	-0.16028431e-08	0.03890696e-11	0.029147e06
N_2	0.03298e02	0.140824e-02	-0.39632e-04	0.05641515e-07	-0.02444854e-10	-0.10208e04
CO_2	0.022757e02	0.099220e-01	-0.10409e-04	0.06866686e-07	-0.02117280e-10	-.048373e06

Table 2.5 Enthalpy curve fit coefficients for species used in Hydrogen-air combustion in the temperature range 1000 K-5000 K from Kee et al. (1992)

Species	Coefficients					
	a_1	a_2	a_3	a_4	a_5	a_6
H_2	0.029914e02	0.070006e-02	-.056338e-06	-0.09231578e-10	0.15827519e-14	-0.08350e04
O_2	0.036975e02	0.0613519e-02	-0.125884e-06	0.01775281e-09	-0.11364354e-14	-0.12339e04
H_2O	0.026721e02	0.0305629e-01	-0.08730e-05	0.12009964e-09	-0.06391618e-13	-0.029899e06
OH	0.028827e02	0.1013974e-02	-0.022768e-05	0.02174683e-09	-0.05126305e-14	0.038836e05
H	0.0250e02	0.0	0.0	0.0	0.0	0.025471e06
O	0.025420e02	-0.027550e-03	-0.031028e-07	0.04551067e-09	-0.04368051e-14	0.029230e06
N_2	0.029266e02	0.1487976e-02	-0.056847e-05	0.10097038e-09	-0.06753351e-13	-0.09227e04
CO_2	0.044536e02	0.0314016e-01	-0.12784e-05	0.02393996e-08	-0.16690333e-13	-0.04896e06

As for the transport properties used for turbulent Hydrogen- air computations, the laminar viscosity is obtained from the Sutherland law of viscosity given as follows

$$\frac{\mu}{\mu_{ref}} = \left(\frac{T}{T_{ref}}\right)^{3/2} \frac{T_{ref} + 110}{T + 110} \tag{2.34}$$

Where μ_{ref} is the viscosity of air at reference temperature, T_{ref} and the turbulent viscosity is obtained from standard κ-.ε turbulence model. It is to be noted that the turbulent viscosity is much higher than laminar viscosity and hence the computations are dominated by turbulent viscosity. For this reason, the viscosity evaluation for turbulent flow combustion computations was carried out using Sutherland law to reduce the computational effort.

The diffusion coefficient is obtained from the assumption of unity Lewis number and thermal conductivity is obtained from laminar Prandtl number of 0.72 and turbulent Prandtl number of 0.95. It is worth noting that in the case of supersonic combustion in Scramjets, the flow is convection dominated and diffusion velocity is very small compared to the convective velocity.

2.5 Air Chemistry Model

For a general set of elementary chemical reactions involving n_s species, the reactions can be written as

$\sum_{s=1}^{n_s} \alpha_{s,r} \psi_s \Leftrightarrow \sum_{s=1}^{n_s} \beta_{s,r} \psi_s$ where ψ_s are the chemical symbols and the time rate of production of species per unit volume, $\dot{w}_s$ can be written as

$$\dot{w}_s = MW_S \sum_{r=1}^{N_r} (\beta_{s,r} - \alpha_{s,r})(R_{f,r} - R_{b,r}) \tag{2.35}$$

$$R_{f,r} = K_{f,r} \prod_{s=1}^{N_s} \left(\frac{\rho_s}{MW_s}\right)^{\alpha_{sr}} \text{ and } R_{b,r} = K_{b,r} \prod_{s=1}^{N_s} \left(\frac{\rho_s}{MW_s}\right)^{\beta_{sr}} \tag{2.36}$$

The Dunn-Kang air chemistry model (1973) for 7 species is given in Table 2.6.

Forward and Backward reaction rate coefficients are given by

$$K_{f,r} = C_{f,r} T^{nf,r} e^{-(E_{f,r}/kT)} \tag{2.37}$$

$$K_{b,r} = C_{b,r} T^{nb,r} e^{-(E_{b,r}/kT)} \tag{2.38}$$

Table 2.6 Dunn-Kang chemistry model (1973)

No	Reaction	$C_{f,r}$	$n_{f,r}$	$E_{f,r}/k$	$C_{b,r}$	$n_{b,r}$	$E_{b,r}/k$
1	$O_2 + M \Leftrightarrow 2O + M$ $(M = N, NO)$	0.36e19	-1.0	0.595e05	0.30e16	-0.5	0.0
3	$O_2 + O \Leftrightarrow 3O$	0.9e20	-1.0	0.595e05	0.75e17	-0.5	0.0
4	$2O_2 \Leftrightarrow O_2 + 2O$	0.324e20	-1.0	0.595e05	0.27e17	-0.5	0.0
5	$N_2 + O_2 \Leftrightarrow N_2 + 2O$	0.72e19	-1.0	0.595e05	0.60e16	-0.5	0.0
6	$N_2 + M \Leftrightarrow 2N + M$ $(M = O, NO, O_2)$	0.19e18	-0.5	0.113e06	0.11e17	-0.5	0.0
7	$N_2 + N_2 \Leftrightarrow N_2 + 2N$	0.47e18	-0.5	0.113e06	0.272e17	-0.5	0.0
8	$NO + M \Leftrightarrow N + O + M$ $(M = O_2, N_2)$	0.39e21	-1.5	0.755e05	0.10e21	-1.5	0.0
9	$NO + M \Leftrightarrow N + O + M$ $(M = O, N, NO)$	0.78e21	-1.5	0.755e05	0.20e21	-1.5	0.0
10	$NO + O \Leftrightarrow O_2 + N$	0.32e10	1.0	0.197e05	0.13e11	1.0	0.358e04
11	$N_2 + O \Leftrightarrow NO + N$	0.70e14	0.0	0.38e05	0.156e14	0.0	0.0
12	$O + N \Leftrightarrow NO^+ + e^-$	0.14e07	1.5	0.319e05	0.67e22	-1.5	0.0
13	$N_2 + M \Leftrightarrow N + N + M$ $(M = N)$	0.4085e23	-1.5	0.113e06	0.227e22	-1.5	0.0
14	$N_2 + O_2 \Leftrightarrow NO + NO^+ + e^-$	0.138e21	-1.84	0.141e06	0.10e25	-2.5	0.0
15	$N_2 + NO \Leftrightarrow N_2 + NO^+ + e^-$	0.22e16	-0.35	0.108e06	0.22e27	-2.5	0.0

In the case of Park-87 model as given in Table 2.7 the backward rate coefficient is given by

$$K_{b,r} = K_{f,r} / K_{c,r}^{eq} \tag{2.39}$$

where $K_{c,r}^{eq}$ is the equilibrium reaction coefficient and is given by

$$K_{c,r}^{eq} = \exp(B_1^r + B_2^r \ln z + B_3^r z + B_4^r z^2 + B_5^r z^3 \tag{2.40}$$

where $z = 10000/T$

Table 2.7 Park-87 reaction model by Park(1987)

Sl no	REACTION	$C_{f,r}$	$n_{f,r}$	$E_{f,r}/k$	B_1^r	B_2^r	B_3^r	B_4^r	B_5^r
1	$O_2+M \rightleftharpoons 2O+M(M=N,O)$	2.900E+23	-2.00	5.975E+04	2.855	0.988	-6.181	-0.023	-0.001
2	$O_2+M \rightleftharpoons 2O+M(M=N_2,O_2,$ NO ions)	9.680E+22	-2.00	5.975E+04	2.855	0.988	-6.181	-0.023	-0.001
3	$N_2+N \rightleftharpoons 2N+N$	1.600E+22	-1.60	1.132E+05	1.858	-1.325	-9.856	-0.174	0.008
4	$N_2+O \rightleftharpoons 2N+O$	4.980E+22	-1.60	1.132E+05	1.858	-1.325	-9.856	-0.174	0.008
5	$N_2+M \rightleftharpoons 2N+M(M=N_2, O_2)$	3.700E+21	-1.60	1.132E+05	1.858	-1.325	-9.856	-0.174	0.008
6	$N_2+NO \rightleftharpoons 2N+NO$	4.980E+21	-1.60	1.132E+05	1.858	-1.325	-9.856	-0174	0.008
7	$N_2+ions \rightleftharpoons 2N+ions$	8.300E+24	-1.60	1.132E+05	1.858	-1.325	-9.856	-0174	0.008
8	$NO+M \rightleftharpoons N+O+M(M\neq$electrons)	7.950E+23	-2.00	7.550E+04	0.792	-0.492	-6.761	-0.091	0.004
9	$NO+O \rightleftharpoons O_2+N$	8.370E+12	0	1.945E+04	-2.063	-1.480	-0.580	-0.114	0.005
10	$N_2+O \rightleftharpoons NO+N$	6.440E+17	-1.00	3.837E+04	1.066	-0.833	-3.095	-0.084	0.004
11	$O_2^+ +O \rightleftharpoons O_2+O^+$	6.850E+13	-0.52	1.860E+04	-0.276	0.888	-2.180	0.055	-0.003
12	$N_2+N^+ \rightleftharpoons N_2^+ +O$	9.850E+12	-0.18	1.210E+04	0.307	-1.706	-0.878	-0.004	-0.001
13	$N O^+ +O \rightleftharpoons NO+ O^+$	2.750E+13	0.01	5.100E+04	0.148	-1.011	-4.121	-0.132	0.006
14	$N_2+O^+ \rightleftharpoons N_2^+ +O$	6.330E+13	-0.21	2.220E+04	2.979	0.382	-3.237	0.168	-0.009
15	$N O^+ + O_2 \rightleftharpoons NO+ O_2^+$	1.030E+16	-0.17	3.240E+04	0.424	-1.098	-1.941	-0.187	0.009
16	$N O^+ +N \rightleftharpoons N_2^+ +O$	1.700E+13	0.40	3.550E+04	2.061	0.204	-4.263	0.119	-0.006
17	$N+O \rightleftharpoons N O^+ +e^-$	1.530E+09	0.37	3.200E+04	-7.053	-0.532	-4.429	0.150	-0.007
18	$O+O \rightleftharpoons O_2^+ +e^-$	3.850E+09	0.49	8.060E+04	-8.692	-3.110	-6.950	-0.151	0.007
19	$N+N \rightleftharpoons N_2^+ + e^-$	1.790E+09	0.77	6.750E+04	-4.992	-0.328	-8.693	0.269	-0.013
20	$O+ e^- \rightleftharpoons O^+ + e^- + e^-$	3.900E+33	-3.78	1.585E+05	-6.113	-2.035	-15.311	-0.073	0.004
21	$N+ e^- \rightleftharpoons N^+ + e^- + e^-$	2.500E+33	-3.82	1.686E+05	-3.441	-0.577	-17.671	0.099	-0.005

2.6 Hydrogen-Air Chemistry Model with 7 Species

The species considered in the 7 species chemistry model are H_2, O_2, H_2O, OH, H, O, and N_2. The chemical kinetics corresponds to the ONERA chemical kinetics model developed by ONERA is reported by Dmitry et al. (2003) is shown in Table 2.8 below

Table 2.8 ONERA Hydogen-air chemistry model reported by Dmitry et al. (2003)

No	Reaction	A,mol-cm-s	b	C(K)
1	$H_2 + O_2 -> 2OH$ $2OH -> H_2 + O_2$	$1.700X10^{13}$ $4.032X10^{10}$	0.0 0.317	24044 14554
2	$H_2 + M -> 2H + M$ $2H + M -> H_2 + M$	$5.086X10^{16}$ $9.791X10^{16}$	-0.362 -0.6	52105 0.0
3	$H_2 + OH -> H_2O + H$ $H_2O + H -> H_2 + OH$	$1.024X10^{8}$ $7.964X10^{8}$	1.6 1.528	1660 9300
4	$2OH -> H_2O + O$ $H_2O + O -> 2OH$	$1.506X10^{9}$ $2.220X10^{10}$	1.14 1.089	50 8613
5	$H_2 + O -> OH + H$ $OH + H -> H_2 + O$	$5.119X10^{4}$ $2.701X10^{4}$	2.67 2.649	3163 2240
6	$H + O_2 -> OH + O$ $OH + O -> H + O_2$	$1.987X10^{14}$ $8.930X10^{11}$	0.0 0.338	8456 -118
7	$H + OH + M -> H_2O + M$ $H_2O + M -> H + OH + M$	$2.212X10^{22}$ $8.963X10^{22}$	-2.0 -1.835	0.0 59743

H_2-O_2 reactions have chain reaction mechanism wherein an intermediate product produced in one step generates a reactive intermediate species in a subsequent step which in turn generates another reactive intermediate and so on. In such reaction process, highly reactive species are atoms such as H and O or radical species like OH. These chemical species which have unpaired electrons and can react very actively with other molecules are called free radicals. Elementary reactions which produce free radicals are called chain initiation reactions. The first and second reactions mentioned in the Table 2.8 are chain initiation reactions. While the first reaction produces highly reactive free radical OH, the second reaction produces a highly

reactive free radical species like H. Formation of Hydroxyl radical (OH) is a very important step in the combustion of H_2 and O_2. In the third and fourth reactions, the ratio of free radicals in the product to that in the reactants is 1 and such elementary reactions are called chain propagation or chain carrying reactions. The fifth and sixth reactions are called chain-branching reactions wherein the total number of free radicals in the products is more than that of reactants. The seventh reaction is a chain termination reaction which has the destruction of free radical OH.

The production rate of species involving a third body as given by Kee et al. (1986) is given below

$$\dot{w}_s = MW_s \sum_{r=1}^{N_r} (\beta_{s,r} - \alpha_{s,r}) \tilde{T}_r (R_{f,r} - R_{b,r}) \text{ Where } \tilde{T}_r = \sum_{s=1}^{N_s} t_{s,r} \frac{\rho_s}{MW_s} \quad (2.41)$$

Where $R_{f,r}$ and $R_{b,r}$ are given by Equation (2.36) and reaction rate coefficient K (forward and backward) is given by

$$K = AT^b e^{-C/T} \quad (2.42)$$

The coefficients A, b and C for forward and backward reactions are given in Table-2.8 and $t_{s,r}$ are the third body efficiencies for the reaction r. If all component species of the mixture contribute equally as third bodies in a particular reaction then all $t_{s,r}$ are unity for this case. Any particular species may be excluded from being a third body in a reaction by setting its third body efficiency to zero for that reaction. The third body efficiency of water vapour is taken as 12 and of Hydrogen 2.5, and for all other species the third body efficiency is taken as 1.

CHAPTER-3
NUMERICAL METHOD

The numerical solution of the Navier-Stokes equation in a finite volume scheme consists of the important steps of computation of inviscid fluxes and viscous fluxes across the faces, calculation of production rate of species in the control volume and update of the variables. Before the computation of inviscid fluxes, the primitive variables are linearly reconstructed at the interface. The gradients computed for linear reconstruction are limited by means of limiters like Min-Mod or Venkatakrishnan limiter proposed by Venkatakrishnan (1995). After the reconstruction of primitive variables, the interface fluxes are calculated by means of an approximate Riemann solver. The approximate Riemann solver used for the present work is Advective Upstream Splitting Method (AUSM). The viscous fluxes are evaluated at each face from gradients obtained using the Green-Gauss approach. Local time stepping is done and the local time step ensures that the disturbance do not propagate more than the particular cell in one time step. The numerical scheme is fully explicit and based on the local time step, the conserved variables are updated by a backward Euler method. In the case of species conservation equations, point implicit scheme is used to compute the species density at the next time step.

3.1 Computation of Inviscid Fluxes

The inviscid fluxes across the interfaces are computed using the AUSM scheme. The Advective Upstream Splitting Method developed by Liou and Stefen (1993) is formulated using the time-dependent Euler equations and relies on splitting the flux vector F into convective component $F^{(c)}$ and a pressure component $F^{(p)}$. The basic idea of this approach is from the observation as given in equation (3.1) that the convective flux vector consists of two physically distinct parts, namely the convective and the pressure part. The first term of the Equation (3.1) represents the scalar quantities which are convected with the contravariant velocity, V and the second

term consisting of pressure is governed by acoustic wave speed. The convective terms are discretised in purely upwind manner by taking the left or the right state, depending on the sign of the contravariant velocity V . The pressure term on the other hand includes both states in the subsonic case and becomes fully upwind in the supersonic flow case.

$$F(U) = V \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \\ \rho Z_{N_2} \\ \rho Z_{O_2} \\ \rho Z_{NO} \\ \rho Z_{O} \\ \rho Z_{N} \\ \rho Z_{NO^+} \\ \rho Z_{e} \end{bmatrix} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.1}$$

For the x split three dimensional flux, the intercell numerical flux $F_{i+1/2}$ is defined as $F_{i+1/2} = F_{i+1/2}^{(c)} + F_{i+1/2}^{(p)}$ where the convective flux component is given by

$$F^{(c)}_{i+\frac{1}{2}} = M_{i+\frac{1}{2}} \left(\hat{F}^{(c)} \right)_{i+\frac{1}{2}} \tag{3.2}$$

$$[\bullet]_{i+\frac{1}{2}} = [\bullet]_i \text{ if } M_{i+\frac{1}{2}} \geq 0 \tag{3.3}$$

$$[\bullet]_{i+\frac{1}{2}} = [\bullet]_{i+1} \text{ if } M_{i+\frac{1}{2}} \leq 0 \tag{3.4}$$

The flux vector in Equation (3.2) is upwinded according to the sign of the convection or advection speed implied in the inter-cell Mach number $M_{i+\frac{1}{2}}$. Thus this scheme is called Advection Upstream Splitting Method. The cell interface Mach number is given by splitting

$$M_{i+\frac{1}{2}} = M_i^+ + M_{i+1}^- \tag{3.5}$$

The splitting of Mach number into positive and negative component is done as follows

$$M^{\pm} = \pm\frac{1}{4}(M \pm 1)^2 \text{ if } |M| \le 1 \tag{3.6}$$

$$M^{\pm} = \frac{1}{2}(M \pm |M|) \text{ if } |M| > 1 \tag{3.7}$$

The pressure is split as follows

$$p^{\pm} = \frac{1}{2}p(1 \pm M) \text{ if } |M| \le 1 \tag{3.8}$$

$$p^{\pm} = \frac{1}{2}p\frac{(M \pm |M|)}{M} \text{ if } |M| > 1 \tag{3.9}$$

$$F(U) = M_{i+1/2}\begin{bmatrix} \rho a \\ \rho a u \\ \rho a v \\ \rho a w \\ \rho a H \\ \rho a Z_{N_2} \\ \rho a Z_{O_2} \\ \rho a Z_{NO} \\ \rho a Z_O \\ \rho a Z_N \\ \rho a Z_{NO^+} \\ \rho a Z_e \end{bmatrix} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.10}$$

3.2 Solution Reconstruction and Limiter

In order to have second order accuracy, the primitive variables from the cell center are linearly extrapolated to the cell face by assuming a linear variation. This can be expressed as follows where U is the vector of primitive variables

$$U_L = U_I + \psi_I(\nabla U_I . \vec{r}_L) \tag{3.11}$$

$$U_R = U_J + \psi_J (\nabla U_J . \vec{r}_R) \tag{3.12}$$

Where ∇U_I (gradient of U) $=(\partial U/\partial x, \partial U/\partial y, \partial U/\partial z)^T$ at the cell center I and ψ denotes the a limiter function. $\vec{r}_L, \vec{r}_R$ are vectors from the cell centroid to the face midpoint as shown in Figure 3.1.

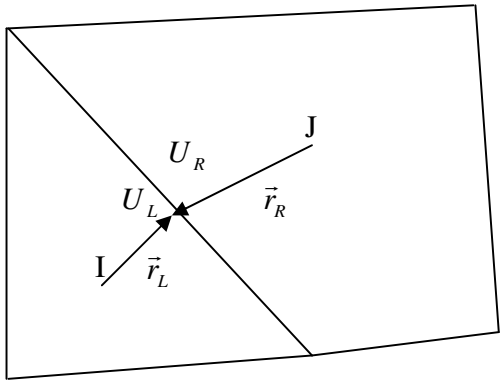


Figure 3.1 Linear reconstruction from cell center I and J

The second and higher order upwind discretisations require the use of limiters in order to prevent oscillations and spurious solutions in regions of large gradients like shocks. This means that a monotone preserving scheme is sought after which will have a non-increasing maxima and non-decreasing minima. The limiter limits the slopes of the primitive variables during reconstruction procedure and in regions of very strong gradients the slope reduces to zero or in other words reduces to a first order scheme. The existing Cartesian mesh solver has a Min-Mod limiter and Venkatakrishnan limiter is incorporated for unstructured solver. The Venkatakrishnan limiter (1995) is widely used for unstructured solvers because of its good convergence properties. The limiter reduces the reconstructed gradient ∇U at the cell center I by the factor

$$\psi_I = \min_J \left\{ \frac{1}{\Delta_2} \left[\frac{(\Delta_{1,\max}^2 + \varepsilon^2)\Delta_2 + 2\Delta_2^2 \Delta_{1.\max}}{\Delta_{1,\max}^2 + 2\Delta_2^2 + \Delta_{1,\max}\Delta_2 + \varepsilon^2} \right] \right\} \text{ if } \Delta_2 > 0 \tag{3.13}$$

$$\psi_I = \min_J \left\{ \frac{1}{\Delta_2} \left[\frac{(\Delta_{1,\min}^2 + \varepsilon^2)\Delta_2 + 2\Delta_2^2 \Delta_{1.\min}}{\Delta_{1,\min}^2 + 2\Delta_2^2 + \Delta_{1,\min}\Delta_2 + \varepsilon^2} \right] \right\} \text{ if } \Delta_2 < 0 \tag{3.14}$$

$$\psi_I = 1 \text{ if } \Delta_2 = 0 \tag{3.15}$$

Where $\Delta_{1,\max} = U_{\max} - U_i$ and $\Delta_{1,\min} = U_{\min} - U_i$ (3.16)

$\Delta_2 = (\nabla U_i . \vec{r}_L)$ and $U_{\max} = \max(U_i, \max_j U_j), U_{\min} = \min(U_i, \min_j U_j)$ (3.17)

In Equation (3.15), $U_{\max}$ and $U_{\min}$ stand for the maximum and minimum values of all surrounding cells including the cell I. $\vec{r}_L$ is the vector from cell center to the mid point of corresponding cell face. The parameter ε^2 controls the amount of limiting and in practice it should be proportional to a local length scale as follows

$$\varepsilon^2 = (K\Delta h)^3 \quad (3.18)$$

where Δh is the cube root of the control volume. The value of K varies from 0 to 50. As the K value increases the solution value gradually tends to be unlimited resulting in overshoot at the shock location.

3.3 Computation of Viscous Fluxes

For the evaluation of diffusive fluxes F_v in Equation (2.1), the flow quantities and their first derivatives have to be known at the faces of the control volumes. The control volume for the viscous fluxes is chosen to be the same as that of the convective fluxes in order to obtain a consistent spatial discretisation and to simplify the data structure. Because of the elliptic nature of the viscous fluxes, values of the velocity components, the dynamic viscosity, and the heat conduction coefficient which are needed for the computation of viscous fluxes are simply averaged at a face.

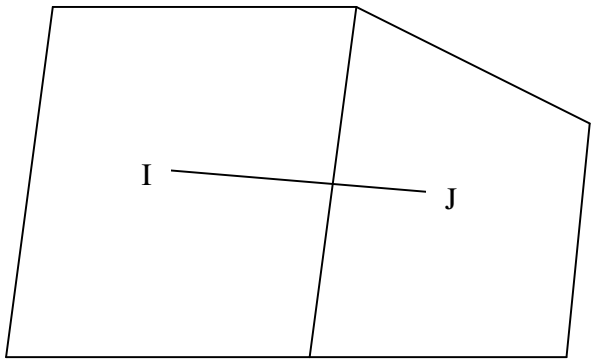


Figure 3.2 Viscous flux computation stencil with cell centers I and J.

Thus for a cell centered finite volume scheme, the values of the variables at face between cell I and cell J as shown in Figure 3.2 of the control volume is given by

$$U_{IJ} = \frac{1}{2}(U_I + U_J) \tag{3.19}$$

However, for the gradients at the cell face, the simple averaging of gradients of the two adjoining cells of the face would give rise to decoupling of the solution on quadrilateral and hexahedral grids. This decoupling can be overcome by using the directional derivative along the connection between cell centroids i.e.

$$\left(\frac{\partial U}{\partial l}\right)_{IJ} \approx \frac{U_J - U_I}{l_{IJ}} \tag{3.20}$$

Where l_{IJ} represents the distance between the cell centroids I and J. The unit vector $\vec{t}_{IJ}$ along the line connecting I and J is given by

$$\vec{t}_{IJ} = \frac{\vec{r}_{IJ}}{l_{IJ}} \tag{3.21}$$

Considering all the above points the average gradient is written by Crumpton et al. (1997)

$$\nabla U_{IJ} = \overline{\nabla U_{IJ}} - \left[\overline{\nabla U_{IJ}}.\vec{t}_{IJ} - \left(\frac{\partial U}{\partial l}\right)_{IJ}\right]\vec{t}_{IJ} \tag{3.22}$$

Where $\overline{\nabla U_{IJ}}$ is the average of cell center gradient at cells I and J and expressed as

$$\overline{\nabla U_{IJ}} = \frac{1}{2}(\nabla U_I + \nabla U_J) \tag{3.23}$$

The cell center gradients are evaluated by the standard Green-Gauss procedure. The Equation (3.21) when implemented leads to strongly coupled stencils on tetrahedral, prismatic, and hexahedral grids.

3.4 Local Time-Stepping and Update Procedure

An explicit scheme starts from a known solution U^t and employs the corresponding residual R^t in order to obtain a new solution at time $(t + \Delta t)$. The time-stepping

scheme remains stable only up to a certain value of time step Δt. To be stable, a time-stepping scheme has to satisfy the Courant-Freidrichs-Levy (CFL) criterion. This physically means that the disturbance should not propagate more than one cell in the explicit scheme. In the one dimensional condition, the time step for a linear convection equation is

$$\Delta t = \sigma \frac{\Delta x}{|\Lambda_C|} \tag{3.24}$$

Where σ is the CFL number and is a positive coefficient, Δx is the cell size and Λ_C denotes the maximum eigen value of the convective flux Jacobian. For linear equation, the maximum time step can be calculated with the help of von Neumann stability analysis. However, for non-linear equations in multiple dimensions, there is no exact theory to compute the maximum time step for a particular cell size and flow conditions. The time step calculation for any general type of mesh can be estimated with the following expression of Vijayan and Kallinderis (1994)

$$\Delta t_{cell} = \sigma \frac{\Omega_{cell}}{(\Lambda_C^x + \Lambda_C^y + \Lambda_C^z)_{cell} + (\Lambda_v^x + \Lambda_v^y + \Lambda_v^z)_{cell}} \tag{3.25}$$

where σ is the CFL number and $\Lambda_C^x, \Lambda_C^y, \Lambda_C^z$ are the convective spectral radii which are given as

$$\left.\begin{aligned} \Lambda_C^x &= (|u| + a)\Delta S_x \\ \Lambda_C^y &= (|v| + a)\Delta S_y \\ \Lambda_C^z &= (|w| + a)\Delta S_z \end{aligned}\right\} \tag{3.26}$$

The viscous spectral radii for the x direction is expressed as

$$\Lambda_v^x = \max\left(\frac{4}{3\rho}, \frac{\gamma}{\rho}\right)\left(\frac{\mu_{laminar}}{\mathrm{Pr}_{laminar}} + \frac{\mu_{turbulent}}{\mathrm{Pr}_{turbulent}}\right)\left(\frac{(\Delta S_x)^2}{\Omega_{cell}}\right) \tag{3.27}$$

The variables $\Delta S_x, \Delta S_y, \Delta S_z$ are the projections of the control volume on the y-z, x-z, and x-y plane. They are given by the following expressions

$$\left.\begin{aligned}\Delta S_x &= \frac{1}{2}\sum_{j=1}^{N_F}\left|S_x\right|_j \\ \Delta S_y &= \frac{1}{2}\sum_{j=1}^{N_F}\left|S_y\right|_j \\ \Delta S_z &= \frac{1}{2}\sum_{j=1}^{N_F}\left|S_z\right|_j\end{aligned}\right\} \tag{3.28}$$

Where S_x, S_y, S_z are the x,y and z component of the face vector $\vec{S} = \vec{n}.\Delta S$. To have faster convergence local time stepping is employed i.e each cell will have different time step. However if unsteady time accurate solutions are needed then the minimum of the local time step over all the cells should be used for all cells. The update procedure can be explained from equation below, which the governing Navier-Stokes equation is already given by Equation (2.1) in Section 2.1

$$\frac{\partial}{\partial t}\int_{\Omega} U d\Omega + \oint (F_c - F_v) dS = \int_{\Omega} W d\Omega$$

This equation can be expressed as

$$\frac{(U_{cell}{}^{t+\Delta t} - U_{cell}{}^{t})}{\Delta t_{cell}}\Omega_{cell} = R^t_{cell} \tag{3.29}$$

Where R^t_{cell} is the residue of the cell at time t. From the above equation $U^{t+\Delta t}_{cell}$ can be easily calculated using the backward Euler method since that is the only unknown.

3.5 Point Implicit Method for Source Terms

In chemically reacting flows, the characteristic time of chemical reaction can be much smaller than the characteristic flow time. Under such circumstances the explicit treatment of source term would give rise to stiffness if the time step is based on flow time. Stiffness of system of equations is defined as the ratio of largest to smallest time scales present. In spite of local time stepping, the chemical species change is much

more rapid than the evolution of the fluid dynamic variables such as momentum and make the system stiff to the explicit time integration methods. This problem can be alleviated by individually scaling the time scales associated with each of the species equation to the same order of magnitude as the global fluid dynamic equation. This effect is introduced through the point implicit treatment of the source terms by Bussing and Murman (1988) as given below

From Equation (2.1) the solution from explicit scheme would give rise to

$$U^{t+\Delta t} = U^{t} + \Delta t.R(U^{t}) \tag{3.30}$$

For the point implicit scheme the above expression can be written as

$$U^{t+\Delta t} = U^{t} + \Delta t.R(U^{t}, U^{t+\Delta t}) \tag{3.31}$$

$$R(U^{t}, U^{t+\Delta t}) = W^{t+\Delta t} - \frac{1}{\Omega}\sum_{faces}(\vec{F}.\vec{n})\Delta S \tag{3.32}$$

Where the flux vector $\vec{F}$ has both inviscid and viscous fluxes and the source vector W is evaluated at $t+\Delta t$ i.e. at next time level instead of the current time level t. A Newton linearisation of the source term gives rise to

$$W^{t+\Delta t} = W^{t} + \left(\frac{\partial W}{\partial U}\right)^{t}(U^{t+\Delta t} - U^{t}) \tag{3.33}$$

Which leads to

$$U^{t+\Delta t} = U^{t} + \frac{\Delta t.R(U^{t})}{(I - \Delta t.H^{t})} \tag{3.34}$$

Where H is the source term Jacobian expressed as

$$H^{t} = (\partial W / \partial U)^{t} \tag{3.35}$$

$$\frac{\partial W_i}{\partial U_i} = \frac{MW_i}{\rho_i}\sum_{r=1}^{N_R}(\beta_{i,r} - \alpha_{i,r})\left\{K_{fr}\alpha_{i,r}\prod_i (X_i)^{\alpha_{i,r}} - K_{br}\beta_{i,r}\prod_i (X_i)^{\beta_{i,r}}\right\} \tag{3.36}$$

Where $\alpha_{i,r}$ is the stoichiometric coefficient of i^{th} species reactant in r^{th} reaction and $\beta_{i,r}$ is the stoichiometric coefficient of i^{th} species product in r^{th} reaction. K_{fr} and K_{br} are the forward and backward reaction rate coefficients and X_i is the mole fraction of i^{th} species. Solution of Equation (3.32) involves solution of simultaneous

equations and for 7 species chemistry model, a 7X7 matrix inversion is needed to estimate species production terms. However, if only the diagonal terms of the source term Jacobian is calculated which are the most dominant terms, then the scheme is called Diagonal Point Implicit Scheme and thus no matrix inversion is involved in the solution of equations. The point implicit scheme essentially under-relaxes the species source terms, while allowing the flow solution to be updated by Δt dictated by the CFL condition for stability. This would result in each of the species advancing with a different time step and hence the chemical composition no longer consistent, or the method time accurate. However, the correct final steady state solution is reached much more rapidly. In conjunction with the local time stepping, this provides faster signal propagation and improvement in the convergence to steady state.

3.6 Species Under-Relaxation

Species under-relaxation is an explicit method for handling the source terms of the chemical reactions. In the time marching method of driving the solution to steady state, the chemical species evolve rather violently over the initial stages of the solution. The effect manifests as wild oscillations in the convergence history and could also lead to solution blow up because of the interaction of wildly errant transient chemical states with other flow variables like temperature. If this oscillation is controlled in the initial phase, then the algorithm would stably attain the steady state solution as the mixture reaches the final composition and temperature. A simple method to damp these wild oscillations in the initial phase is to under relax the solution updates. Thus only a fraction of the changes in the species mass fractions are applied while updating for the next level. However the species limiting during update must be physically consistent, lest the reaction kinetics are numerically modified resulting in an erroneous steady state solution. The changes in species mass fractions over a time step are computed from changes in species densities as developed by Palmer (1989) is given below

$$\Delta Z_i = \frac{\Delta\rho_i - Z_i^t \Delta\rho}{\rho^{t+\Delta t}} \tag{3.37}$$

A parameter, α which is the species under-relaxation parameter, controls the maximum allowable change in the species mass fraction over a time step. If $\Delta Z_i > \alpha$ then the changes are rescaled as follows

$$\Delta Z_i = \alpha \frac{\Delta Z_i}{|\Delta Z_{\max}|} \tag{3.38}$$

And the species mass fractions are updated as per the following equations

$$Z_i^{t+\Delta t} = Z_i^t + \Delta Z_i \text{ and } \rho_i^{t+\Delta t} = Z_i^{t+\Delta t} \rho^{t+\Delta t} \tag{3.39}$$

The above mentioned procedure retains the basic physics of the evolving reactions. For example if the original system produced twice as much atomic oxygen as atomic nitrogen, the nature is retained after scaling. This method is very useful for cold start-ups.

While the Point implicit scheme has also species under relaxation in its solution methodology, the species under-relaxation of Palmer is more explicit in nature. The species under-relaxation parameter converges to correct physical solutions over a large range of values from $0.0001 < \alpha < 0.1$. It is best to have this value as large as possible without the solution diverging. Setting this parameter to a very small value would result in very slow convergence although to a physically correct solution. Based on studies conducted, α value of 0.001 was found to give solutions with out diverging for wide range of problems.

3.7 Global Mass Conservation

The species mass conservation equations track the evolution of each of the chemical species. In the course of computations, the sum of the species densities would not add up to equal the global density obtained from the global mass conservation equation. This happens because of either numerical round off errors or

inconsistencies in forming the diffusive fluxes. Although such errors are not substantial, this could lead to deterioration of convergence and sometimes even solution divergence. This problem can be overcome by rescaling the species densities obtained either from point implicit or under-relaxation procedure so as to satisfy the global mass conservation as follows

$$\rho_i^{t+\Delta t} = \rho_i^{'} \frac{\rho^{t+\Delta t}}{\sum_i \rho_i^{'}} \tag{3.40}$$

Where $\rho_i^{'}$ is the density obtained after species under-relaxation or point implicit method. The above procedure can be considered as an effective over-relaxation method applied to species densities. By rescaling the species densities to satisfy the global density at each cell, which is at a higher time level of evolution, the chemical state of the gas is effectively extrapolated to larger time level.

CHAPTER-4
CARTESIAN MESH BASED SOLUTION FOR HIGH SPEED VISCOUS FLOWS

Solution of hypersonic viscous flow with Cartesian mesh based approach as applicable to reentry type flow and Scramjet engine flows with turbulent combustion is described in this Chapter. Laminar hypersonic flow is normally encountered for a substantial portion of ballistic reentry vehicle trajectory due to lower level of Reynolds number because of low atmospheric density at high altitudes. For such type of laminar flows, a mesh of Cartesian grid based prism layer near the wall and standard Cartesian grid away from the wall is adopted. The near wall prism layer is meant to resolve high gradients in the near wall viscous region so as to obtain wall quantities like heat flux. This approach is applied to laminar hypersonic perfect gas flows as well as to laminar real gas flows in chemical non-equilibrium. For the solution of high speed flows in Scramjet engines wherein the flow is turbulent and undergoes combustion, Cartesian mesh with wall function approach is followed. In this approach, the flow and geometry are both complex and the focus is to obtain the pressure distribution in the combustion chamber of the engine. Hence this method was found suitable from point of view of obtaining good accuracy in pressure with very low turnaround time owing to a completely automated grid generation and solution process.

The first approach of combined hybrid prism layer and Cartesian mesh approach is validated against experimental results for certain standard geometries and flow conditions. The second approach of pure Cartesian mesh approach with wall function is compared with that of the experimental results available for a typical Scramjet combustor tested with vitiated air under connected pipe mode conditions. The factors affecting the performance of the Scramjet engine and the effect of

vitiation of air in ground test conditions which are absent in the actual flight are also brought out in this chapter.

4.1 Combined Hybrid Prism Layer and Cartesian grid Approach for Laminar Hypersonic Flows

Cartesian grid that is generated for an arbitrary three dimensional body is a body piercing mesh which gives rise to three types of cells namely, the cells which are fully inside the body called the body cells, the cells which are fully outside the body called the air cells and cells which are partially inside the body and partially outside the body called as the partial cells. It is the partial cell that actually represents the body in a Cartesian mesh.

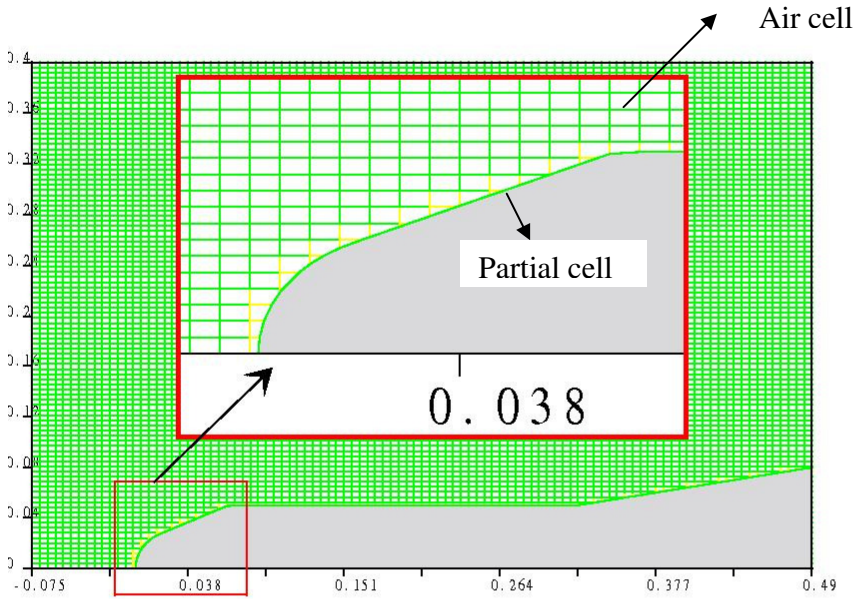


Figure.4.1 Cartesian mesh for a typical geometry with enlarged portion of the nose cone showing partial cell and air cells.

Figure 4.1 above shows typical sphere cone cylinder flare geometry captured with a basic Cartesian mesh and the enlarged region of the nose cone is shown which shows the partial cells near the body. A Cartesian mesh that intersects a three dimensional body in a plane would give rise to surface panels of 3 sides to 6 sides.

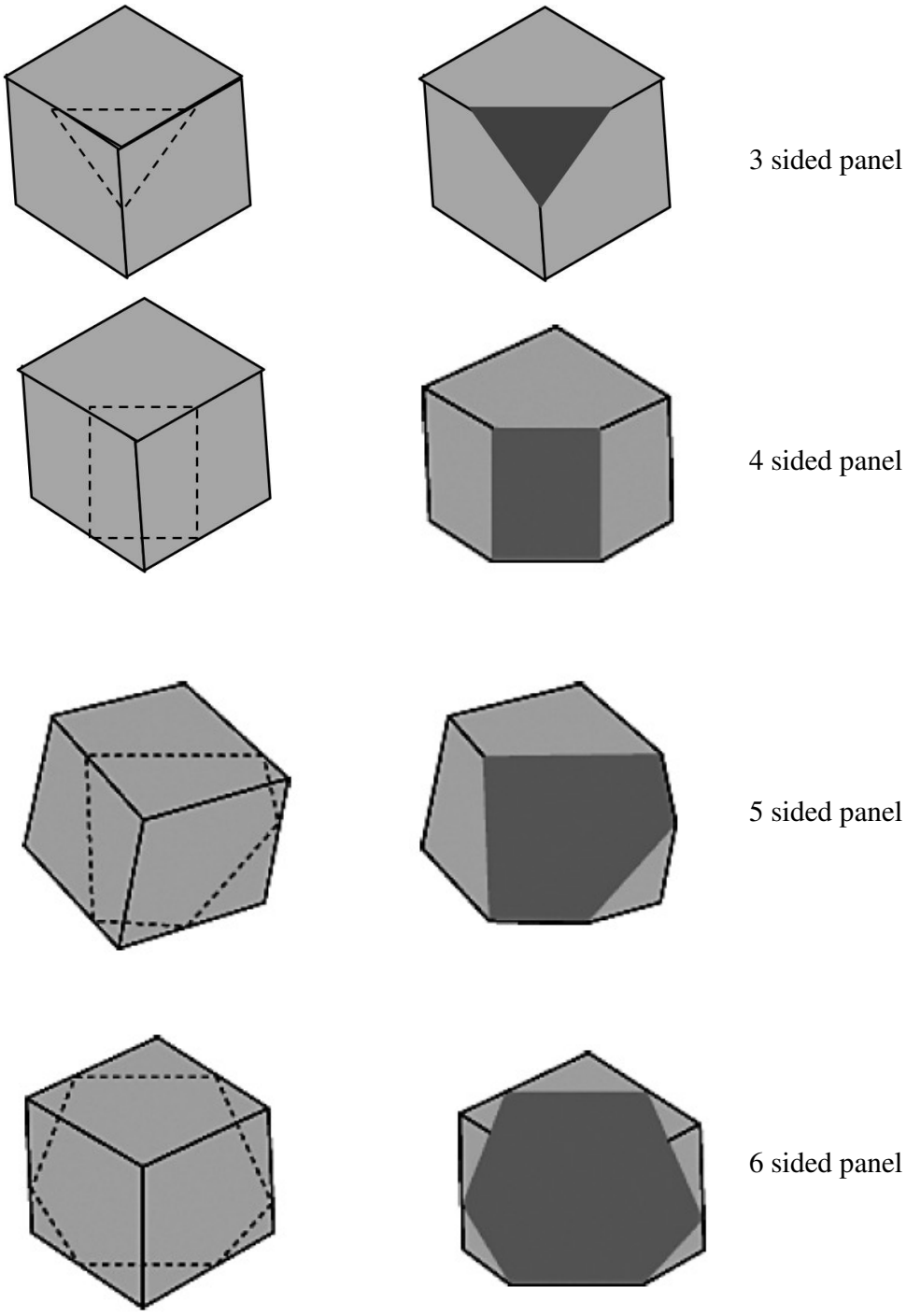


Figure 4.2 Surface panel of 3 sides to 6 sides produced by Cartesian cell intersecting with a plane

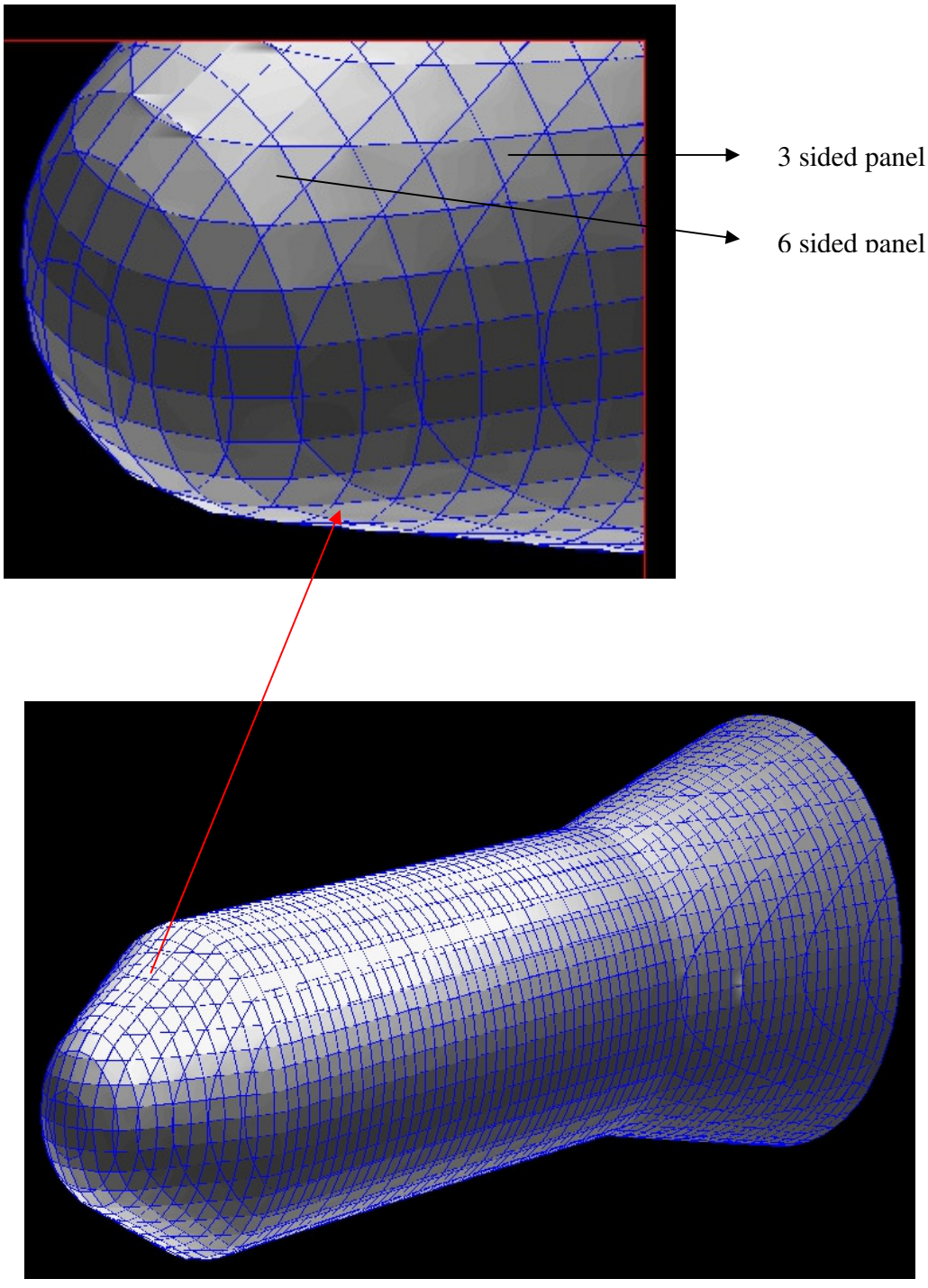


Figure.4.3 Panels formed by the intersection of Cartesian Mesh with a cone-cylinder-flare body with zoomed portion of the nose cone

Figure.4.2 shows the panels formed by the intersection of body with a Cartesian cell. Figure 4.3 shows the panels formed on the surface of the body by intersection of Cartesian mesh with a cone-cylinder-flare body. The procedure to generate hybrid prism layers from Cartesian mesh is listed in next section.

4.1.1 Procedure for generation of hybrid prism layers from Cartesian mesh

Step-1 Generate the Cartesian Mesh with the existing Cartesian mesh generator

Step-2 Find out the panels formed from the intersection of Cartesian mesh with the body

Step-3 Find out the average normal at each panel node. Normal of each panel is the normal corresponding to the largest triangle of the panel. To find out the normal at each panel node, the panels sharing a node is found out and from there the average normal at the node is found out.

Step-4 Along the average normal, the points are generated in a stretched fashion so as to capture the boundary layer. The stretching used is an algebraic stretching function with a stretching parameter varying from 1.01 to 1.2. The first prism layer height, Δ is given by the following expression

$$\Delta = \frac{h_o (sp-1)}{(sp)^{np} - 1} \qquad (4.1)$$

Where h_o is height from the wall up to which the prism layer has to be constructed, sp is the stretching parameter which is generally between 1.01 to 1.2 and np is the number of cells in the prism layer. The height of the second cell of the prism layer will be stretching parameter times the first cell height of prism layer. The number of points and the height up to which the prism layer are to be constructed are user defined inputs. Normally the height from the wall up to which the prism layer is to be constructed would be almost same as boundary layer thickness which can be estimated from approximate empirical relations.

Step-5 Join the points so that prism layer cells are obtained near the wall from the Cartesian mesh panels.

Step-6 In case there is cross over of the cells at regions of concave corners, which will give rise to negative volumes, the height up to which the prism layer has to be constructed is reduced or the average normal has to be changed so that the crossover of normals are avoided. Change of average normal is achieved by a user defined normal in the code. However this correction made is not automated.

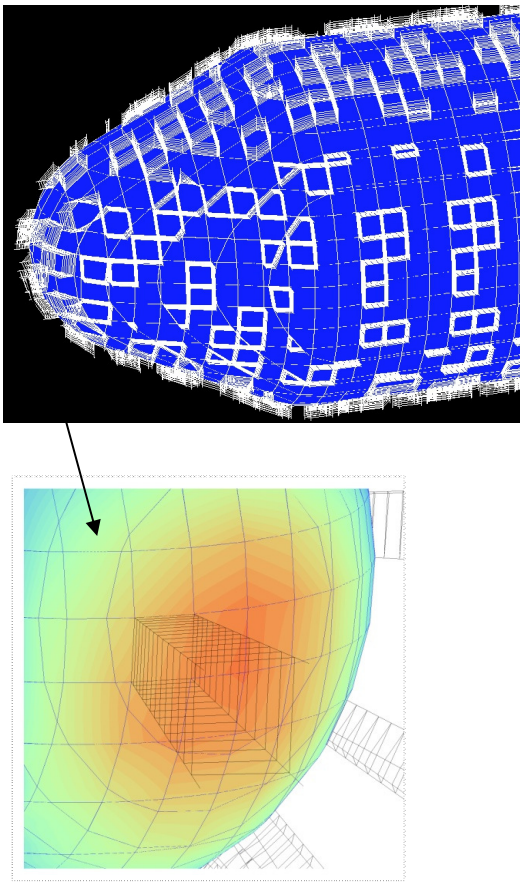


Figure 4.4 Hybrid prism layer for select panels with nose portion in zoomed view

In the above procedure, the stitching of the prism layer with the outer Cartesian mesh is not addressed and hence will not have flux continuity in the interface of prism layer cells and Cartesian mesh. Owing to this, the solution is obtained in two steps, namely

the Cartesian mesh solution on the initial mesh in the first step and the Navier-Stokes solution in the prism layer cells alone with the outer boundary condition being the Cartesian mesh solution in the next step. Figure 4.4 shows the hybrid prism layer extruded for select panels from the Cartesian mesh panels with the zoomed hybrid prism layer for one select panel.

4.1.2 Computation of flow over HB-2 geometry

To validate, the hybrid solver, a standard AGARD HB-2 model, which is a sphere- cone-cylinder-flare geometry, as shown in Figure.4.5 is chosen and for which hypersonic wind tunnel experiments conducted by Kuchi-Ishi et al. (2005) is available. The experiments are conducted in JAXA 1.27 m blow-down cold type hypersonic wind tunnel and one of the objectives of the tests is for generating accurate experimental data for HB-2 geometry, which would serve as benchmark for hypersonic computational fluid dynamics codes. The free stream conditions for which the numerical simulations are carried out are shown in Table.4.1.

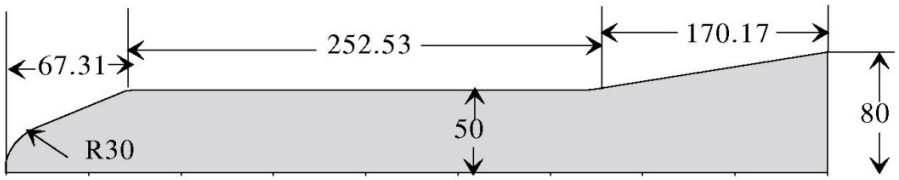


Figure 4.5 HB-2 geometry from Kuchi-Ishi et al. (2005) – (All dimensions in mm)

Table 4.1 Free stream conditions for HB-2 geometry from Kuchi-Ishi et al. (2005)

P_0 (Mpa)	T_0 (K)	M_∞	ρ_∞ (Kg/m^3)	T_∞ (K)	p_∞ (Pa)	U_∞ (m/sec)	Re (X10^5)	α (deg)	T_{wall} (K)
2.515	1027.4	9.59	0.00469	55.20	74.6	1430.8	1.85	0.0	300

Initially a basic Cartesian grid of 100X100 cells is generated as shown in Figure 4.6. An Euler solution is carried out with available Cartesian mesh solver, PARAS-3D which is widely used for solution to flow problems and reported by

Chakraborty et al.(2003), Manokaran et al. (2003) and Singh et al.(2009). The boundary conditions are supersonic inflow in the left boundary and supersonic outflow at all other outer boundaries and symmetry boundary conditions in the symmetry plane and slip condition for Euler solution and no slip and isothermal conditions on the wall for Navier-Stokes solution. The Cartesian mesh solution is given in Figure 4.7 and the figure shows that the all the flow features associated with inviscid flow field at high Mach numbers like the bow shock, expansion from the cone cylinder junction and the flare shock are captured satisfactorily. With this as the initial solution, the hybrid prism layer is generated for a prism layer height of 5cm and the Cartesian mesh Euler solution is mapped on to the hybrid prism layer. The laminar axi-symmetric Navier-Stokes solution is carried out for the prism layer alone with pressure of the Euler solution as boundary condition applied on the outer prism layer.

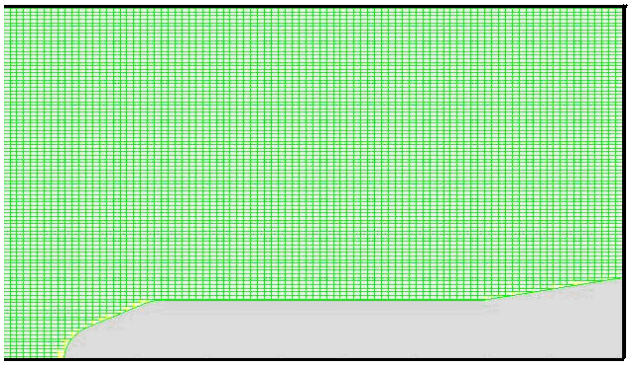


Figure.4.6 Basic Cartesian Mesh over HB-2 geometry

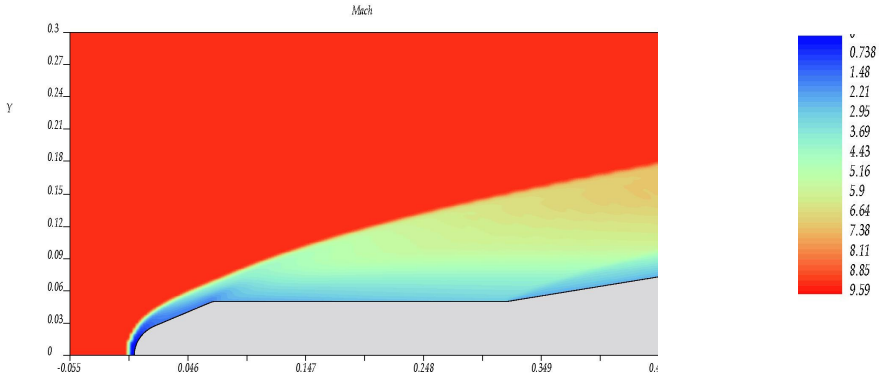


Figure 4.7 Mach number field from the Cartesian mesh Euler Solution

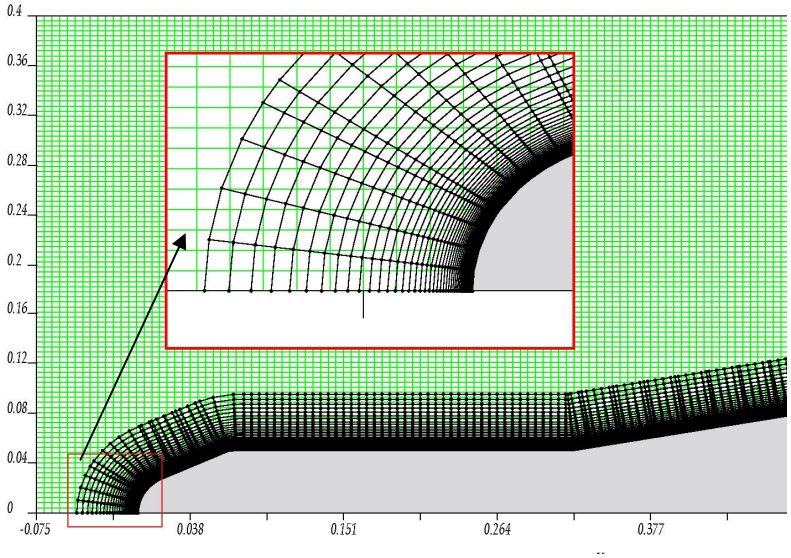


Figure 4.8 Hybrid prism layer with prism layer height of 5cm generated from the background Cartesian mesh

The cell data structure written in C- language and used for the computations is given below.

.

```
typedef union cell{                    typedef  struct SPCL{
        struct{                                    CELLS dcells[8];
    char level;                       }DCELLS;
    char ncel;
    char hybrid;              typedef struct PTCL{
    char type;                        double  Pt[10];
    double *U;                         char ncut;
    struct PTCL *pcp;                  int pnl;
  }item;                               } PCELL;
  struct {
     char st;
      struct SPCL *next;
   }attr;
}CELL;
```

```
typedef struct panel{
   int panel_number; double nx; double ny; double nz;
   struct Node *node; CELL *partial_cell; struct hybrid **hybrid_cell;
   struct side *side;int I; int J; int K;
    char i_arr[3][8];
}
typedef struct face{
 int nb_pnl_no; int nb_face_no; double dU[U_DIM]; double ddU[U_DIM]; char
flux_flag; char split; double d; double crd[3]; CELL **cart_cell;
} FACE
typedef struct hybrid_cell{
Int icell; double *Nx; doublel *Ny; double *Nz;;
double *Area; double Volume; struct face *face
} HYBRID_CELL;
```

The Cartesian mesh cell is a union of two structures where in if the cell is split into 8 children, it is represented by split cell structure, “SPCL”. Each cell has a level represented by a character (“char lev”) i.e denoting to what level it is already split (up to 8 levels is what is allowed in the present program) and another character “ncel”, representing whether it is partial cell, air cell or body cell. It also has a pointer to conserved variable vector U and depending on the type of problem Euler, Turbulent, or Chemically reacting flow, the number of flow variables are allocated during the start of the problem. If the cell is a partial cell then a pointer to the partial cell structure can get all the information about the partial cell, namely the partial fluxing area, outward normal of the panel of the partial cell, the number of sides of the panel of the partial cell and also the panel number corresponding to the partial cell.

The panel is represented by a data structure having the information regarding the pointer to the Cartesian partial cell which contains the panel, the number of sides of the panel (“char ncut”) which varies from 3 to 6 sides, normals to the panel, the

position “I, J, K” of the parent cell of the Cartesian mesh and also its position as a child represented by “char i_arr [3][8]”, if it is a panel corresponding to a split partial cell. If the partial cell is split, then the position of the child cell in I, J and K directions will be either 0 or 1 because of the oct-tree structure. Thus by knowing the split position in each level of split, the exact position of the split partial cell can be found out. The panel also has pointer to the array of hybrid prism layer cells and each hybrid prism layer cell has all the information about its face neighbours and the normals of each face. Each face is also a structure which has all the geometric information like area, normals and also the inviscid and viscous flux vector information. The panel also has information about coordinates of its node points.

As for the code modification to the existing Cartesian solver code; as the program encounters a partial cell, the solution branches to the calculation of unstructured hybrid prism layer which is a separate function added to the Cartesian solver code. The only user-defined inputs to the program are the number of prism layer cells, the stretching factor to adjust the first prism layer height and the normal distance from the wall up to which the prism layer has to constructed i.e. the extent to which the Cartesian mesh panels need to extruded in the normal direction.

For the above mentioned problem, the prism layer is not stitched with the outer Cartesian mesh and hence the coupling of unstructured prism layer viscous solution to the inviscid solution of the outer Cartesian mesh is not accounted. If the unstructured prism layer extends beyond the viscous inviscid interaction zone of the hypersonic flow, then the decoupled solution as done in the present case will yield good results. Figure.4.8 shows the Cartesian mesh with prism layer near the wall and figure 4.9 shows the velocity vector of the hybrid prism layer portion with the zoomed region clearly showing the boundary layer.

Figure 4.10 shows the cold wall heat flux along the length of the model compared with experimental results of Kuchi-Ishi et al. (2005). Wall heat flux is

obtained from the temperature gradient values evaluated at the wall based on the temperature of first wall cell and a cell above it. A good match is seen between the computations and the experiments. The solution converged after 20000 iterations. The maximum cold wall heat flux is as expected and is at the stagnation point and sharp reduction is seen in the conical portion. A slight increase in the flare region is noticed due to the presence of flare shock.

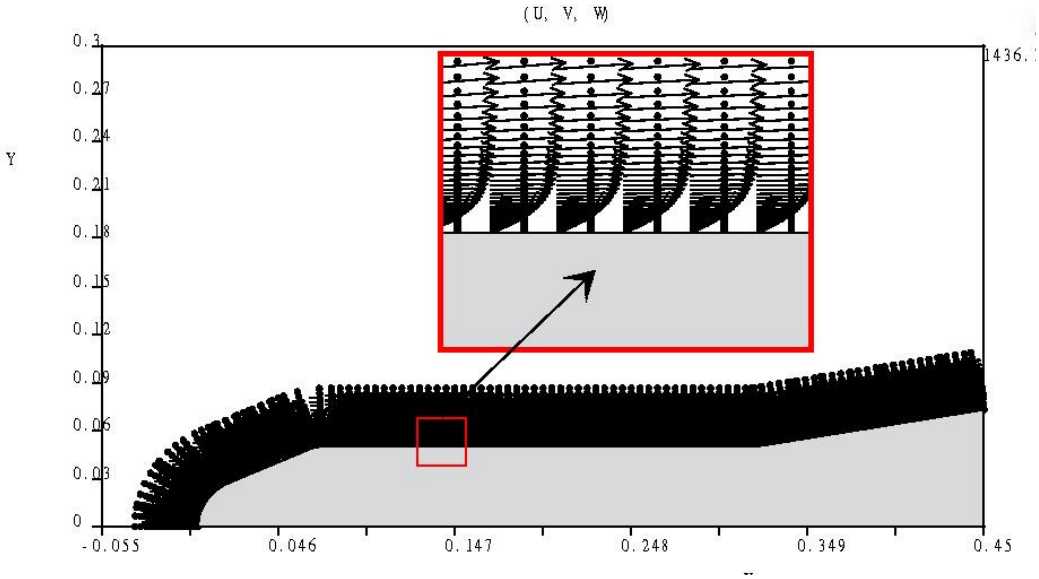


Figure.4.9 Velocity vector plot for the near wall prism layer cells

These computations are done in two steps, namely the inviscid computation on the Cartesian mesh for the whole domain in the first step and the near wall viscous computation for the unstructured prism layer in the subsequent step. To achieve this, the prism layer has to extend sufficiently to the outer region which encompasses the boundary layer region. In order to avoid the two step method of inviscid solution followed by boundary layer type solution, the prism layer has to be stitched with the outer Cartesian mesh and hence the next logical step in the near wall resolution within the framework of Cartesian mesh is to have a hybrid mesh with prism layer generated from background Cartesian mesh stitched with the outer Cartesian mesh to have flux continuity at the interface of prism layer and the outer Cartesian mesh.

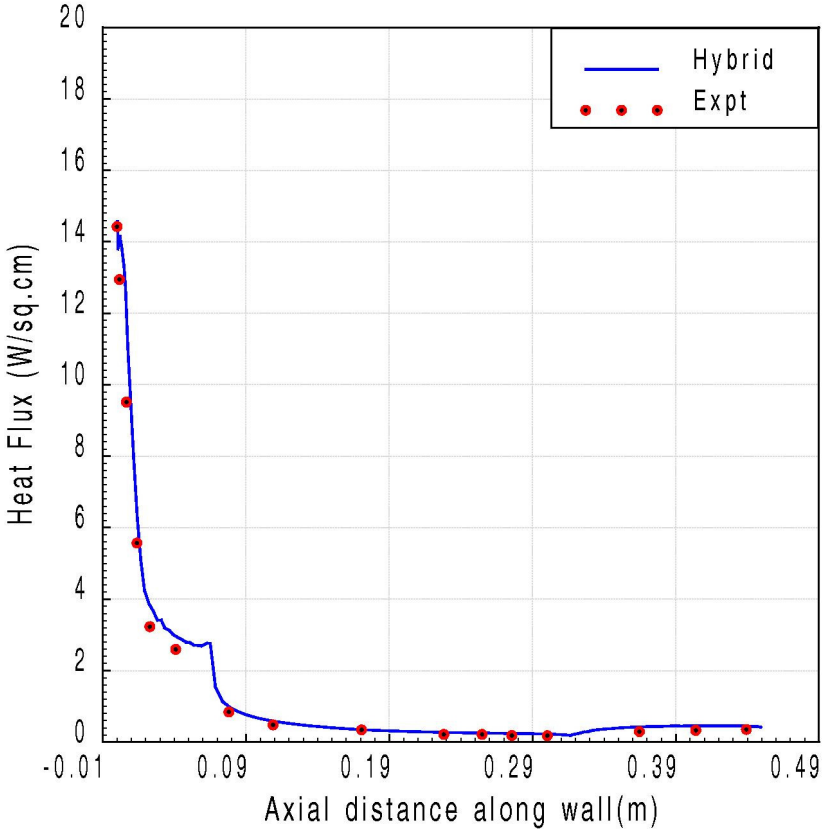


Figure.4.10 Comparison of computed cold wall heat flux with experiments from Kuchi-Ishi et al. (2005)

4.1.3 Near-wall resolution with Cartesian mesh based prism layer stitched with outer Cartesian mesh

In this methodology, the strong viscous gradients near the wall are resolved through the unstructured prism layer and away from the wall where the inviscid effects are dominant, the Cartesian mesh is used. In order to have flux continuity and to capture the viscous inviscid interaction effects, the prism layer is stitched to the outer Cartesian mesh. The methodology to generate the hybrid mesh is described below.

Steps to generate near wall prism layer stitched with outer Cartesian mesh

1) Start with the Cartesian mesh having information on the intersection point of mesh with the body. The surface of the body is essentially the panels formed by intersection of Cartesian mesh with the body. For 3D geometry, the surface panels have 3 to 6 sides whereas for a 2D geometry surface panels have 4 sides with one grid in the third direction as in the present problem.
2) The normal of each panel corresponds to the normal of the largest triangle of the panel. Find out the average normal at each node which is essentially the average of the normal of the panels sharing the node.
3) Since the Cartesian mesh would sometimes give rise to very small panels while cutting a body, the small panels whose area is less than $1/10^{th}$ of the neighboring panel is merged with the large panel.
4) Generation of the hybrid prism layer by extrusion of the surface panels up to a height that is user specified. Normally extrusion is based on the average normal of a node. However this can also be user defined way of projection for ease of stitching with the outer Cartesian mesh.
5) Stitch the prism layer with the outer Cartesian mesh by joining the prism layer to the nearby outer Cartesian mesh node.
6) Split the last hybrid layer if it is too large as compared to neighbouring prism layer.

Figures 4.11 to 4.14 show the various steps of the hybrid prism layer generation and finally stitched with the outer Cartesian mesh for an axisymmetric geometry. Figure 4.14 shows the final hybrid Cartesian mesh and the magnified region of the nose and flare portion. From the figure it can be seen that there are six types of cells possible after the stitching of hybrid prism layer with the outer Cartesian mesh. The first type of cell is the hybrid prism layer cell whose neighbours are also prism layer cells. The second type of cell is an outer edge cell which is between the prism layer cell and the Cartesian mesh which can have more than one Cartesian cell as its neighbour. The third type of cell is the outer hybrid prism layer hugging the Cartesian mesh which

can have more than one Cartesian mesh as its neighbor. The fourth type of cell is the outer hybrid prism layer that does not hug the Cartesian mesh. The fifth type of cell is a Cartesian cell where all the neighbours are not pure Cartesian cells and finally the sixth type of cell is the one whose all neighbours are Cartesian cells.

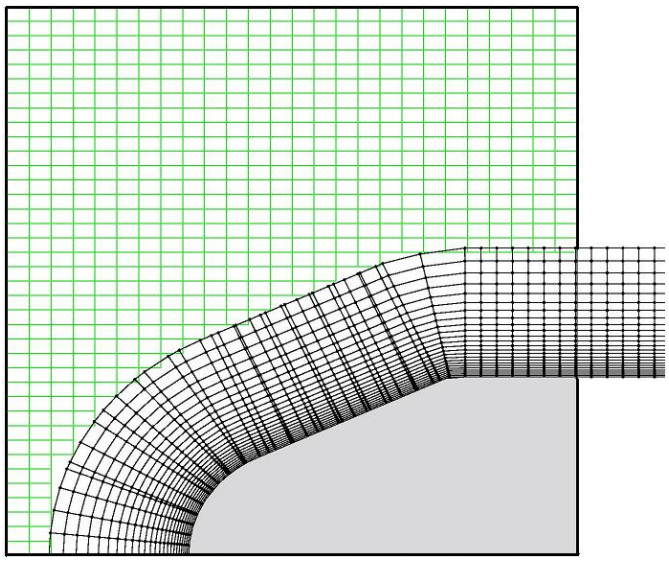


Figure 4.11 Prism layer at sphere cone region without merging of small panels

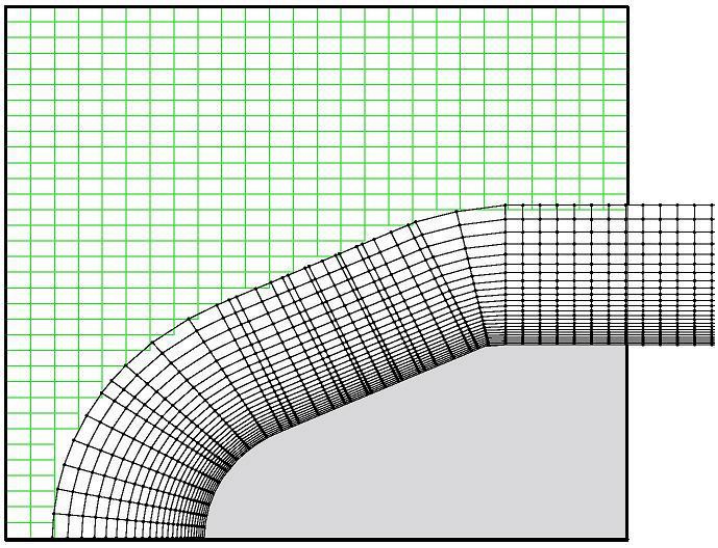


Figure 4.12 Prism layer at sphere cone region with merging of small panels

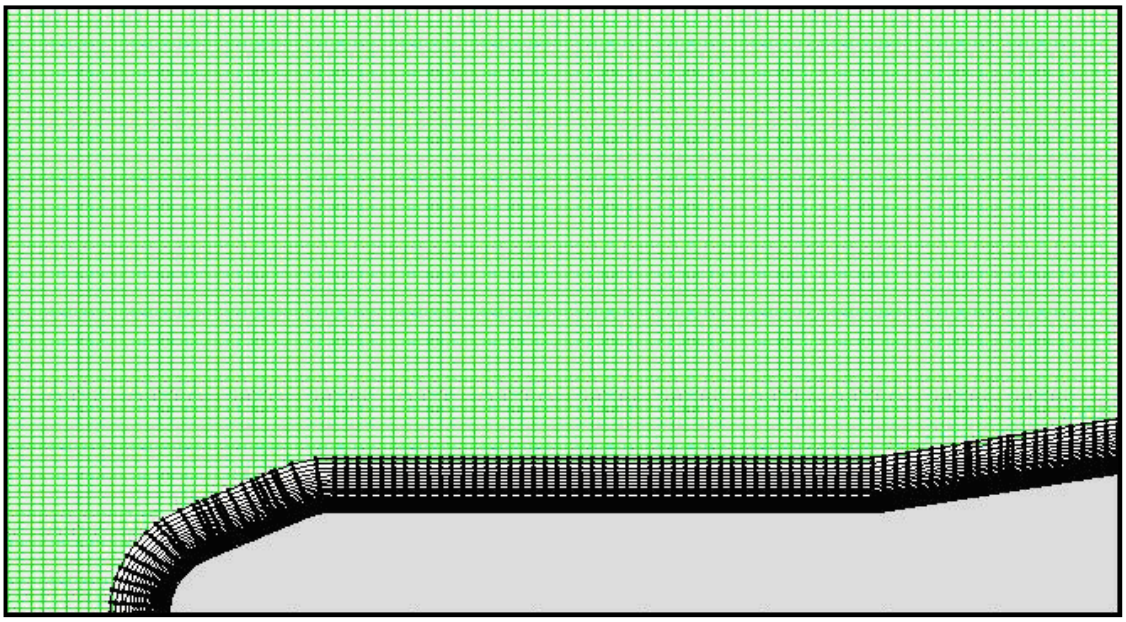


Figure 4.13 Prism layer mesh for the full geometry

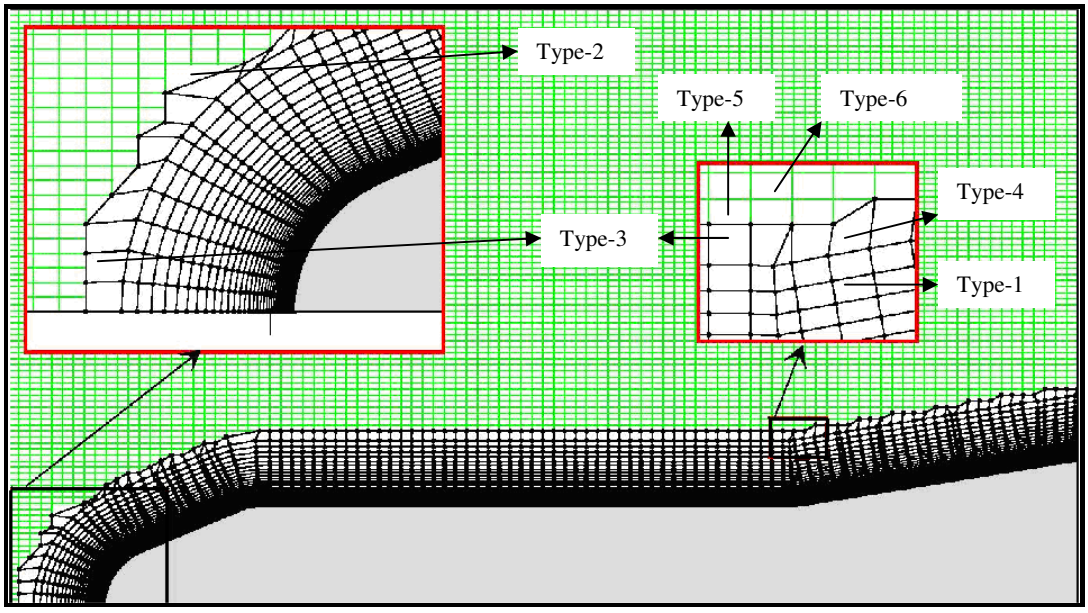


Figure 4.14 Hybrid prism layer stitched with outer Cartesian mesh with six types of cells shown in inset

.

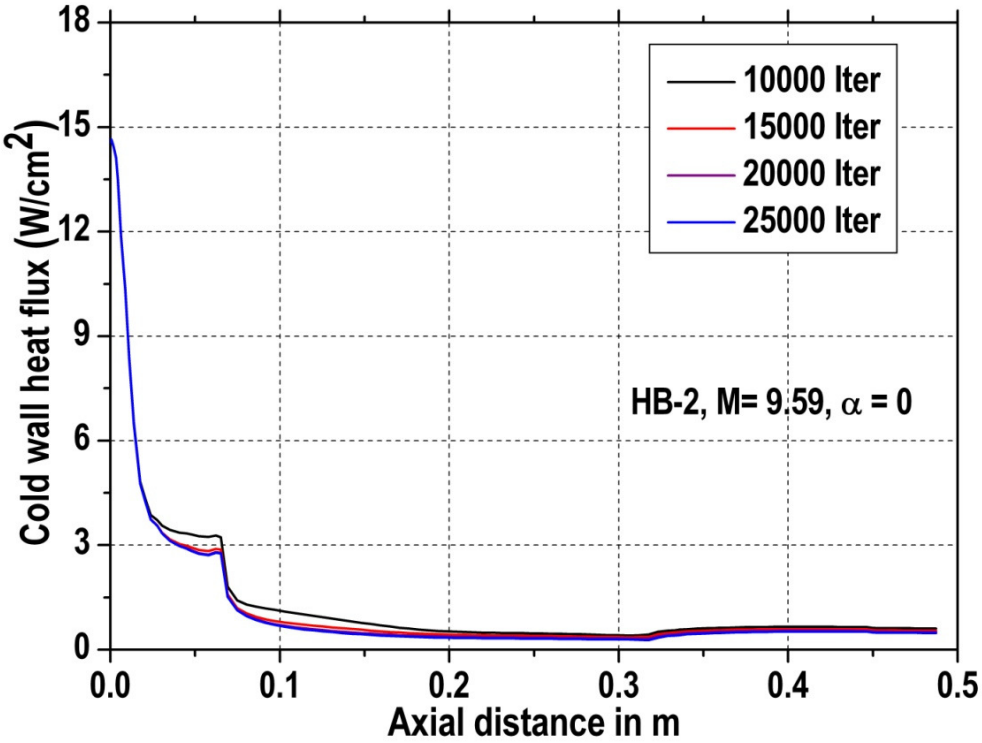


Figure 4.15 Iteration convergence for heat flux

Simulations are carried out for zero angle of attack with the stitched mesh for the conditions as given in Table 4.1. Figure 4.15 shows the wall heat flux along the length of the body for various iterations for 20mm prism layer height with 55 cells in which the first cell height is 5 microns. The iteration convergence is clearly seen after 20000 iterations. Figure 4.16 and 4.17 show the Mach number profiles at two typical stations. The two locations along which the profiles are taken are also shown in the inset. This exercise was carried out to see whether the prism layer height has any influence on the profiles and especially in the region of transition from the unstructured prism layer to the Cartesian mesh. It is seen that the Mach number profiles at the two typical stations are almost same for different prism layer heights. As expected, the boundary layer is thicker at the location 0.335 m from nose as compared to 0.153 m from nose. Figure 4.18 shows the Mach number contour which shows the bow shock, expansion waves and the oblique shock at the cone flare junction.

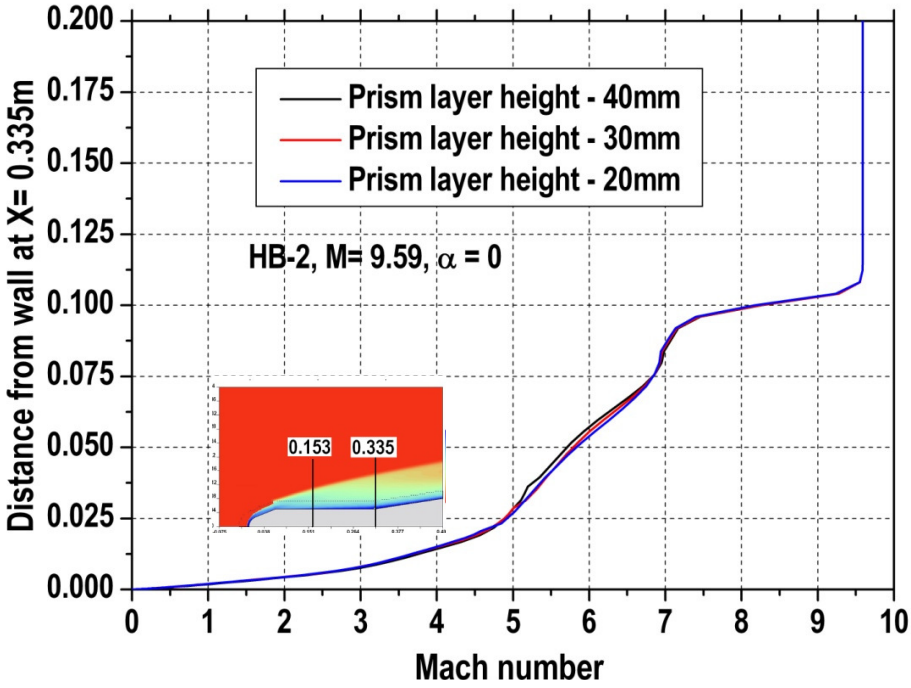


Figure 4.16 Mach number profile at X = 0.335m

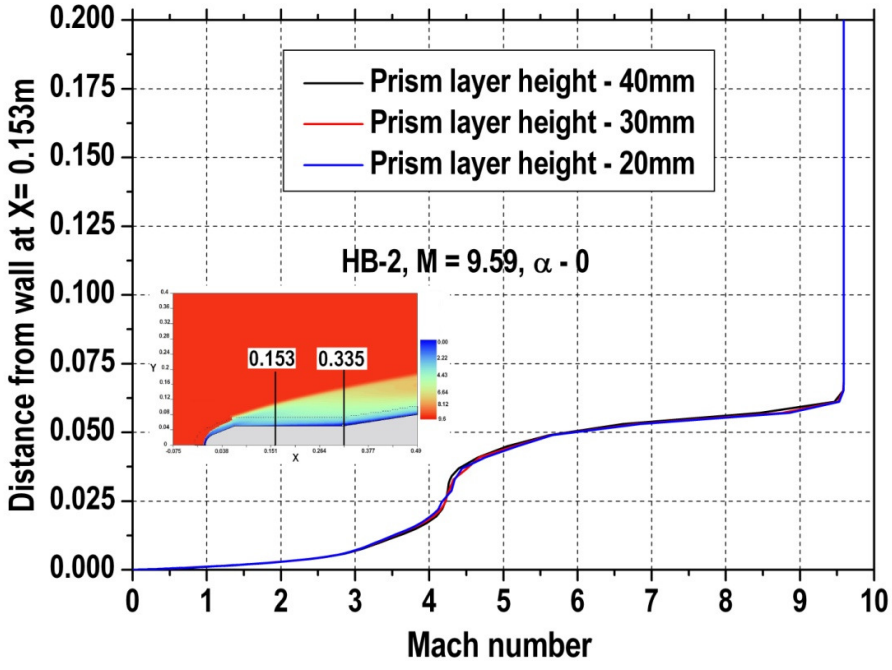


Figure 4.17 Mach number profile at X =0.153 m

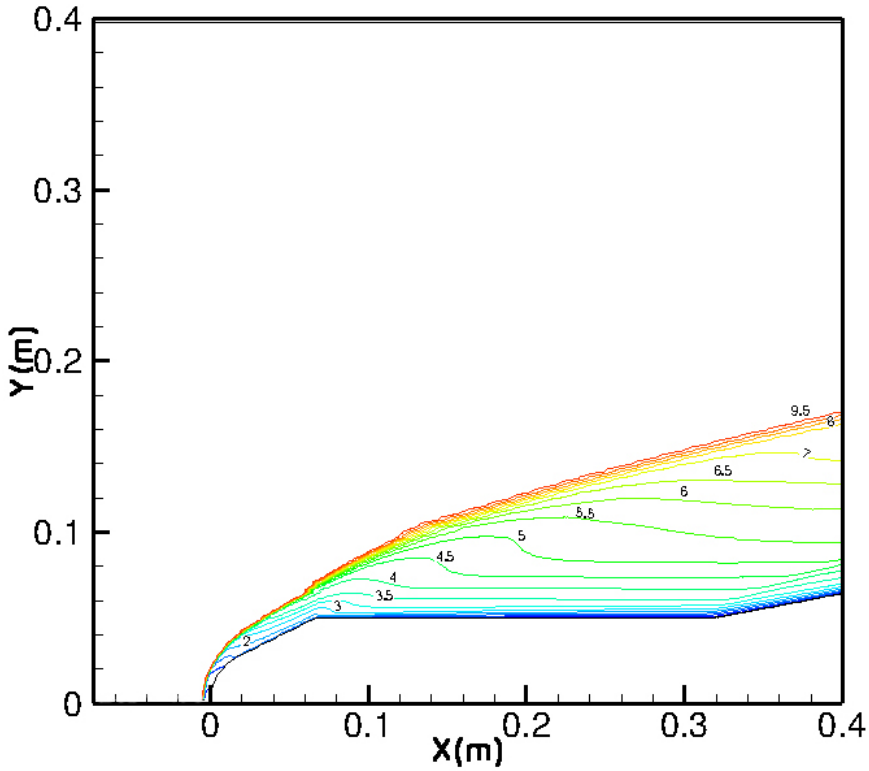


Figure 4.18 Mach number contours over the HB-2 geometry

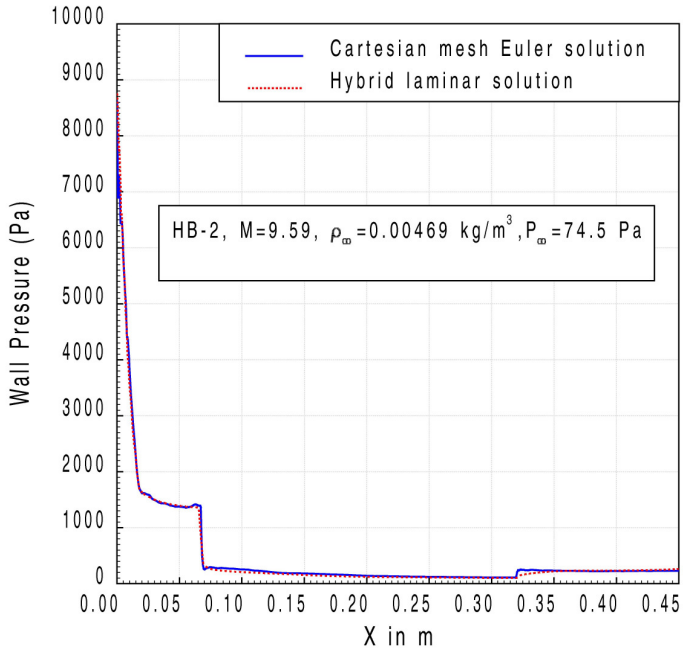


Figure 4.19 Static pressure along the wall

Figure 4.19 shows the static pressure along the geometry for both Cartesian mesh Euler solution as well as the hybrid solution. Since the flow is attached, the pressure is impressed on the boundary layer and hence similar solution is seen for inviscid and viscous solution. The solutions show maximum pressure at stagnation point followed by expansion at the spherical cap and then constant pressure on cone followed by sharp drop in pressure due to expansion at the cone cylinder junction after which the pressure is constant in the cylindrical region. From the cylinder to flare, the inviscid solution shows a sharper jump as compared to laminar solution due the fact that shock wave laminar boundary layer interaction at the cone cylinder junction makes the pressure rise gradual

Figure 4.20 shows the comparison of computed heat flux for the HB-2 geometry with the experimental results of Kuchi-Ishi et al. (2005). The heat flux is maximum at the stagnation point and decreases sharply as the flow expands over the conical region and then remains nearly constant in the cylindrical region with slight increase in the flare region due to compression. The plot shows that the results are clearly grid independent and shows good match with the experimental data. Figure 4.21 shows enlarged view of the wall heat flux computed which are compared with the experimental results. The present methodology, demonstrated for an axi-symmetric geometry to obtain the near wall resolution of a laminar hypersonic flow from a Cartesian mesh based approach, can be considered as the first step before extending to three dimensional geometries.

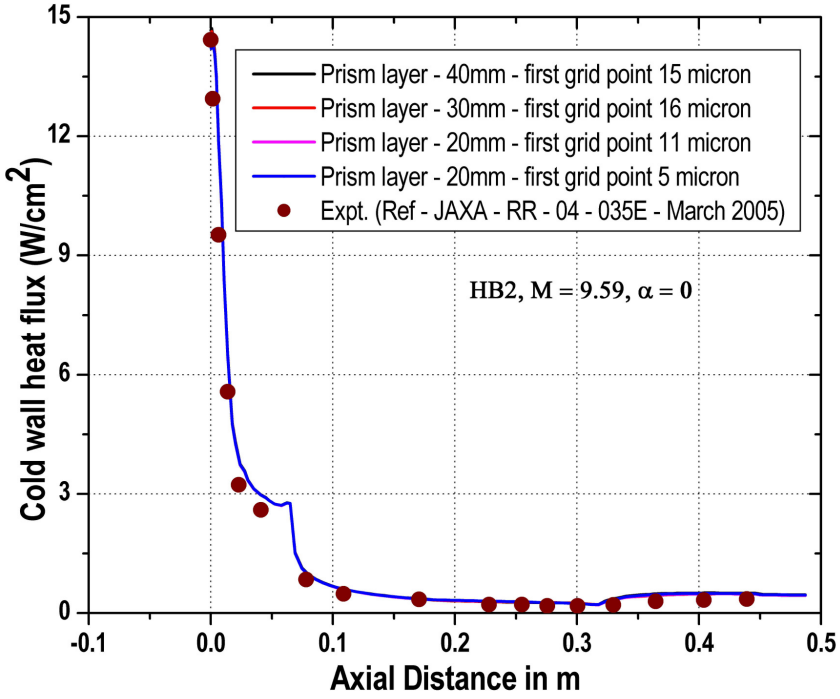


Figure 4.20 Comparison of Heat flux along the wall for different prism layer grids

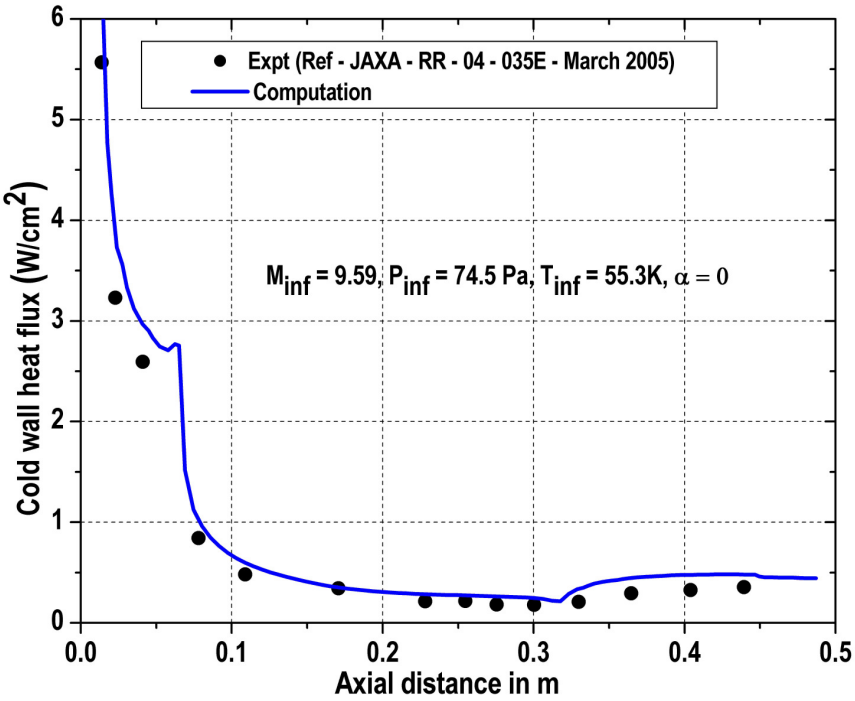


Figure 4.21 Enlarged view of the heat flux along the wall

4.1.4 Heat Flux Estimation for a Typical Bulbous Heat Shield

The present hybrid solution methodology is also applied to a typical bulbous heat shield geometry for which the shock tunnel experimental results conducted by Srinivasa (1991) are available at hypersonic mach numbers. The geometric details are given in Table 4.2 and figure 4.22. The free stream conditions are given in Table 4.3.

Table 4.2 Geometry details of a typical bulbous heat shield

D1 (mm)	Rn/D1	θc (°)	L1/D1	θb (°)	D1/D2	L2/D2	L (mm)
50	0.2188	20	1.4062	15	1.1429	1.5	195

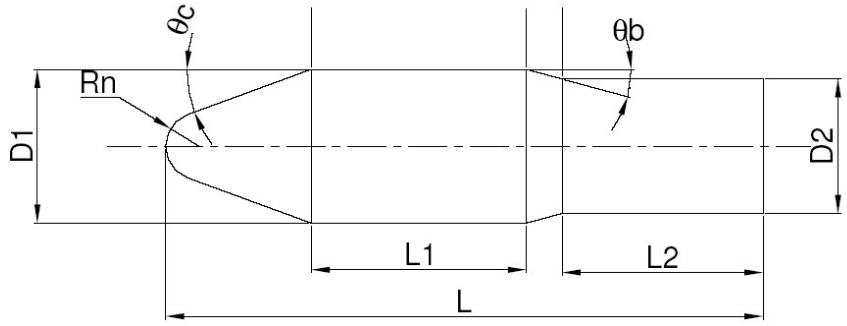


Figure 4.22 Schematic of a typical bulbous heat shield

Table 4.3 Flow conditions of the shock tunnel experiment (Srinivasa [1991])

M_∞	P_∞	ρ_∞	T_0	T_{wall}	R_e based on 50mm diameter	α (Angle of attack)
5.75	1320 Pa	0.019 kg/m^3	1829 K	300 K	1.143x10^5	0

The computations are carried out with hybrid mesh consisting of prism layer height of 20 mm stitched to the outer Cartesian mesh, for laminar flow conditions as in the experiments. The first grid point is of the order of 11 microns with about 45 number of prism layer cells. Figure 4.23 shows the heat flux along the length of the model which gives a reasonable match with the shock tunnel measurements. The maximum heat flux of about 119 W/cm^2 is computed at stagnation point against the experimental measurement of 114 W/cm^2. This is followed by a steep fall due to the expansion and nearly constant in the cone portion followed by a drop in the heat flux

after the cone due to the expansion at the cone cylinder junction and after little downstream of the cylindrical region constant heat flux is observed followed by further small drop in the boat tail region. After the boat tail a slight increase in heat flux is noticed due to compression and remains constant after the pressure recovery on the cylinder.

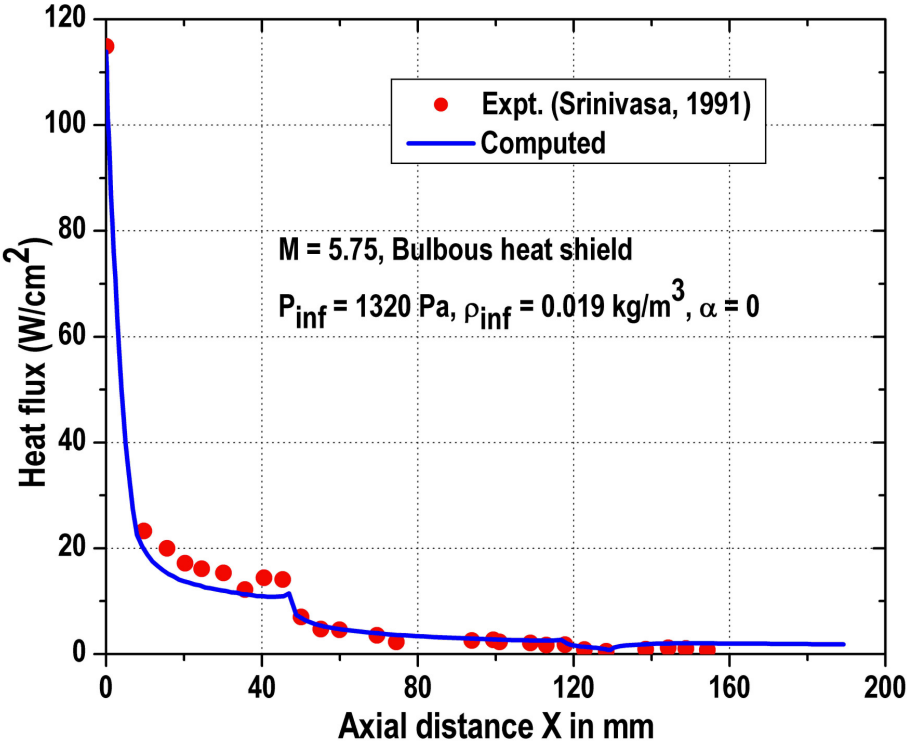


Figure 4.23 Cold wall heat flux along the wall of the bulbous heat shield

4.1.5 Hybrid solution for three dimensional flows

After reasonable validation of the hybrid solution methodology to axi-symmetric flows, the next logical step is to extend it to three dimensional flows. The methodology followed for three dimensional flows is same as described in Section 4.1.2. Initially an Euler solution is obtained over the body with pure Cartesian mesh. In the subsequent step, the prism layers are extruded from the background Cartesian mesh panels up to a certain height and the Cartesian mesh Euler solution obtained in

the first step is mapped to this prism layer. Subsequently, the laminar viscous solution is carried out for this prism layer of cells alone with inviscid Cartesian mesh solution imposed as the outer boundary condition for the prism layer cells. It is to be noted that, in this case, the interaction of the viscous layer would not be considered with the outer inviscid solution and hence will be similar to the boundary layer type of solution. However if the prism layer cells are stitched to the outer Cartesian mesh then this interaction would automatically be taken into account as in the hybrid solution methodology described for axi-symmetric flows in the previous section. The stitching of this prism layer of cells in a three dimensional case with the outer Cartesian mesh is a very involved task and is planned to be taken up as future work. However in many cases, if the prism layer is extended sufficiently to a distance beyond the interaction region, this in itself would give good solution. In order to demonstrate this methodology, a three dimensional flow case for the HB-2 geometry described in Figure 4.5 is chosen for which the free stream conditions are given in Table 4.4 below.

Table 4.4 Free stream conditions for flow over HB-2 geometry at angle of attack from Kuchi-Ishi et al (2005)

P_0 (MPa)	T_0 (K)	M_∞	P_∞ (Pa)	ρ_∞ (kg/m^3)	T_∞ (K)	Re(1X10^5) based on core dia	α (deg)	T_{wall} (K)	U_∞ (m/s)
4.021	1040.7	9.65	114.7	0.00719	55.6	2.85	15	300	1441.9

In the first step, an Euler solution is obtained for the pure Cartesian mesh for the free stream conditions at angle of attack 15 degrees as shown in figure 4.24. In the next step, unstructured prism layer is generated from the Cartesian mesh panels on the body for a distance of 40mm in the normal direction from the wall with 45 numbers of prism layer cells as shown in figure 4.25 and the Euler solution is mapped to the to the unstructured prism layer. Subsequently, solution of laminar Navier Navier-Stokes equation is carried out for the prism layer alone with Euler solution boundary condition imposed for the outer layer of prism cells.

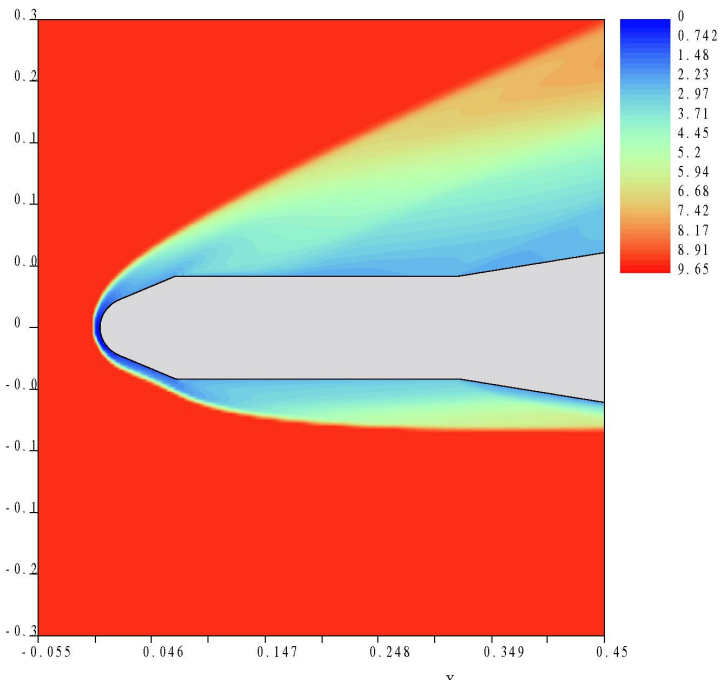


Figure 4.24 Inviscid solution obtained from Cartesian mesh for HB-2 geometry at 15 degree angle of attack

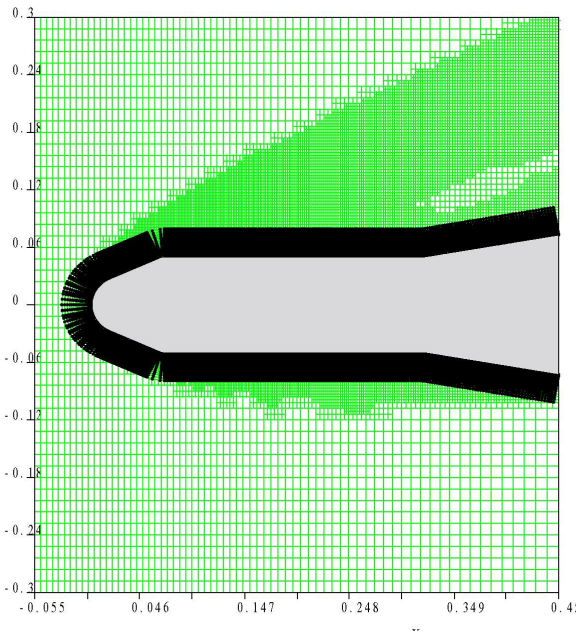


Figure 4.25 Rectangular adaptive Cartesian mesh with extruded prism layer at section z=0.0

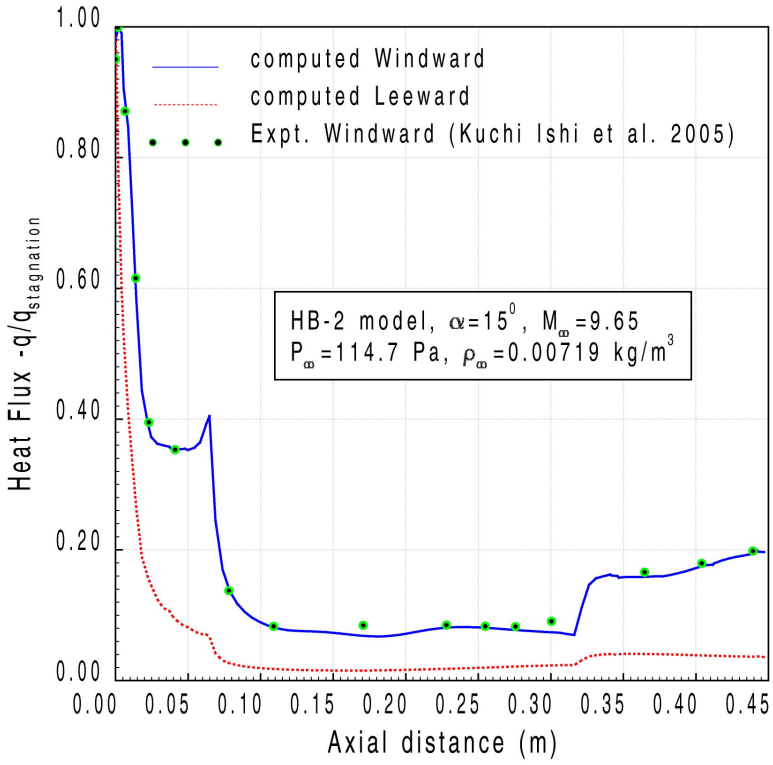


Figure 4.26 Heat flux along HB-2 geometry at 15^0 angle of attack

Figure 4.26 shows the non-dimensional heat flux plotted along the sphere cone cylinder flare geometry. The figure shows that the computed non-dimensional heat flux distribution along the windward side of the body with the hybrid solution methodology shows a reasonable match with the experimentally measured data (available only for windward side) obtained from Kuchi-Ishi et al. (2005). No flow separation was noticed from the velocity vector information. The stagnation point is slightly downstream of the nose cap starting point due to angle of attack effect. The stagnation point heat flux obtained from the present computation is 20.3 W/cm^2 against the experimentally obtained value of 18.23 W/cm^2. Also at the nose cap starting point, the computation shows a higher non dimensional heat flux of 0.98 as compared to the experimentally measured value of 0.95.

4.2 Computation of Laminar Chemically Reacting Hypersonic Flow Using Cartesian Mesh with Near Wall Viscous Resolution

The methodology described in the previous section was extended to the problems of chemically reacting hypersonic flow. Two test cases are described in the following section wherein the comparison is made with results from other CFD codes and limited experimental data. The first test case corresponds to 10 degree wedge at Mach number 25, which is a typical case to validate the diffusion of chemical species in the boundary layer. The second case is the 12.75 mm diameter sphere at Mach number 15 which is a validation test case for the prediction of shock stand-off distance and the temperature along the stagnation streamline. For the shock stand-off distance, the experimental shadowgraph obtained by Lobb (1964) is also available.

4.2.1 Chemically reacting hypersonic flow over a 10 degree wedge

Chemically reacting hypersonic flow over a 10 degree wedge of 3.5 m length at Mach number 25.3 is computed with the following flow conditions as used by Alavilli (1997) who performed computations using the EURANUS (European Aerodynamic Numerical Simulator) code. EURANUS code was developed by Hirsch et al. (1991) and was extended to simulate thermochemical non-equilibrium flows on structured grids by Alavilli (1997). EURANUS is a Multigrid Multiblock numerical flow solver that solves Reynolds averaged Navier Stokes equation based on cell centered approach. The inviscid flux can be either calculated by upwind method or by central differencing with artificial dissipation of Jameson. For this test case the computations from EURANUS were quoted by Alavilli (1997) to be performed with central numerical flux formulation with explicit four stage Runge-Kutta scheme. The convergence acceleration is achieved through local time stepping and Implicit Residual smoothing. The code has the capability to solve both chemical and thermal non-equilibrium with Chemistry models of Park-85,Park-87 and Dunn-Kang models.

Flow is assumed to be in thermal equilibrium and chemical non-equilibrium

Air chemistry model used is 7 species Park 87 models developed by Park (1987)

$p_\infty = 20.3$ Pa , $T_\infty = 253$ K

$V_\infty = 8100$ m/s, Reynolds number = 490000 based on length

$T_{wall} = 1200$ K , Mach number =25.3

Free stream mass fraction of N_2 =0.79

Free stream mass fraction of O_2=0.21

Wall is assumed non-catalytic

Figure 4.27 shows the wedge with the Cartesian mesh and prism layer. A small extension is provided upstream of the wedge to facilitate the implementation of boundary conditions. Mirror wall symmetry is imposed on this extension. Supersonic inflow conditions are imposed on the left boundary and supersonic outflow conditions at the top and right boundaries. In the small extension region in the front of the wedge, symmetry boundary conditions are imposed and isothermal, non-catalytic wall is imposed on the wedge wall. The 7 species considered are $N_2, O_2, NO, O, N, NO^+, e$ and Species under-relaxation of Palmer as described in Section 3.6 is used with under-relaxation parameter of 0.001 which will prevent the run-away of chemical reactions. For a meaningful code to code comparison, Schmidt number of 0.5 is used for the calculation which is the same value used by Alavilli (1997) for this test case. A 70X40 pure Cartesian mesh is generated and prism layer is extruded from the background Cartesian mesh and stitched with the outer Cartesian mesh as shown in figure 4.27. Figure 4.28 shows the temperature profile at the exit of the wedge for different number of prism layer cells. The prism layer height is 25 cm with first grid point of about 0.2 mm for 101 number of prism layer cells and 0.3mm for 36 number of prism layer cells. It can be seen that although the thermal boundary layer is same for all the numbers of prism layer cells, the wedge shock is more sharply captured with larger number of prism layer cells. Hence 101 number of prism layer cells is chosen for computations. Since grid adaptation capability for prism layer cells is not present in the existing code, the number of cells in the prism

layer had to be increased in an offline fashion and separate computations were carried out. Figure 4.29 shows the convergence of the thermal boundary layer at the exit of the wedge with iterations. It can be seen that by 20000 iterations the solution is almost converged. The CFL number used for the fully explicit computations is 0.5. Figure.4.30 shows the temperature profiles at the exit section of the wedge for perfect gas and real gas compared with the results of EURANUS (European Aerodynamic Numerical Simulator) code. As expected, the real gas temperature is lower in comparison to perfect gas temperature because of the endothermic chemical reactions caused by dissociation of molecular Oxygen and Nitrogen making the real gas cooler. Good agreement with the EURANUS computation by Alavilli (1997) is seen which uses a numerical scheme of central differencing with artificial dissipation. The shock captured by the present code is sharper because of more number of grids used as compared to the EURANUS code which used 65X65 structured mesh with almost same domain as in the present computation.

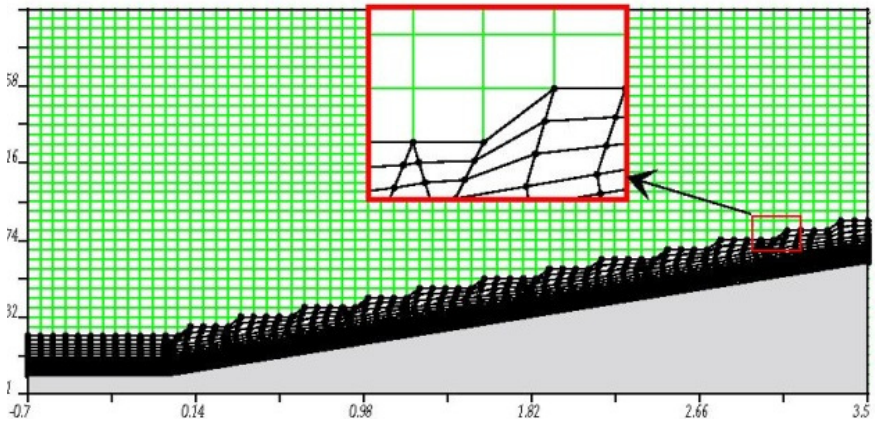


Figure 4.27 10 degree wedge with hybrid prism layer stitched with outer Cartesian mesh

However, there is a small over-shoot of temperature at the shock location for the present computation possibly because of the effect of numerical scheme and the grid. Since the peak boundary layer temperature is quite large of the order of 5500 K which is greater than the temperature required for start of dissociation for Oxygen and Nitrogen, atomic Oxygen, atomic Nitrogen and Nitric oxide are formed. Figure

4.31 shows the mass fraction of Nitric Oxide and figure 4.32 shows the atomic Oxygen at the exit section of the wedge. It is to be noted that, although an isothermal wall temperature of only 1200 K is imposed on the wall, the presence of atomic oxygen and Nitric oxide is seen. This is caused due to the diffusion of Nitric oxide from the hotter regions of the boundary layer towards the wall. The comparison of species mass fraction profile at the wedge exit section with that of the results of EURANUS code by Alavilli(1997) is good. It is to be noted that species mass fraction show a smooth variation from hybrid prism layer zone to the Cartesian mesh zone

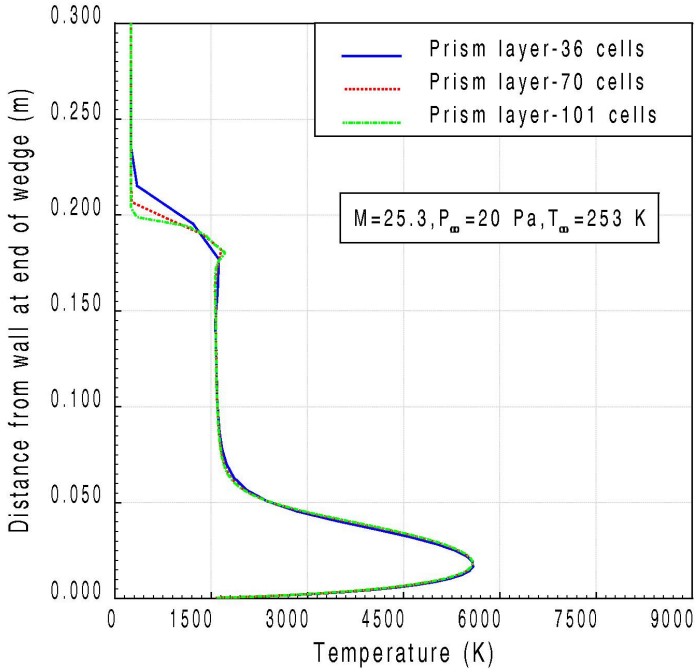


Figure 4.28 Temperature profile at the exit section of wedge for different number of cells in prism layer

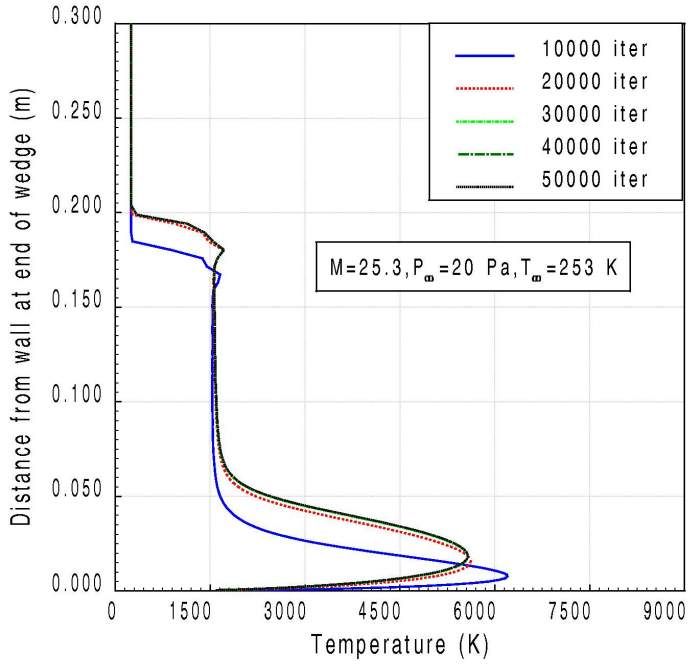


Figure 4.29 Convergence plot of temperature profile at the exit section of wedge

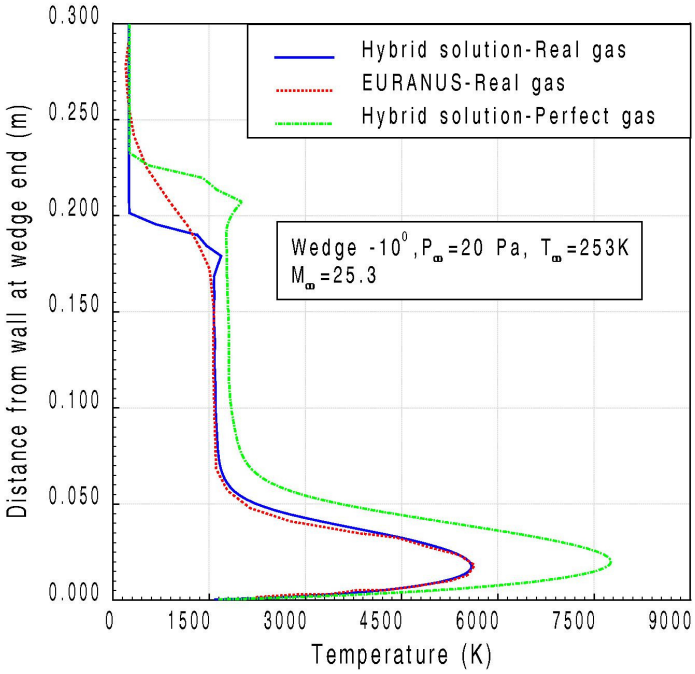


Figure 4.30 Temperature profile at the exit section of wedge compared with the results of EURANUS code from Alavilli (1997)

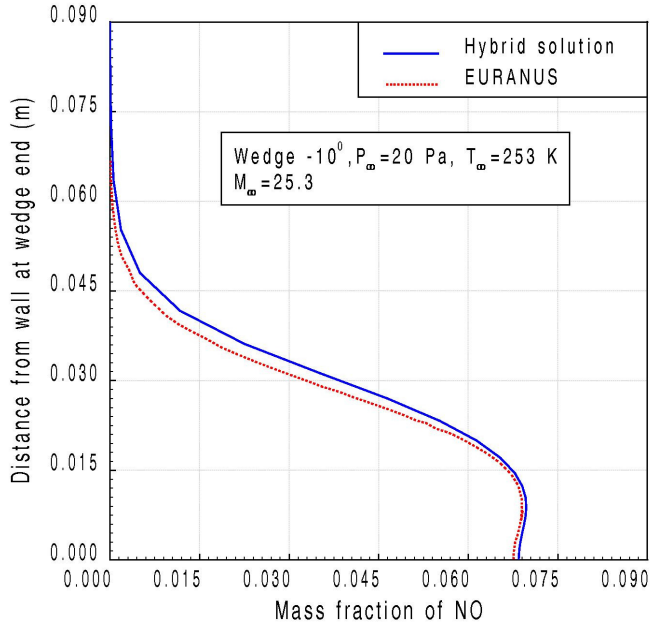


Figure 4.31 Comparison of Nitric oxide mass fraction profile at the exit section of wedge with EURANUS results from Alavilli (1997)

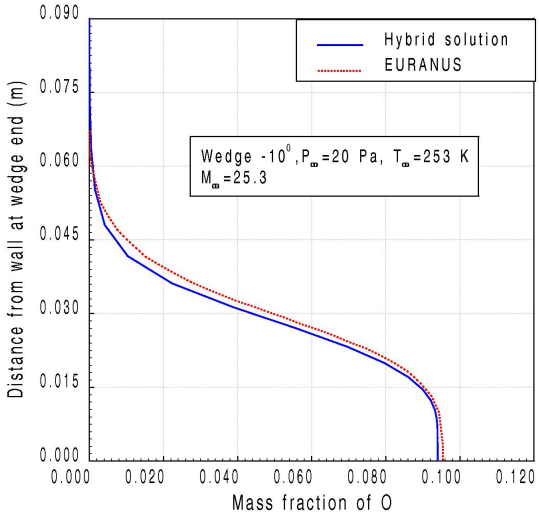


Figure 4.32 Comparison of atomic oxygen mass fraction profile at the wedge exit with EURANUS results from Alavilli (1997)

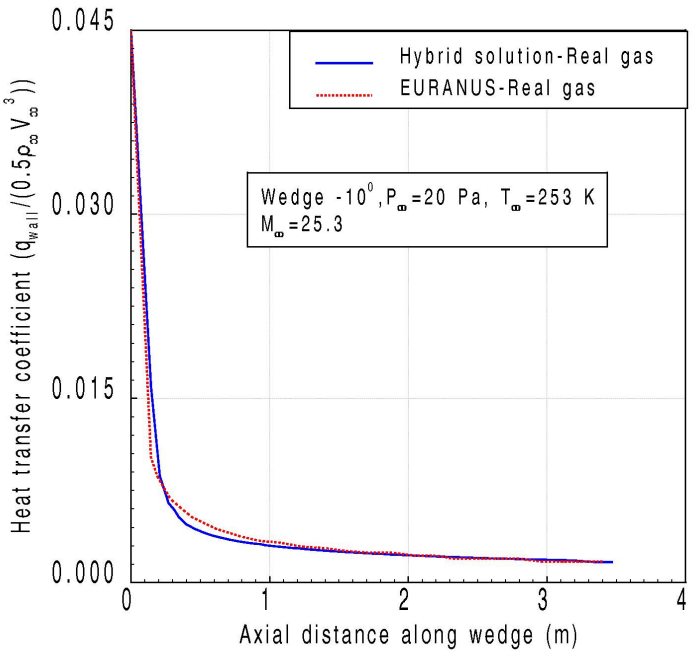


Figure 4.33 Comparison of heat transfer coefficient along the wedge with EURANUS results from Alavilli (1997)

Figure 4.33 shows the non-dimensional heat transfer coefficient distribution along the wall compared with the results of EURANUS code. The heat flux is highest at the leading edge because of the shock being very close to the surface. A good match is observed in the heat flux also with the EURANUS code as shown in the figure.

4.2.2 Chemically reacting hypersonic flow over Lobb sphere

Understanding the flow phenomenon at high Mach numbers for blunt bodies is very essential as most of the space recovery capsules that reenter the atmosphere have such blunt body shapes so as to reduce the heat flux on the body as well as to have higher drag to reduce the velocity. The heat flux on the blunt body is chiefly governed by the shock stand-off distance, the shock strength and the chemical reactions that occur behind the shock. To understand the shock shapes ahead of the spherical blunt bodies, Lobb (1964) carried out a series of experiments on spheres of various diameters and made shock shape measurements. One such experimental case is chosen for validation of the present approach and comparison made for

experimental shock stand-off distance and with other CFD results available in literature.

Following are the flow conditions for 12.7 mm diameter Lobb sphere

Free stream Mach number=15.3

$p_\infty = 664$ Pa $\qquad T_\infty = 293$ K

$V_\infty = 5280$ m/s, Reynolds number = 14600 based on radius

$T_{wall} = 1000$ K

Free stream mass fraction of molecular Nitrogen is 0.79 and Oxygen is 0.21.

Figure 4.34 shows the hybrid grid for the Lobb sphere with 2mm prism layer height with 40 prism layers and with a first grid point of 11 microns. Chemistry model of Park-87 is used and diagonal point implicit scheme is used to calculate the species production rates of species in thermal equilibrium and chemical non-equilibrium and a Schmidt number of 0.5 is used. The chemistry model and the Schmidt number values are the same as that used by Alavilli (1997) which will provide code to code comparison. At the inflow boundary, supersonic inflow condition is used and at the top and right boundaries supersonic outflow conditions are used and at the symmetry plane, symmetry boundary condition is used. .

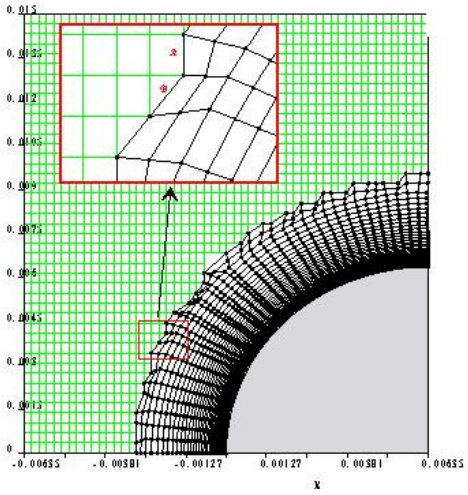


Figure 4.34 Hybrid mesh for Lobb sphere

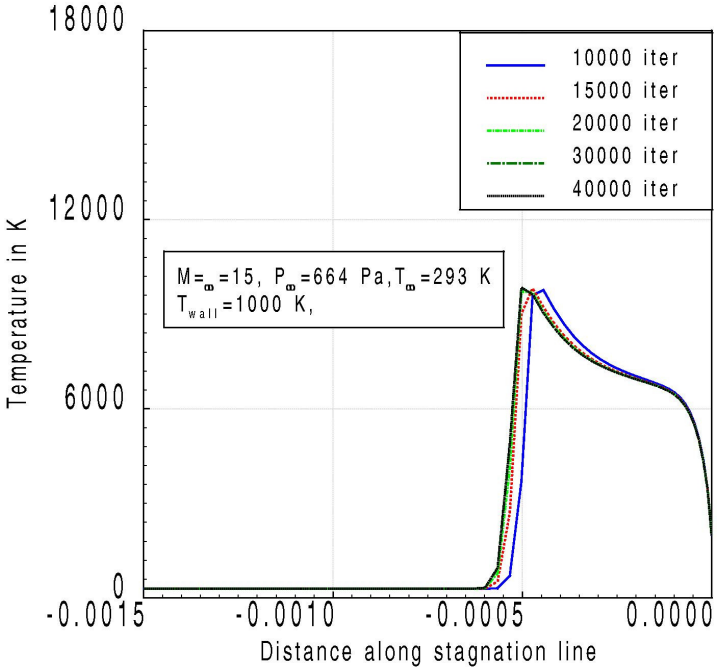


Figure 4.35 Convergence plot of temperature along stagnation line for Lobb sphere

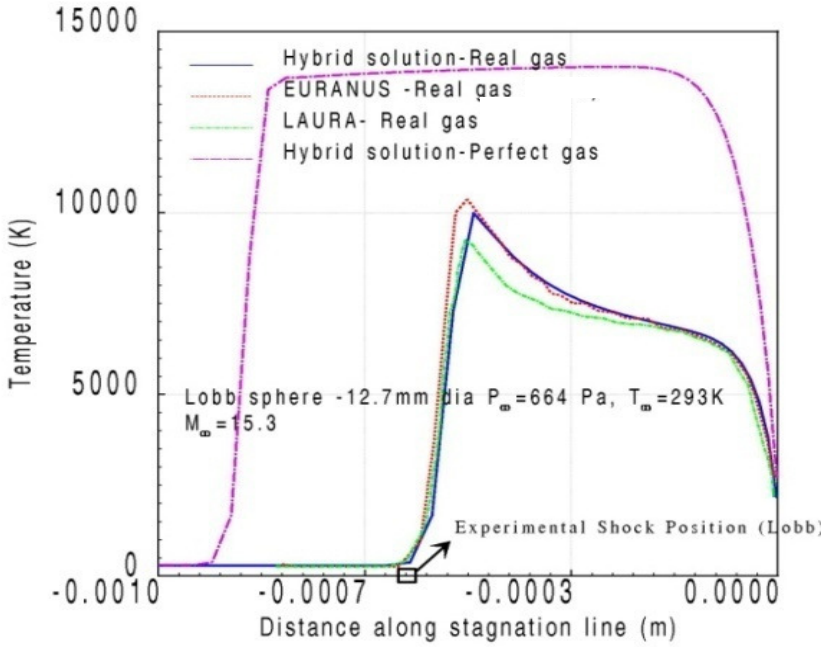


Fig.4.36 Temperature along the stagnation stream line for Lobb sphere

Figure 4.35 shows the convergence plot of temperature along the stagnation line which is of interest for this problem and it is seen that by 20000 iterations the solution has reached convergence. Figure 4.36 shows the temperature along the stagnation stream line. The shock stand-off distance for perfect gas simulation is much more than real gas. This is because, in the case of high temperature real gas effects, the dissociation of molecular nitrogen and molecular oxygen reduces the temperature of the gas, which in turn increases the density of the mixture of species. Owing to the increased density, more mass can be pushed through the stream tube resulting in forward movement of the shock to satisfy the continuity equation. Also the prediction of shock stand-off distance from EURANUS code by Alavilli (1997), from LAURA code by Gnoffo (1989) and from the present code are almost same, although there is some variation in the peak temperature predicted using LAURA. The EURANUS code uses the central differencing with artificial dissipation numerical scheme and LAURA employs upwind biased point implicit line relaxation algorithm, and one of the reasons for these differences in peak temperature between various solutions could be attributed to the differences in numerical schemes. The predicted shock position is quite close to the experimentally measured value by Lobb (1964).

From the above validation cases, it can be concluded that Cartesian mesh based hybrid approach for near wall viscous resolution can be used to compute non-equilibrium chemically reacting hypersonic flows.

4.3 Computation of High Speed Flows with Combustion

While high speed chemically reacting flow encountered during reentry of space vehicles from outer space involves mainly endothermic reactions because of dissociation reactions, the chemically reacting flow in air-breathing propulsion is exothermic in nature. A comparative study will show that, the computations of flows involving endothermic reactions are less problematic to deal with as compared to strong exothermic reactions like that of Hydrogen-Oxygen combustion. This is

because, in the case of an endothermic reaction during reentry, the temperature falls after dissociation process which in turn lowers the dissociation reaction rates and eventually leading to stabilization of the system. On the other hand, for exothermic reaction, the abrupt release of energy would lead to sharp rise in temperature which will further increase the exothermic reaction rates leading to run-away conditions if not properly handled. Flow field in a Scramjet engine involves supersonic combustion of Hydrogen-Air in the combustion chamber and the prediction of ignition delay in the Hydrogen-Oxygen combustion systems is an important aspect of the combustion prediction process. In order to validate the prediction of ignition delay occurring during Hydrogen-Oxygen combustion process, a shock-induced combustion test case, for which the experiments carried out by Lehr (1972) are available, is chosen.

4.3.1 Prediction of shock-induced combustion for Lehr cylinder

Lehr (1972) performed experimental studies with 15 mm diameter hemisphere-cylinder fired through stoichiometric Hydrogen-Oxygen and Hydrogen-air mixtures at sub and super detonative speeds which involves shock-induced combustion kinetics. Computations are carried out with Cartesian mesh with a hybrid prism layer for the following experimental flow conditions of Lehr (1972);

Free stream Mach number =3.55

Free stream pressure = 186 Torr

Free stream temperature = 292 K

Free stream species mass fraction of Hydrogen = 0.1112

Free stream species mass fraction of Oxygen =0.8888

Angle of attack = 0 degree

Flow is considered laminar and in chemical non-equilibrium and the wall is treated as adiabatic and non-catalytic.

The above conditions pertain to sub-detonative speed and the flow phenomenon is convection dominated. The measured detonation speed of stoichiometric $H_2 - O_2$ is 2550 m/s. 7-species 7-reaction model of ONERA as given in Table 2.7 of Section

2.6 is used. The chemical species considered are $H_2, O_2, H_2O, OH, H, O, N_2$ for the finite rate chemical reactions. Point-Implicit scheme as described in Section 3.5 is used to overcome the stiffness of species conservation equations. The boundary condition on the left is the supersonic inflow condition and at the symmetry plane, symmetry boundary condition is applied. At all other boundaries, supersonic outflow condition is applied and fully explicit scheme with CFL of 0.1 is used. The solution converged in about 10000 iterations as seen in figure 4.37. The adiabatic wall conditions are expected to converge faster than the isothermal wall conditions, since thermal boundary layer formation is not needed. Figure 4.38 shows the temperature plot over the Lehr cylinder which shows a temperature rise at the shock front and another temperature rise at the combustion front. Although immediately after the shock, the temperature is beyond the ignition temperature of $H_2 - O_2$ mixture, a finite time is needed for the ignition to occur as there are several elementary steps for the reactions to complete and form water vapour. These are the chain initiation, chain propagation, chain branching and chain termination reactions. By the time these reactions occur, the flow would have reached very near the spherical wall. Figure 4.39 shows the temperature along the stagnation stream line. The shock position and the combustion initiation position can be clearly seen. The distance between the two fronts is the incubation length caused due to ignition delay which is also clearly visible in the experimental shadowgraph from Lehr (1972) shown in the inset. Computations show a good match with the experimentally observed shock and combustion front location. Although the post shock temperature is high enough to cause reactions, the non-equilibrium effects subdue effects in the incubation period. Figure 4.40 shows the water vapour mass fraction showing the region of combustion and figure 4.41 shows the mass fraction of various species along the stagnation stream line. Through the above validation case good confidence is obtained for computation of non-equilibrium Hydrogen-Oxygen combustion.

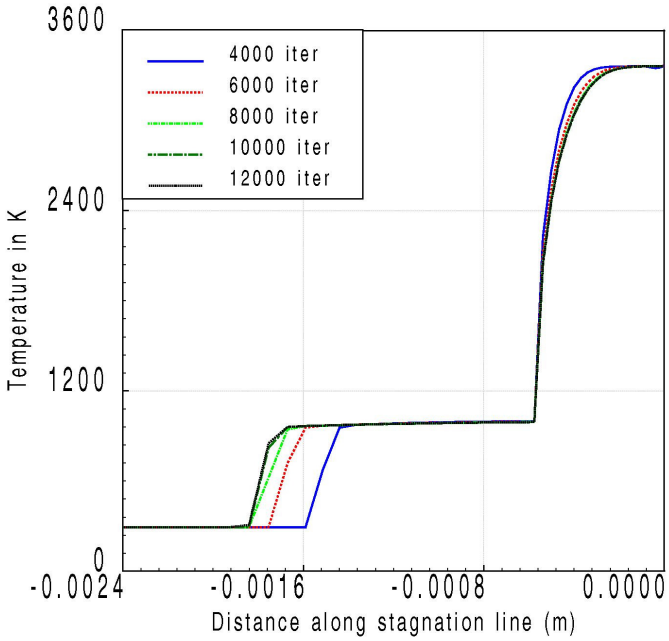


Figure 4.37 Convergence plot of temperature along stagnation line for Lehr cylinder

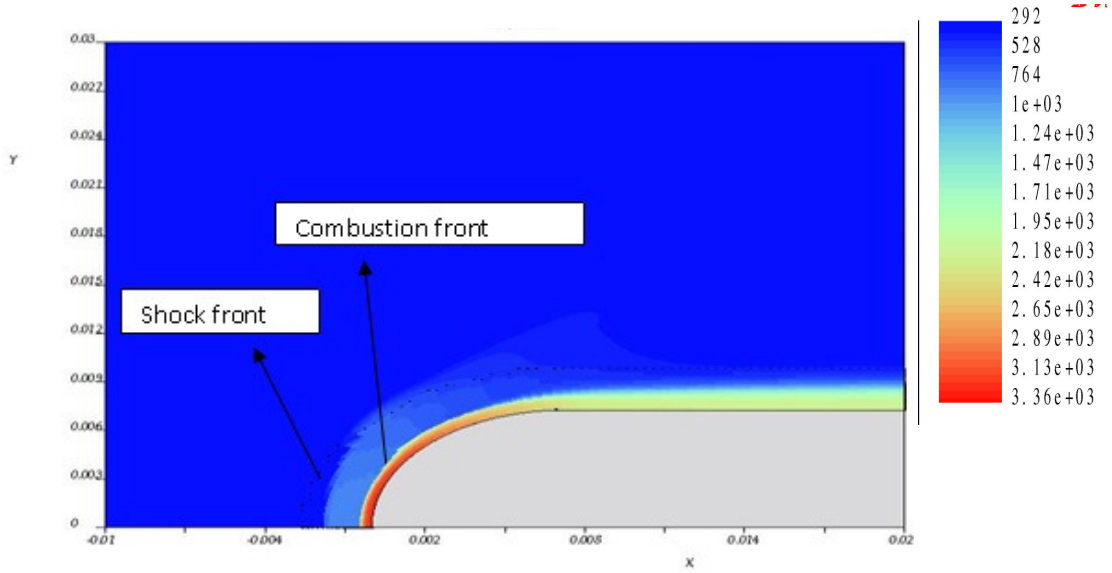


Figure 4.38 Temperature plot for Lehr cylinder at M=3.55 in stoichiometric mixture of Hydrogen-Oxygen

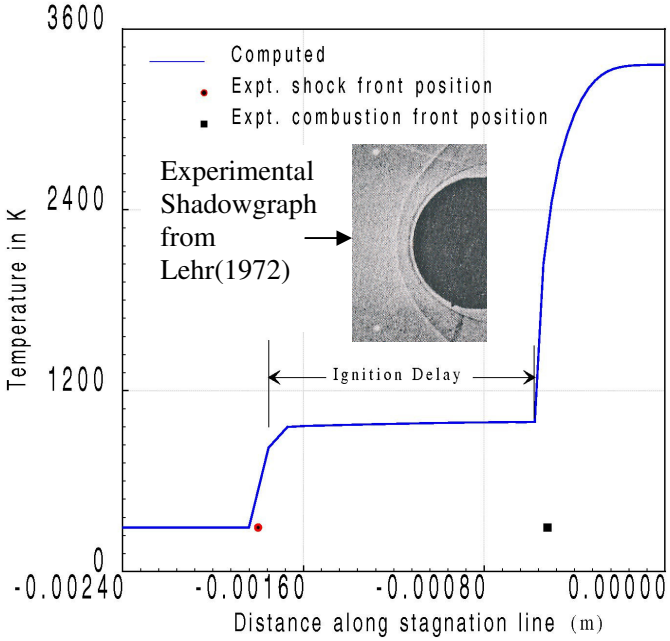


Figure 4.39 Temperature along the stagnation stream line computed for Lehr cylinder at M=3.55 along with positions of shock and combustion front from experiments by Lehr (1972)

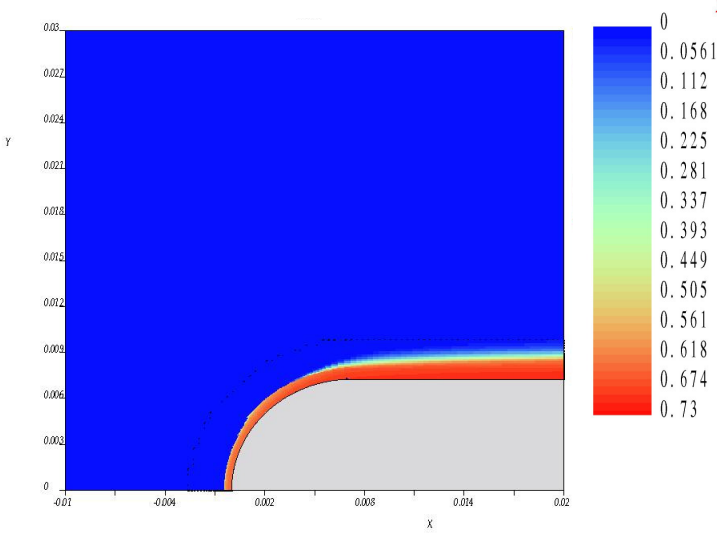


Figure 4.40 Water vapour mass fraction plot for Lehr cylinder at M=3.55

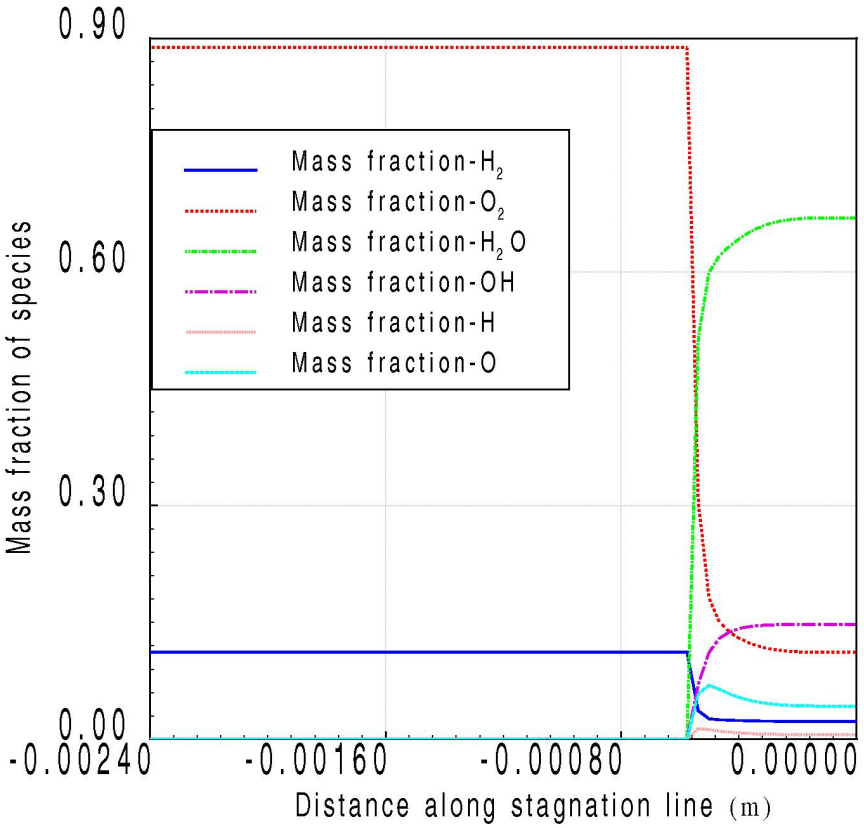


Figure 4.41 Mass fractions of various species along the stagnation stream line for Lehr cylinder at Mach number 3.55

4.4 Computation of High Speed Turbulent Flows with Combustion for Scramjet Engines with Cartesian Mesh

Computation of high speed turbulent flows with combustion in Scramjet engines is extremely challenging because of the complex geometries and flows involved. Cartesian mesh has considerable advantage in handling the complex geometry and hence developing a capability to solve such flows on a Cartesian mesh has a huge advantage from the industrial perspective in terms large reduction in turn around time from geometry to solution. Also since such flows are convection dominated, the Cartesian mesh would be able to capture the flow features with a good adaptive mesh that can capture the strong gradients arising from shocks and shear layers. It is to be noted that, in most of the cases the interest in such high speed flows

as applied to Scramjet engines is to obtain the thrust delivered by the engine and hence it is not always necessary to get near wall quantities like heat flux directly from computation. Under such circumstances, the Cartesian mesh with suitable wall functions can make good estimates of pressures obtained from high speed turbulent combustible flows. Hagemann et al. (1996) has carried out Cartesian mesh perfect gas computations on 3D-Plug-Cluster nozzle configurations to obtain the pressure distribution in the nozzle using a modified wall function approach for $\kappa-\varepsilon$ turbulence model adapted to Cartesian mesh and has shown good comparison with experiments. Same approach that is followed for the existing turbulent perfect gas solver is also used here for computation of high speed turbulent flows with combustion and is explained below.

4.4.1 Modified wall function approach for $\kappa-\varepsilon$ turbulence model with Cartesian mesh

The attached flow turbulent boundary layer at the wall has three distinct regions, namely, laminar sub-layer, log layer and the defect layer. In order to describe the nature of the flow near the wall, it will be useful to define the following non-dimensional velocity and distance.

Non-dimensional wall velocity = $u/u_\tau = u^+$ (4.1)

Where u_τ is the friction velocity = $\sqrt{\frac{\tau_{wall}}{\rho}}$ (4.2)

Non-dimensional wall distance = $\frac{yu_\tau}{\nu} = y^+$. (4.3)

where y is the distance of first grid point from the wall and ν is the kinematic viscosity of the wall cell.

The layer closest to the wall is the laminar sub-layer region wherein the non-dimensional velocity ($u^+ = u/u_\tau$) varies linearly with non-dimensional wall distance y^+ i.e. $u^+ = y^+$ and this expression is usually used up to a non-dimensional wall distance of 10. In this region, the turbulent stresses are very small and the flow is dominated by molecular viscous stresses. Beyond the laminar sub-layer, there is a

region of the flow wherein the inertial terms are quite small and yet it is sufficiently far off to have molecular viscous stresses very small compared to turbulent stresses. This region is called the log-layer and usually applied for region of y^+ between 10 and 1000. In this region a well known logarithmic relationship exists between non-dimensional velocity and non-dimensional wall distance as given below

$$u^+ = \frac{1}{\eta}\ln y^+ + B \quad \text{where } \eta = 0.41\text{=Karman constant and B=5.0} \tag{4.4}$$

While solving the Reynolds-Averaged Navier-Stokes equations (RANS) with $\kappa - \varepsilon$ turbulence equations on very fine mesh near the wall, it is found that the standard $\kappa - \varepsilon$ model makes the convergence of the numerical solution very slow due to the stiffness of the equations. Hence wall functions like the one described above are utilized. In this procedure, the flow structure between the wall and the first grid point is assumed to be similar to a boundary layer flow which would have the same velocity as the velocity of the first grid point. Knowing the velocity u_1 at the first grid point y_1 from the wall, Equation (4.4) can be solved iteratively to obtain the friction velocity u_τ. Once the friction velocity is obtained, the non-dimensional wall distance can be calculated. If the calculated wall distance exceeds 1000 then the mesh need to be refined i.e. the first grid point distance has to be made smaller for the law of the wall to be valid. Once the wall shear stress is obtained for y^+ values less than 1000, the value of kinetic energy of turbulence, κ and the turbulent kinetic energy dissipation rate ε are obtained at the first grid point using the following expression based on general experimental observation of turbulence equilibrium in the near wall region, where production of turbulence is equal to dissipation

$$\kappa_1 = \frac{u_\tau^2}{C_\mu^{1/2}} \text{ and } \varepsilon_1 = \frac{u_\tau^3}{\eta y_1} \tag{4.5}$$

The above methodology can be applied to grids that have a constant wall normal distance to the first mesh, which needs some sort of a structured mesh arrangement near the wall. Hence the above approach cannot be directly applied to a Cartesian

mesh which is locally adapted to the body and does not have a constant wall normal distance throughout the flow field. Hence a modified wall function approach, as described by Hagemann et al. (1996), and which is available in the existing Cartesian mesh perfect gas turbulent flow solver is also used for the present computation of turbulent flows with combustion. A brief description of the modified wall function is given below.

In the modified wall function approach, a semi-empirical model to evaluate the influence of boundary layer on the main flow is applied without resolving the main structures and exact velocity profiles of the boundary layer. The flow quantities for the turbulent kinetic energy κ and its dissipation ε are specified at the wall, instead of being specified at a fixed wall distance y^+ away from the wall. The wall shear stress is estimated by the expression

$$\tau_{wall} = -c_f \rho u_i u_i \tag{4.6}$$

Where u_i is the velocity of the cell adjacent to the wall i.e. partial cell and is in principle equal to the slip condition for Euler solutions, ρ is the density of the partial cell and c_f is an equivalent skin friction coefficient which varies between 0.003 and 0.03. Based on the application problems on high speed internal flow in nozzles by Hagemann et al. (1996) it is found that a value of 0.003 gives good comparison with experiments and the same value is used for the present Scramjet computations. Boundary conditions for the momentum equations are specified in terms of fluxes and not of velocities for the partial or wall cells. Skin friction effect is taken in to account by way of subtraction of wall shear stress effect calculated by Equation (4.6) from the momentum equations. As for the turbulent quantities, the Dirichlet conditions is set for the dissipation at the wall as given below by Hagemann et al. (1996)

$$\varepsilon_{wall} = \frac{C_\mu \rho \kappa^2}{y_p B \mu_l} \quad \text{Where } y_p = 80 \text{ and B=0.42} \tag{4.7}$$

For the turbulent kinetic energy Neumann condition is applied. The modified wall function approach has been applied to several turbulent perfect gas flows on

Cartesian mesh and has given good results and reported by Chakraborty et al. (2003), Manokaran et al.(2003), and Singh et al. (2009).

4.4.2 Computation of turbulent flow with combustion for a typical Scramjet combustor

Computation of non-equilibrium chemically reacting turbulent flow with combustion was carried out for a typical Scramjet combustor in connected pipe mode which has strut based injection. For this connected pipe mode ground test, experimental results are already available and mentioned by Gnanasekar et al (2009).

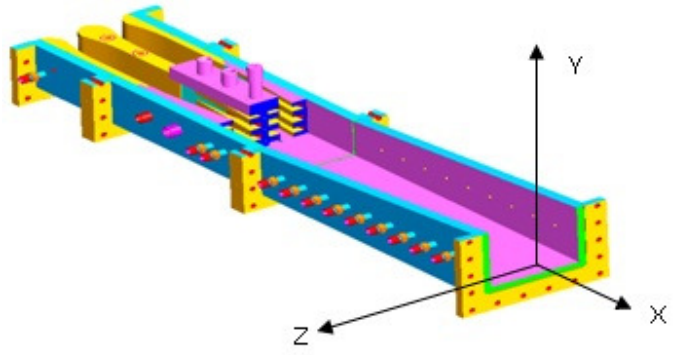


Figure 4.42 Typical Scramjet combustor with strut based injection

Struts provide fuel injection and aid the mixing of Hydrogen with air. Since in the ground test, the stagnation temperature conditions were achieved by burning ethanol the air entering the combustor was vitiated with Carbon-dioxide and water vapour. Computations were carried out for the connected pipe mode test conditions with vitiated air and results of pressure distribution along the wall for an air fuel equivalence ratio is compared with experimental results. In the connected pipe mode test conditions, the facility nozzle expands the working medium to conditions expected at the entry to the combustion chamber of the engine and the flow discharged from the nozzle enters the combustion model directly. In this testing procedure, the engine inlet and nozzle are absent and hence their influence, especially that of the air intake is absent.

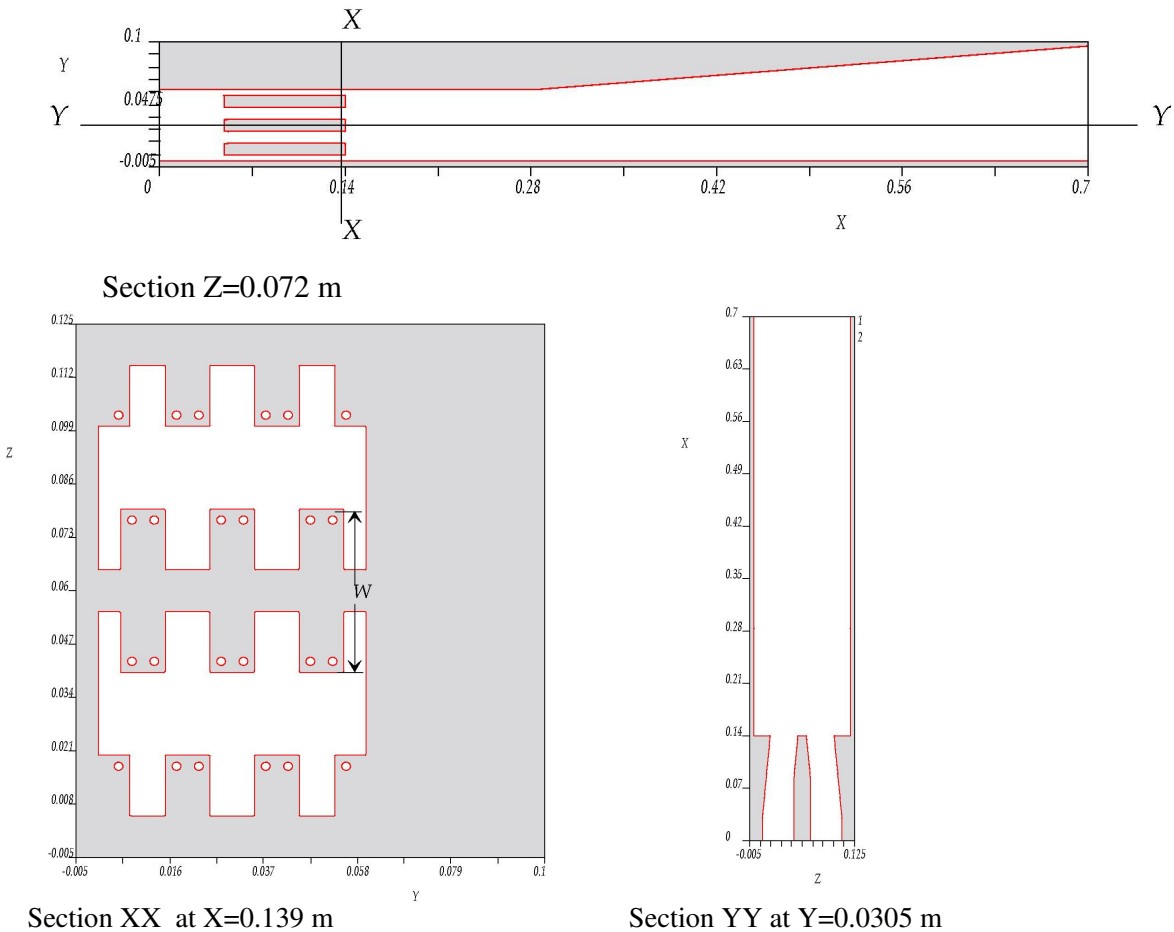


Figure 4.43 Geometry of the Scramjet combustor

Figure 4.42 shows the isometric view of the Scramjet combustor with the struts and the coordinate system shown. The origin of the coordinate system is at the inlet of the combustor although for clarity sake in figure it is shown at the exit. Figure 4.43 shows the geometrical details of the tested Scramjet combustor with strut based injection. Gaseous hydrogen is injected through 24 holes of 2.5 mm diameter located in the strut base at angle of 16 degrees from the horizontal. The struts provide streamwise vortices which will aid in the mixing of gaseous hydrogen and incoming vitiated air. Figure 4.44 shows the schematic of the test set up of Scramjet connected pipe mode test. The high temperature high pressure vitiated air from the heater of circular cross section is taken through an interface adaptor to Scramjet combustor having rectangular cross section. The flow after the adaptor enters the nozzle

(100mm in length) and which provides Mach number 2 flow to the strut based combustor. It is to be noted that the leading edge of the strut is not simulated in this test. The combustor consists of one full strut and two half struts without leading edge, a constant area duct portion followed by a five degree divergent. The combustor end is open to atmosphere and has ambient conditions. The stagnation pressure, stagnation temperature, and mass fractions are measured in the heater exit. Static pressure at the end of the nozzle is also measured and is used as the entry condition to the combustor for the CFD computations. The facility nozzle is of about 100 mm in length and is not simulated in the present simulations since the length of the facility nozzle is small and the influence of its boundary layer on the flow is not expected to be significant.

Numerical computations are performed from the exit of the nozzle to the combustor exit. Supersonic inflow conditions are imposed corresponding to the nozzle exit conditions at the start of the combustor and since no flow separation is seen at the exit of the combustor from the tests for the present experimental conditions, supersonic outflow conditions are imposed at the combustor exit. Only one half of the geometry is considered for computations with symmetry conditions imposed in the symmetry plane. All other boundaries are wall boundaries.

The computation was done for the following test conditions

Flow rate of vitiated air = 2.5 kg/s

Total flow rate of gaseous Hydrogen from 24 holes = 56.92 gm/s (Equivalence ratio ϕ=0.778)

Mass fraction of Oxygen = 0.2327, Mass fraction of Nitrogen = 0.5691

Mass fraction of Carbon-dioxide = 0.1215, Mass fraction of water-vapour =0.0767

Incoming vitiated air pressure = 1.267 bar

Incoming vitiated air density = 0.42 kg/m^3

Temperature of incoming vitiated air = 1048 K

Stagnation pressure = 9.66 bar

Hydrogen injection pressure =6.5 bar

Mach number of incoming vitiated air =2.0

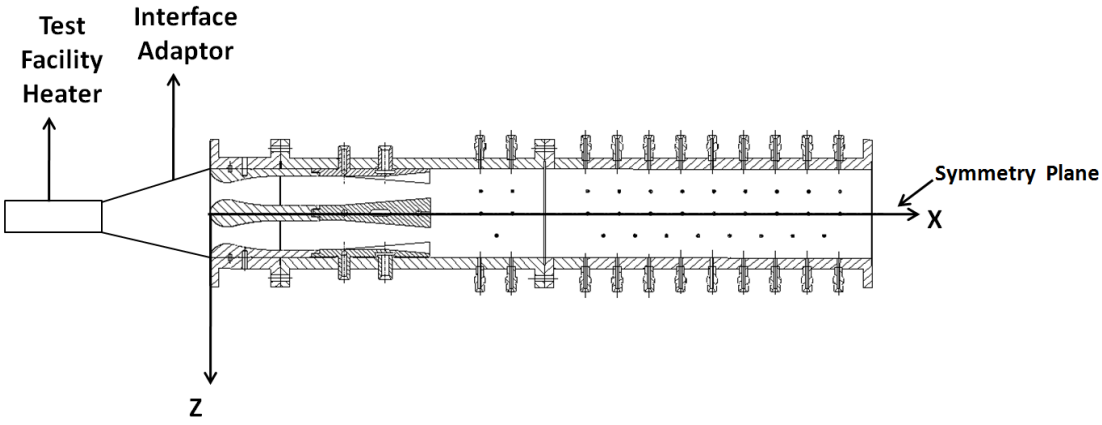


Figure 4.44 Schematic of the Scramjet test combustor

The air intake brings down the velocity of the free stream flow from hypersonic Mach numbers to supersonic Mach numbers through oblique shock compressions with loss in total pressure. The real challenge in the intake design is to achieve this reduction in velocity with maximum pressure recovery and at the same time with minimum total pressure loss. The static temperature at the inlet of combustor due to intake compression for a hypersonic air intake would be normally more than the ignition temperature of Hydrogen-Air mixture and hence the auto ignition would take place. For the above conditions, the numerical simulation for Hydrogen-vitiated air combustion with turbulent flow without turbulence-chemistry interaction was carried out. Boundary conditions for the geometry shown in fig.4.44 is as follows

Xmin – Supersonic inflow

Xmax – Supersonic outflow

Zmin – Symmetry

Zmax – Wall

Ymin – Wall

Ymax –Wall

The wall is considered adiabatic. 8-species ($H_2, O_2, H_2O, OH, H, O, N_2, CO_2$) and 7- reactions ONERA chemical kinetics model is used as given in Table 2.7 of section 2.6. The chemical species N_2 and CO_2 are considered inert to reactions

which is a good approximation for temperatures less than 2500 K as in the present case. Standard κ-ε turbulence model is used with a turbulent Prandtl number of 0.92 and inlet turbulence intensity of 1% and inlet turbulent viscosity taken same as laminar viscosity which is widely used inflow conditions for turbulent quantities. The computations were performed for only one half of the combustor since there is a geometrical symmetry existing in Z direction as shown in Figure 4.44.

Point implicit scheme is used for computing the species production terms. The computations were carried out with a CFL number of 0.1. Initial grid used was 70 X 50 X 22 with three levels of oct-tree division near the body to capture the geometry properly and this resulted in a total number of cells of 122600. The solution was refined after every 15000 iterations and the refinement criterion was based on differences in flow parameter between adjacent cells. To perform this refinement, a non-dimensional flow gradient parameter ψ_{cell} is defined as follows

$$\psi_{cell} = \sum_{i=1}^{Neighbours} \left(\frac{\Delta P_i}{P_m} + \frac{\Delta \rho_i}{\rho_m} + \frac{\Delta V_i}{V_m} \right) \qquad (4.8)$$

Where

ΔP_i is $\left| P_{cell} - P_i \right|$ is the absolute value of difference in pressure between the cell and its i^{th} neighbour.

$\Delta \rho_i$ is $\left| \rho_{cell} - \rho_i \right|$ is the absolute value of difference in density between the cell and its i^{th} neighbour.

ΔV_i is $\left| V_{cell} - V_i \right|$ is the absolute value of difference in resultant velocity between the cell and its i^{th} neighbour.

$P_m = \max(P_{cell}, P_i)$ is the maximum value of pressure between the cell and its i^{th} neighbour and similar expression is used to get ρ_m and V_m.

If the value of ψ_{cell} exceeds a user-defined value, which is called as the flow refinement criteria then the particular cell would undergo oct-tree division. The value

of flow refinement criteria used is 0.5 for the present problem which was good enough to refine the flow gradients. The solutions underwent three levels of flow gradient based solution adaptation. Figure 4.45 shows the plot of Mach number at a section Y=47 mm. Although the simulations are carried out only for half the geometry, the full section is shown by reflecting the solution in Z direction.

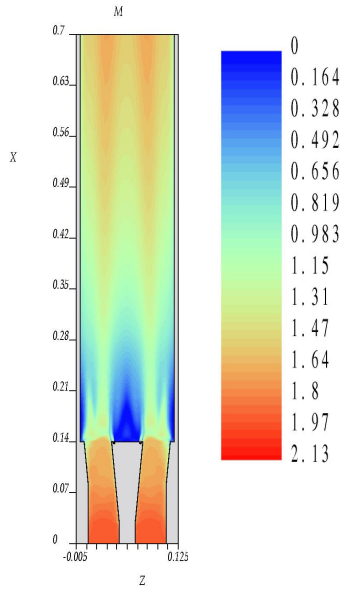
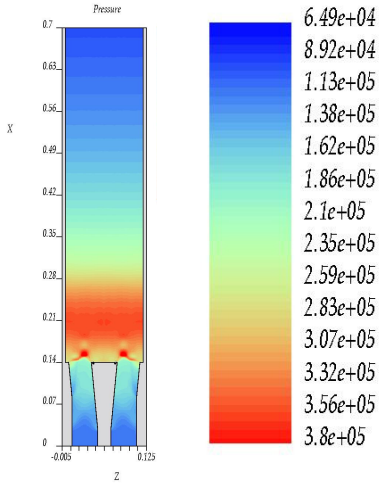


Figure 4.45 Mach number field at section Y= 0.047 m for equivalence ratio 0.778



4.46 Pressure distribution at section Y=0.047 m for equivalence ratio 0.778

Figure 4.45 shows the inlet Mach number of 2.0 in the connected pipe mode condition getting reduced as the combustion occurs. Figure 4.46 shows the pressure distribution at a section 47 mm above the bottom wall and the pressure rise due to combustion can be very clearly seen behind the strut base. The hydrogen that is injected through the holes of the strut mixes with the air in the recirculation zone behind the strut. Thus the strut base is the one that holds the flame which has relatively lower velocities as compared to the other regions. It is to be noted that in the above mentioned computations the interaction of turbulence with the chemical reactions are not considered. In other words, the turbulent flow models will give mean temperature and mean density based on RANS computations which is used to calculate the species production rates. In reality there would be fluctuating temperature and fluctuating species concentrations which would give rise to fluctuating species production rates. However the above turbulence-chemistry interaction effects would be dominant only if the reaction time scales and turbulent mixing time scales are of the same order. If the turbulent mixing time scales are much smaller than the reaction time scales then the flow is mixing dominated and the turbulent chemistry interactions would not play a dominant role. It would be shown, later in this section that even with this approximation, the match of pressure distribution with that of the experimental results are quite good indicating that the flow is mixing dominated in the present case. Figure 4.47 shows the initial grid of 122000 cells used for the solution. As the solution progresses, grid adaptation is done after every 15000 iterations based on flow gradients. Figure 4.48 shows the final grid after adaptation which is about 4.4 million cells and the zoomed portion of the combustion zone showing finer mesh. Figure 4.49 shows the pressure distribution along the combustor top wall for different grids. Plot shows that the results are grid independent by the second level of grid refinement when the cells are about 1.83 million. Figure 4.50 shows the water vapour mass fraction at section 47 mm from the bottom wall.

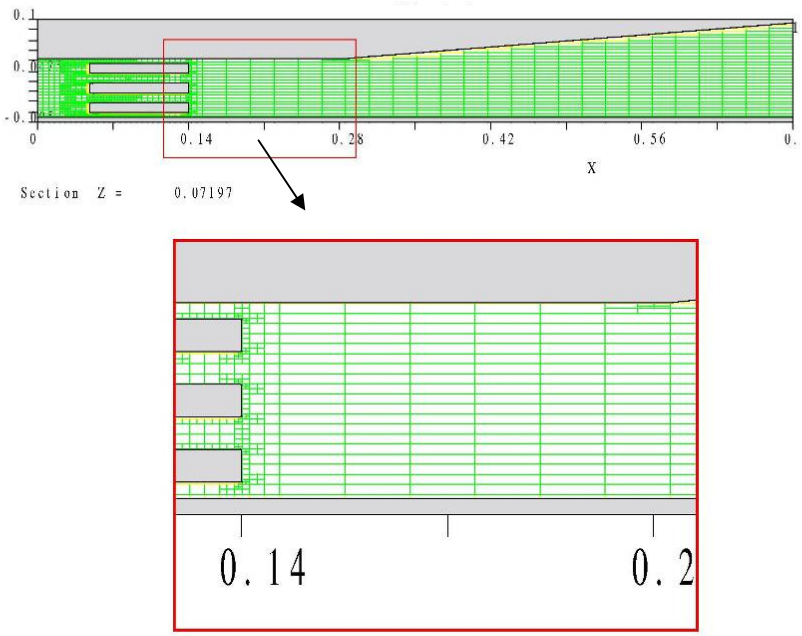


Figure 4.47 Initial grid with 122000 cells with zoomed portion near strut

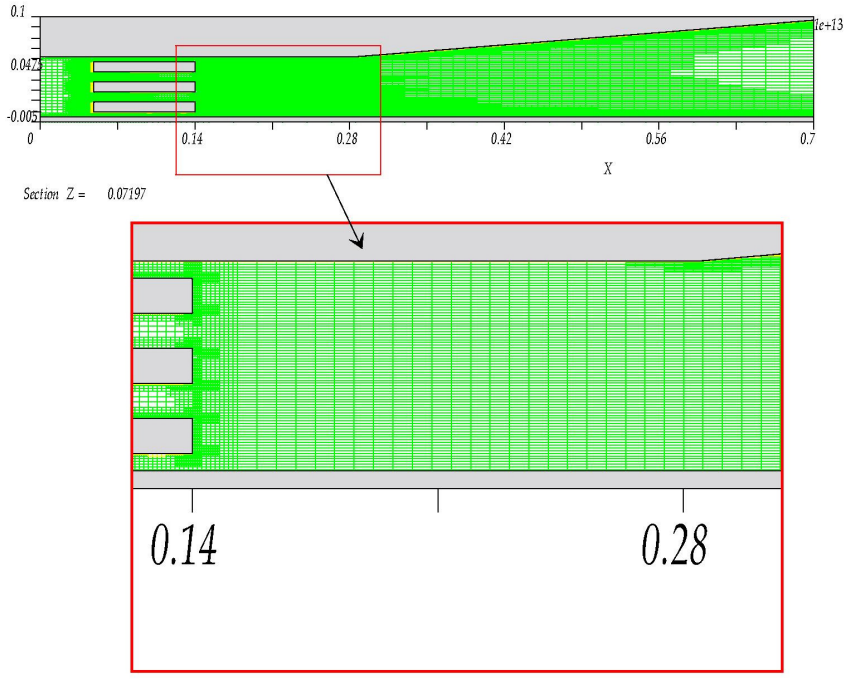


Figure 4.48 Final grid with 4.4 million cells after 3 levels of flow adaptation with zoomed portion near strut shown

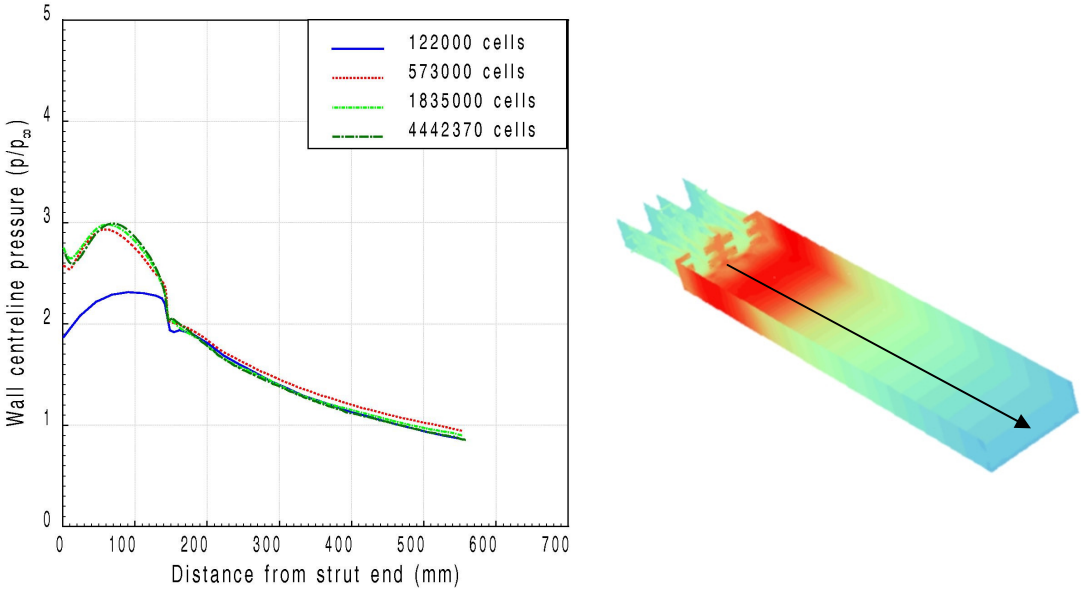


Figure 4.49 Grid independence plot for centerline pressure

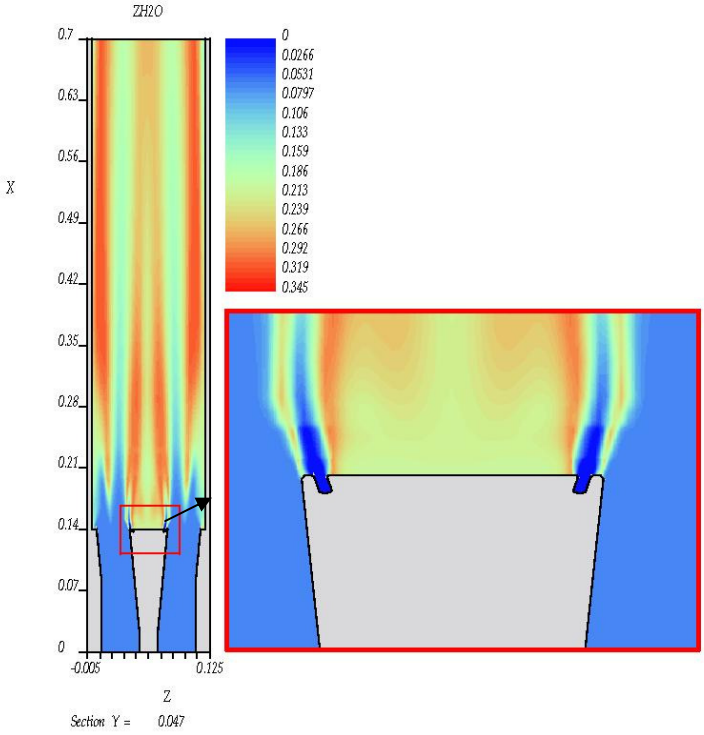


Figure 4.50 Mass fraction of water vapour at a section 47 mm from bottom wall

It can be seen that the gaseous hydrogen has to travel certain distance before the multiple step reactions are completed because of the non-equilibrium chemical effects and hence the water vapour mass fraction is very small in this region.

Figure 4.51 shows the Hydrogen mass fraction at section 47 mm from the bottom wall. It can be seen that most of the hydrogen has undergone mixing and combustion in the portion behind the strut within the constant area region of the combustor. Figure 4.52 shows the combustion efficiency along the combustor length. Cumulative combustion efficiency is defined as the water vapour mass up to a at a particular section to the ideal water vapour mass that would exist at that section if total combustion would have taken place. Ideally the water vapour mass formed for complete combustion should be 9 times the Hydrogen mass which in the present case turns out to be 512.82 gm. Most of the combustion takes place at the base of the strut in the constant area section. Although the combustion process in Scramjet is considered as supersonic, most of the combustion actually takes place at low speed regions behind the struts. However, the mass averaged Mach number at any section which is defined as the ratio of product of mass flow rate and the Mach number at each cell of the section summed over all cells of the section to the total mass flow rate in the section, was seen to be more than one.

Figure 4.53 shows the pressure along the top wall compared with that of the experimental results from the connected pipe mode experimental results. The computation is able to capture all the trends of the experiments. The position of peak pressure is well captured and only very small difference in the magnitude is noticed. Although the turbulence-chemistry interactions are not modeled, the match is quite good. This could be because in the present case, the mixing time and the chemical reaction time scales are not of the same order and hence the turbulence-chemistry interactions are not significant. An estimate of the mixing time of the large scale structures which is ratio of characteristic mixing length to characteristic flow velocity show that the value is about 15 μsec (0.02/1400) for this problem and the reaction

time of the Hydrogen air combustion is of the order of 50-70 μ sec from Chakraborty et al. (2000). The Damkohler number based on the ratio of mixing time to reaction time is then of the range 0.2 to 0.3.

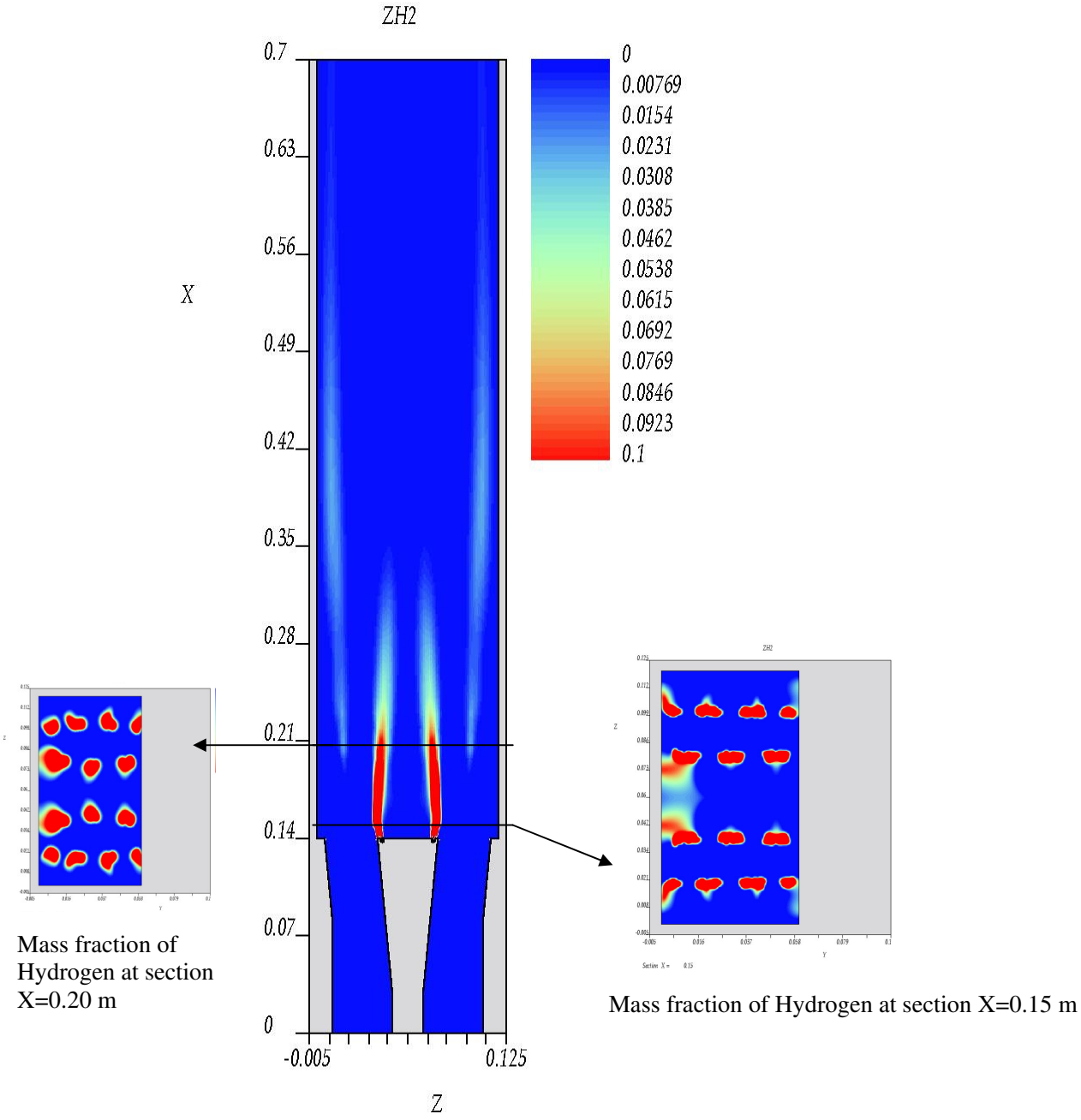


Figure 4.51 Hydrogen mass fraction at a section Y=47 mm from bottom wall

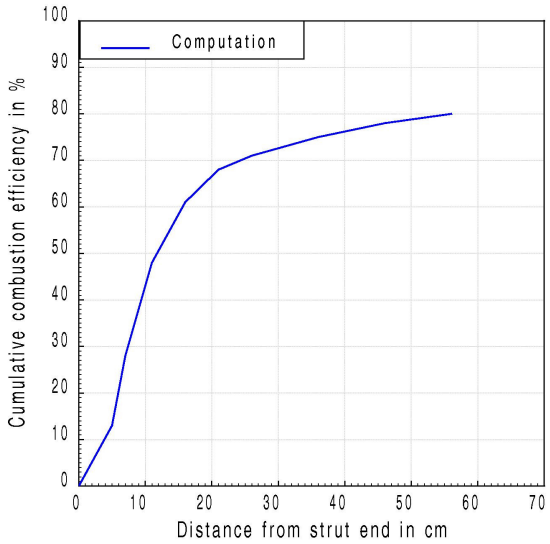


Figure 4.52 Cumulative combustion efficiency plot along the combustor

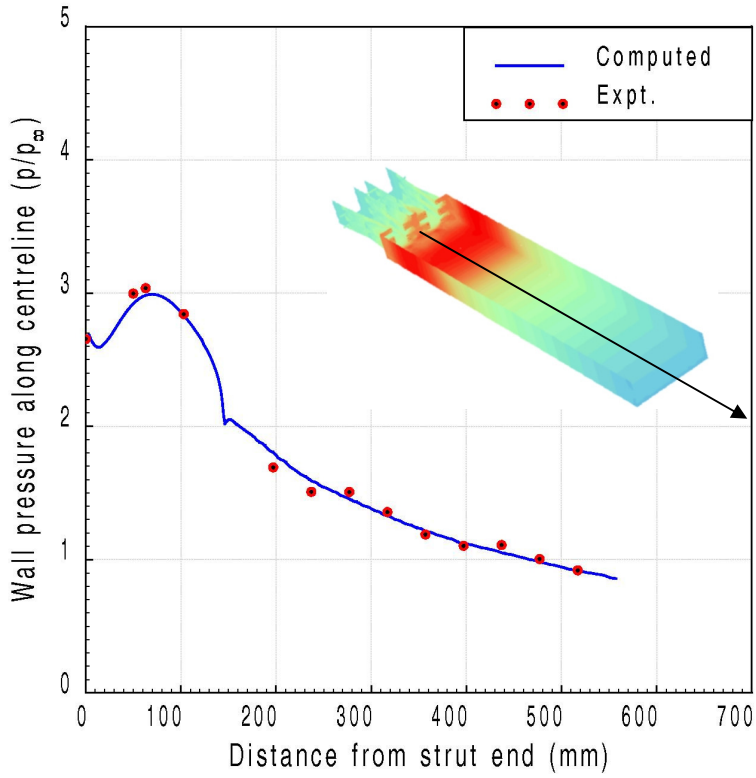


Figure 4.53 Computed non-dimensional pressure along the center line of bottom wall compared with experimental results

In the turbulence-chemistry interaction modeling, the probability of fuel and oxidizer coming together by suitable probability distribution function is taken into account and this consideration would be to reduce the pressure rise as compared to the computations without considering this effect. The experimental measurements also had the stagnation temperature measurements at the exit. Figure 4.54 shows the plot of the stagnation temperature compared with that of the experiments at the measurement location. The measurements were made at center, Y=0.048 m along the Z direction at points 0.105 m,0.09 m and 0.06m (symmetry plane) as shown in figure 4.54. The predicted total temperature matched reasonably well with the experiments considering the fact that the experimental accuracy in measurement of total temperature was of the order of 100 K. The computed total temperature is obtained from the total enthalpy and the species concentrations at the exit. The total temperature is more at the exit which is in line with the strut base because of more mixing and hence good combustion. Whereas for the region between the struts, the mixing and combustion is less and the water vapour mass fraction is less thereby giving lesser total temperature between the struts. Figure 4.55 shows the static temperature at a section Y=47 mm from the bottom of the strut. The plot clearly shows that the combustion is dominated at the strut base with increased temperature of the fluid and gets convected downstream. Figure.4.56 shows the total pressure plot at the exit of the combustor. The total pressure plot shows reduction in total pressure in regions of combustion as expected. Figure 4.57 shows the Mach number plot at the exit section of the combustor. The regions which are in the same line as the strut base have lesser Mach number due to more combustion as expected. It is to be noted that the experiments conducted are in connected pipe mode and do not simulate all the flight conditions that the flight combustor would encounter.

In order to understand the effect of performance on the Scramjet combustor due to differences in the flow conditions encountered by the actual flight combustor as compared to the connected pipe mode conditions, numerical experiments are performed to study this effect and are described in the next section.

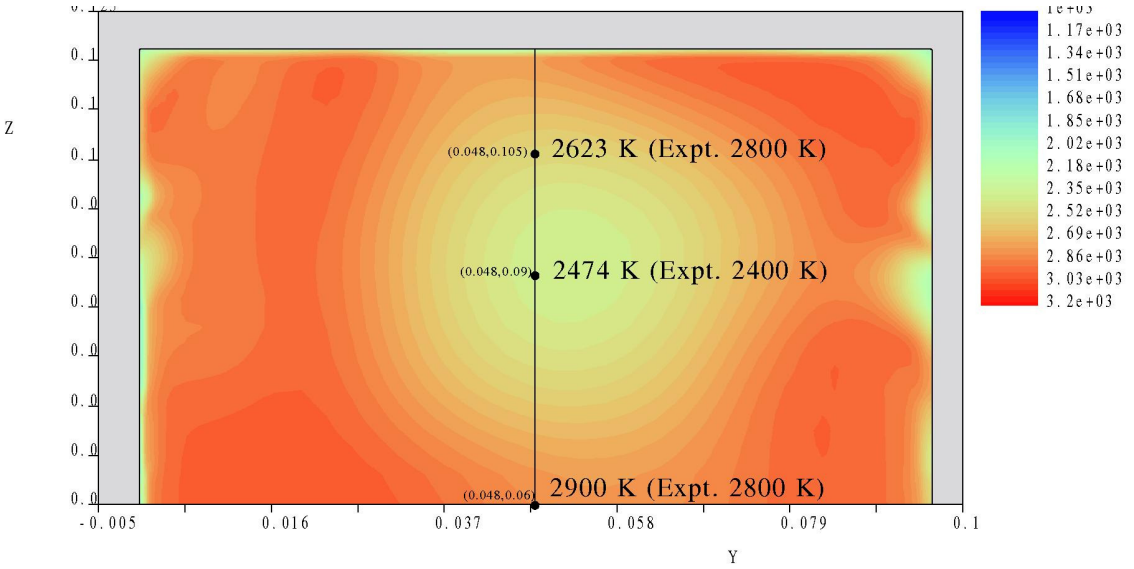


Figure 4.54 Stagnation temperature plot at the exit section of the combustor with available experimental points at three positions

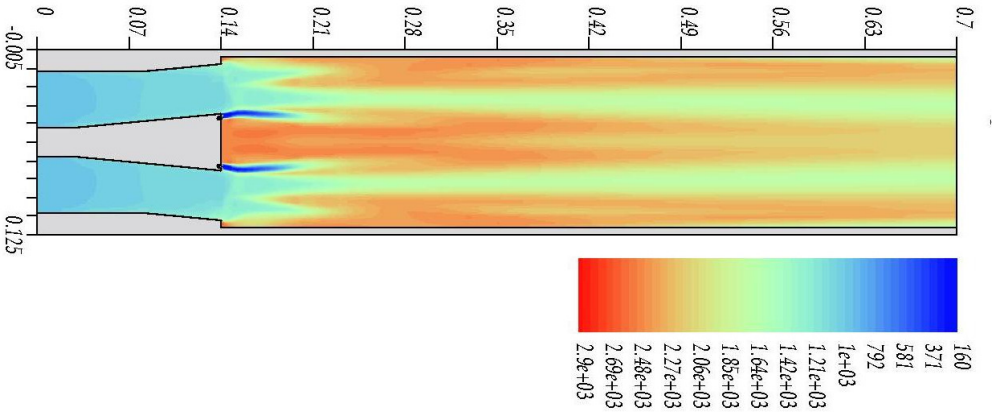


Figure 4.55 Static temperature plot at a section Y=47 mm

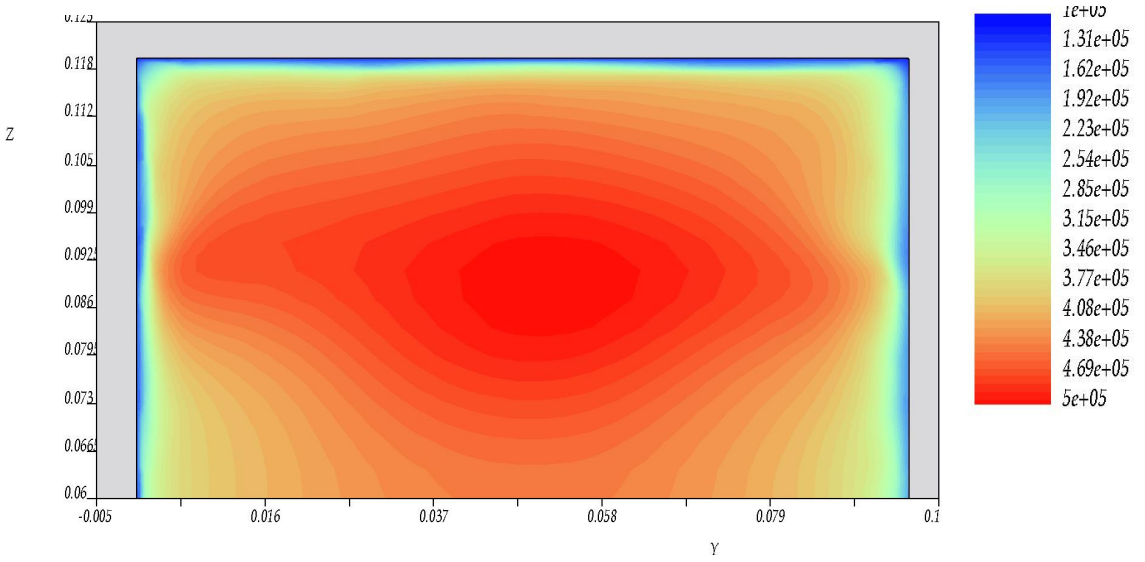


Figure 4.56 Total pressure plot at the exit of the combustor

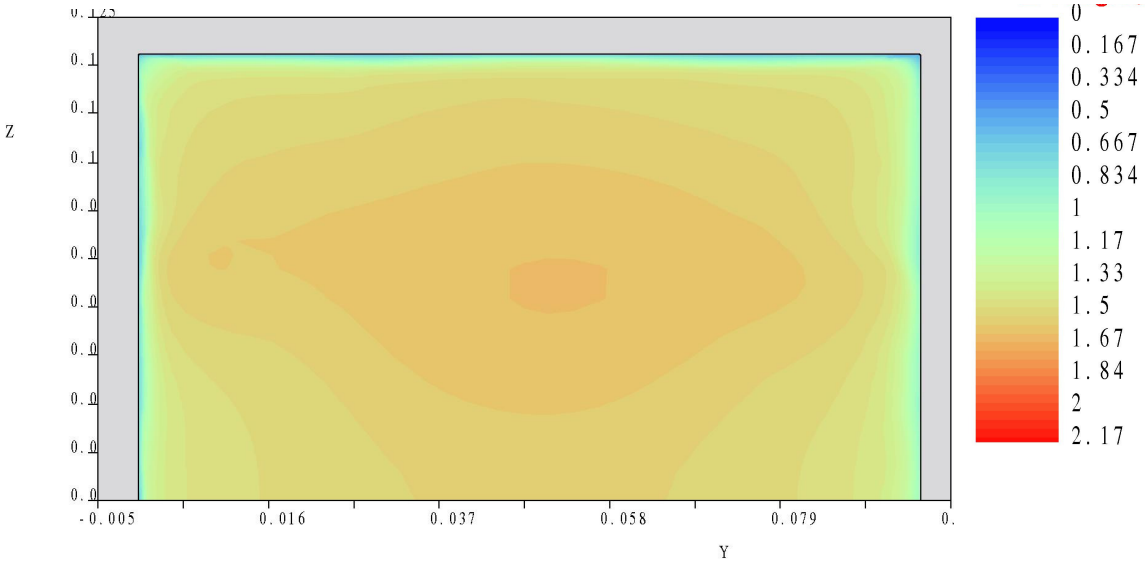


Figure 4.57 Mach number plot at the exit section of the combustor

4.5 Effect of Connected Pipe Mode Test Conditions on the Performance of Scramjet Combustor

Scramjet combustor tested in ground conditions in connected pipe mode has limitations in simulating all the parameters related to flight conditions. To simulate the hypersonic flight conditions in ground tests, stored high pressure air is heated before it is expanded through the nozzle. Total pressure and stagnation temperature would correspond to that of the flight. Generation of high enthalpy flows can be done through the methods of shock tube heating, storage heating, arc heating, electric heating and combustion heating. There are advantages and disadvantages of each type of heating. While shock tubes produce highest enthalpy, the run times are of the order of milliseconds only. Storage pebble bed heaters would contaminate the gas with particulates and arc heaters contaminate with oxides of Nitrogen. Although electric heating produces clean test gas, the power requirements would be prohibitive. Combustion heating through burning hydrogen or hydrocarbons offers low cost method of generating high enthalpy although in this process also the air gets vitiated. Thus in this method of heating the inlet high enthalpy air will have combustion products like Carbon-dioxide and water vapour. Owing to this, the effects of vitiation in the ground tests have to be understood properly to extrapolate the connected pipe mode test results to flight conditions. Effect of vitiation has been reported by Pellet et al. (2002), Goyne et al. (2007) and very recently by Luo Feiteng et al. (2012). To understand the effect of the connected pipe mode test conditions on the combustor performance a numerical experiment was conducted by Gnanasekar, Ashok et al. (2009) to study the effect of inlet static pressure and vitiation and is described below.

Combustor geometry and flow details

Figure 4.58 shows the combustor geometry considered for conducting the study. It has a rectangular cross section with constant area region followed by divergent area region. Length of the combustor is 14.3H and width is 4H where H is the height of the combustor at the entry. Fuel is injected through three equi-spaced strut configuration similar to the one used and reported by Scherrer et al. (1995) . Struts

have leading edge in the front and ramps in the rear from the base through which the fuel is injected through discrete circular holes in the axial direction.

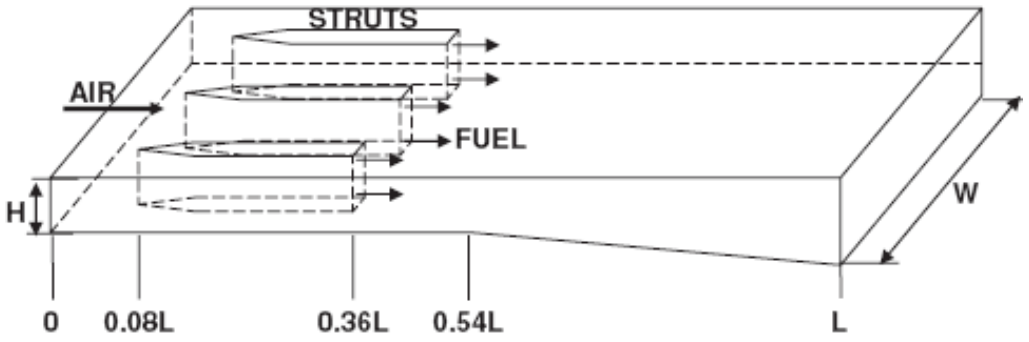


Figure 4.58 Combustor geometry to study the effects of inlet pressure and vitiation

Hydrogen gas is injected at sonic speed and at stagnation temperature of 300 K. Pressure of injection is varied to suit the fuel equivalence ratio. Air enters the combustor at Mach number 2.7 and at a stagnation temperature of 1920 K. This corresponds to the flight Mach number of 6.5. The composition of incoming air considered for the study is Nitrogen with mass fraction of 0.78 and Oxygen with mass fraction of 0.22. However for vitiation studies the vitiated air with water vapour and Carbon-dioxide is considered.

Computational details

Computation domain and the initial grid is shown in figure 4.59. An initial grid 100X100X50 is used for the study. Considering the symmetry of the combustor configuration, only half the geometry with symmetry boundary conditions is considered. Supersonic inflow conditions are imposed at Xmin boundary and supersonic outflow at outflow boundary. At Zmax, the symmetry boundary is imposed and Wall boundary conditions with modified wall function of Hagemann et al. (1996) are imposed on all other boundaries.

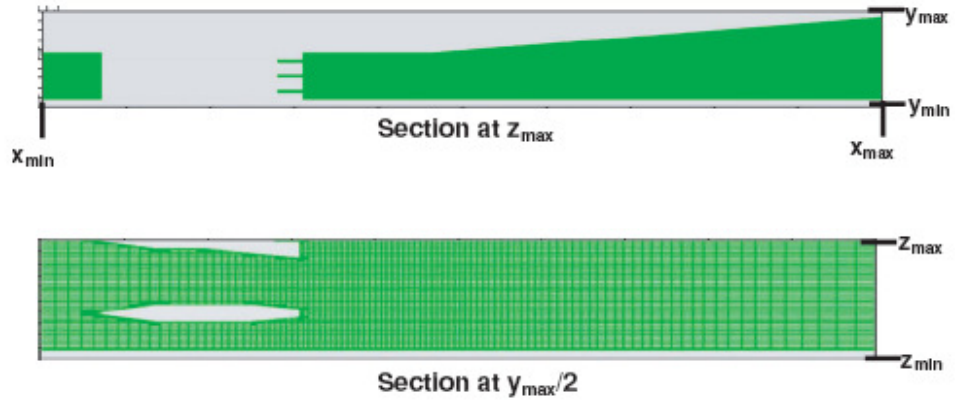


Figure 4.59 Computational domain and initial grid

The computations were carried out till a converged grid independent solution was obtained. The final mesh after refinement is 4.4 million and it was found that by 45000 iterations the solutions had converged.

4.5.1 Effect of inlet pressure on combustor performance

Computations were carried out for a nominal combustor entry pressure of 0.35 bar which corresponds to a typical flight Mach number of 6.5 with 60 kPa free stream dynamic pressure and pressure recovery of air intake of about 15%. The nominal static pressure was varied by 15% to study its effect on performance. Thus the simulations were carried out for static pressure of 0.35 bar, 0.52 bar and 0.23 bar and for two fuel equivalence ratios (ER) 0.42 and 0.65. Figure 4.60 shows the plot of Hydrogen consumption along the combustor length. It can be seen that more than 90% of Hydrogen is consumed in all cases over the entire length of the combustor with most of the consumption taking place immediately downstream of the strut. However for the case of inlet static pressure of 0.23 bar with fuel equivalence ratio of 0.42, Hydrogen consumption increase is delayed although it picks up later. This is because at lower static pressures, the reactions rates are slower due to lower levels of concentrations and thereby having less number of collisions in the molecular level to cause reactions. This causes the slower water vapour formation for lower pressure and equivalence ratio.

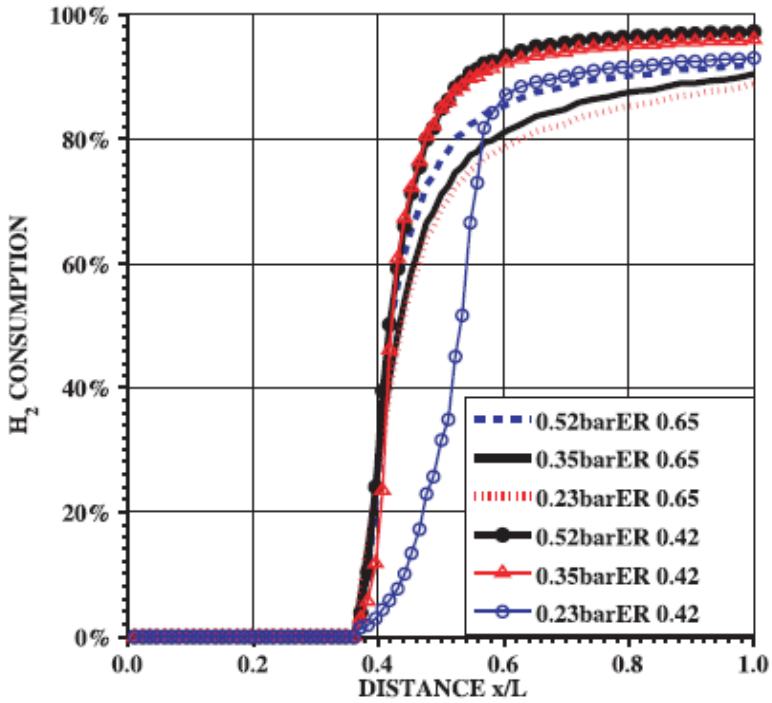


Figure 4.60 Hydrogen consumption along the combustor for various inlet static pressures and fuel equivalence ratios

. Since the inlet pressure at 0.23 bar for equivalence ratio 0.42 showed this behavior, the computations were performed at still lower pressures for higher equivalence ratio of 0.65 to see whether the same phenomenon is noticed there also. Figure 4.61 shows that at higher equivalence ratio of 0.65 also, the same observation of slower Hydrogen consumption is seen as in the case of ER 0.42 although for pressures 0.17 bar and below. This is because for higher fuel equivalence ratio, to have the same level of concentrations as lower equivalence ratio, the static pressures have to be lower. Figure 4.62 shows the Hydrogen converted to water vapour expressed as percentage which is the ratio of actual H_2O formation from the H_2 injected to the H_2O formed if entire H_2 is converted to H_2O (i.e total H_2 flow rate X 18/2). The reduced pressure gives rise to a slower combustion and

reduced formation of water vapour although Hydrogen consumption was found to be almost the same as that for higher pressure.

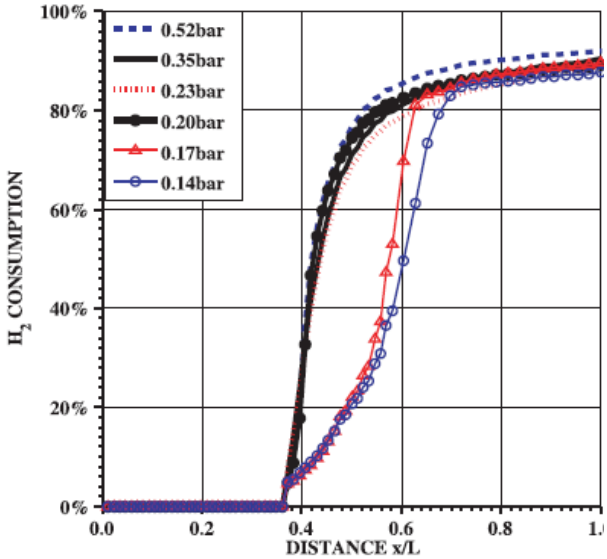


Figure 4.61 Hydrogen consumption along the combustor for various inlet static pressures for equivalence ratio 0.65

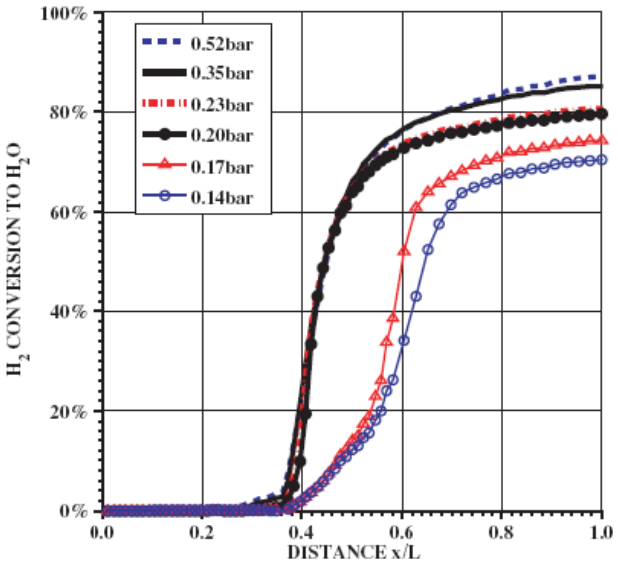


Figure 4.62 Hydrogen conversion to water vapour for various inlet pressures for an equivalence ratio of 0.65

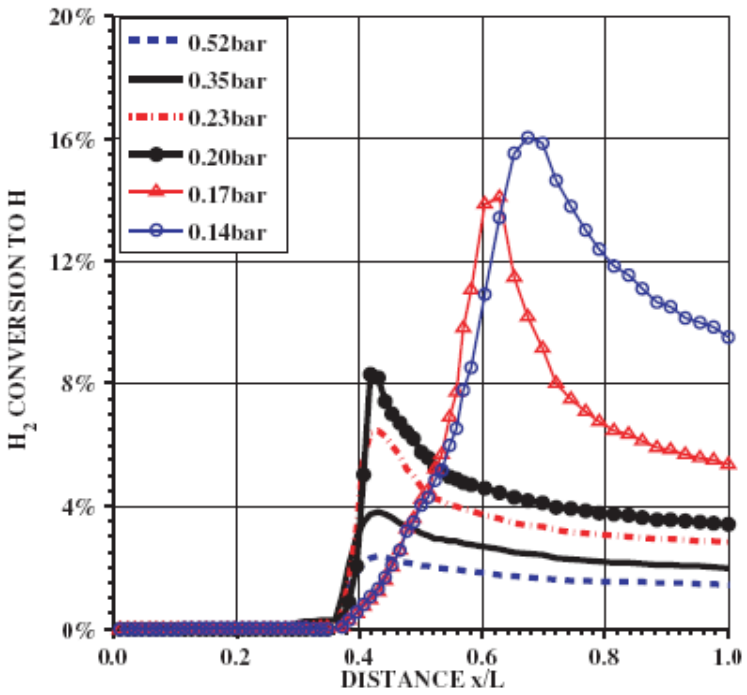


Figure 4.63 Percentage of Hydrogen converted to H for various inlet pressures for equivalence ratio 0.65

Interestingly in the case of lower static pressures, the reaction paths are such that more atomic Hydrogen is formed instead of getting converted to H_2O, as shown in figure 4.63.

4.5.2 Effect of vitiation on combustor performance

In order study the effect of vitiation, inlet air of 0.52 bar pressure at Mach number 2.7 and total temperature 1920 K and with Nitrogen mass fraction of 0.59, Oxygen mass fraction of 0.22, water vapour mass fraction of 0.075 and Carbon-dioxide mass fraction of 0.115 was considered for the simulations. As it is not possible to simulate all flow parameters as that of the clean air, the parameters that are kept same as that of clean air are Oxygen mass fraction, pressure, Mach number and total temperature. The mass flow rate of vitiated air is about 5% less due to slight variation in specific heat and molecular weight. Figure 4.64 shows the rise in mass

averaged total temperature for clean and vitiated air. In the case of vitiated air, the rise in total temperature is about 8% less than that of the clean air, for fuel equivalence ratio 0.42. Whereas for 0.65 the reduction in total temperature for vitiated air is 15%. The presence of Carbon-dioxide and excess water vapour tend to absorb more heat thus reducing the total temperature and thereby the pressure rise.

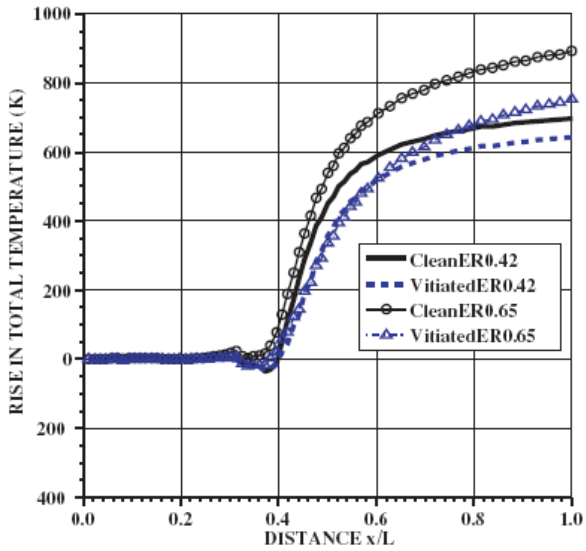


Figure 4.64 Total temperature rise with vitiated and clean air

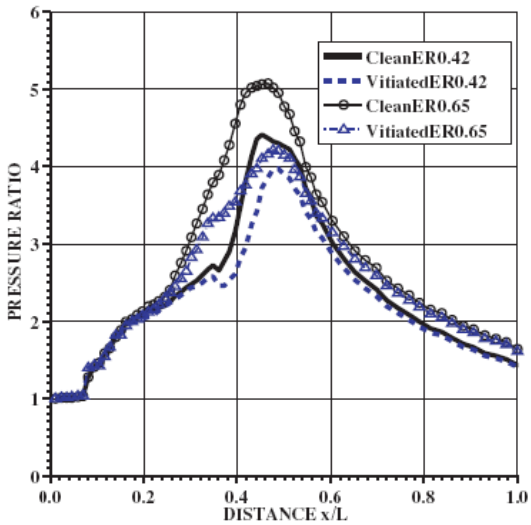


Figure 4.65 Effect of vitiation on pressure rise

Figure 4.65 shows the pressure ratio which is the ratio of area averaged pressure to the inlet static pressure of 0.52 bar. This is evaluated at each section along the combustor length. The pressure rise is more for the clean air as compared to vitiated air as the temperature rise is more for clean air due to lesser coefficient of specific heat.

The validated solution obtained to this typical Scramjet combustor problem demonstrates the capability of the present code to predict the Scramjet combustor performance with non-equilibrium chemically reaction of Hydrogen-Air combustion with turbulence on a Cartesian mesh. Experimentally, one of the ways of assessing the combustor performance is through the connected pipe mode tests which do not simulate all the requisite flight conditions. Hence some of the factors affecting the combustor performance in connected pipe mode test condition are brought out through numerical experiments. Now the next logical step is to perform the end-to-end simulation of the Scramjet engine with intake, combustor and nozzle to get thrust delivered by the engine. Since the engines are normally mounted on some rocket body, it is desirable to perform the computations of Scramjet engine mounted on a rocket body. Such types of simulations demand huge computational power as the mesh sizes are very large. High performance computing plays a key role for such computations. The Graphic Processing Unit (GPU) computations are new generation computing paradigm that offers tremendous advantage for such problems, if the numerical codes have parallel computing algorithms adapted to such type of hardware. The topic of the next chapter is the development and application of such parallel computing algorithms suitable for computation in GPU platforms to harness the power of such new architectures in order to perform the complex non-equilibrium chemically reacting turbulent flows with combustion, typical of Scramjet vehicles.

CHAPTER-5
PARALLEL COMPUTING WITH GPU ACCELERATORS

The advent of parallel computing has brought about tremendous advantages in terms of reduced turnaround time for solution to large scale complex problems like tip-to-tail simulations with combustion of Scramjet engine. Parallel computing works on the philosophy of "divide and conquer" and the computing speed is achieved by means of three approaches. Modern parallel computing clusters usually employ all the three methods in combination as given below to maximize the parallel computing performance.

1) Use of multi-core CPU (Central Processing Unit) processors in a single machine which is like an SMP (Symmetric Multi Processing) system.
2) Using an accelerator like GPU (Graphic Processing Unit) having large number of GPU cores in each unit which can perform SIMD (Single Instruction Multiple Data) computations very efficiently.
3) Using a cluster of multi-core CPU machines with GPU accelerators with a fast interconnect like Infiniband and enable distributed computing with MPI (Message Passing Interface).

In the case of multi-core CPU, a task can be divided into multiple parallel tasks with each core performing the assigned task and all of them using the common memory and hence called as shared memory system. To achieve this, the program has to be multi-threaded using Pthread (Posix thread) library and each task assigned to a thread. From the programming perspective, since all the parallel threads use the common memory, the multi-threaded program has to be thread-safe which means that a variable that gets updated and used by all the threads should essentially be passed through functions and not put in the common memory.

Although it is very much possible to implement an MPI (Message Passing Interface) paradigm in an SMP system, the preferred way is a shared memory approach using multiple parallel threads in a single process. This is because there is an additional overhead of communication among the cores of the machine if MPI is used. However the SMP systems fail to scale up on a large number of processors since they use a common memory. While in a message passing approach, one needs to partition the job into a number of smaller jobs with proper load balance, in a multi-threading method, parallel threads in a program with proper load balance has to be found out. In the case of MPI approach, a proper balance of communication and computation is needed for the efficient use of the system. To get the best performance from parallel computing, a combination of SMP systems and MPI approach is employed in the modern high performance computing approach. Therefore the approach is multi-thread method within a node and MPI approach across the nodes.

In addition to the above two approaches of SMP system and MPI approach in a cluster, the use of accelerators for higher compute performance is a recent approach. In this method, a portion of the computation very much suitable to be done in an accelerator is offloaded to the accelerator. In the case of GPU accelerators, the computations which are of the SIMD (Single Instruction Multiple Data) type are identified and submitted to the GPU. This means that to get good performance from the GPU accelerators, the computational algorithm should be highly data parallel. It is to be noted that GPU is only an accelerator and cannot function without the CPU processor and hence is considered as a co-processor to the CPU. CFD solution methodology with GPU accelerators has been reported by Julien C Thibault and Inanc Senocak (2009), Everett Philips et al. (2010), Dana Jacobsen and Inanc Senocak (2011), and Hai P. Le and Jean-Luc Cambier (2012) for solving various problems. However, GPU implementation on adaptive Cartesian mesh with hanging nodes which is very challenging due to lack of inherent data parallelism is not noticed to be reported in the literature to the best of author's knowledge. The aim of this chapter is to bring out new data parallel algorithm with the important feature of

grouping the Cartesian cells with hanging node into eight different categories for effective GPU implementation which is not noticed to be reported in the literature. Also the factors affecting the performance of the parallel computing with GPU accelerators for CFD solutions using adaptive Cartesian mesh is brought out. The computing algorithms need to be in tune with the architecture of the GPU and CPU hardware. The important features of the MIMD (Multiple Instruction Multiple Data) architecture of CPU and SIMD (Single Instruction Multiple Data) architecture of the GPU are explained below.

5.1 SIMD and MIMD Architecture

In order to get the best performance from the CFD code it is essential to program according to the architecture of the processors. The GPU architecture as given in NVIDIA (2011) is designed such that more transistors are devoted to data processing rather than data caching and flow control unlike the CPU processor.

Figure 5.1 Schematic of GPU architecture (adapted from NVIDIA (2011))

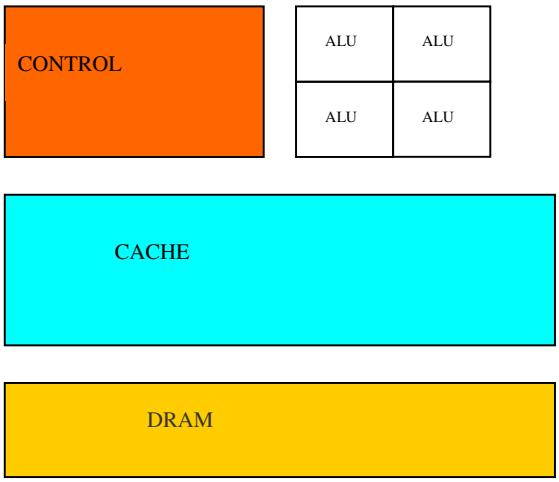


Figure 5.2 Schematic of CPU architecture (adapted from NVIDIA (2011))

The schematic of the GPU architecture is shown in figure 5.1 and that of CPU in figure 5.2. The CPU core is more sophisticated as compared to the GPU core as it has better data caching, flow control and arithmetic logic unit (ALU). While the CPU can handle multiple instruction multiple data type of algorithms in the program, the GPU is most suited to address problems that can be expressed as data-parallel computations. This means that the same instruction is executed on many data elements in parallel with high arithmetic intensity which is the ratio of arithmetic operations to memory operations. Thus the CPU is more suited to task parallel type of jobs whereas GPU architecture is tuned to data parallel type of instructions both of which are explained below.

5.1.1 Task Parallelism

Task Parallelism is execution of threads (tasks) on different or same code with different or same data across different parallel computing cores.

For example if we consider a Quadcore processor, each of the core is capable of performing independent tasks of the same code as given below.

if(processor 1) do c=a+b

if(processor 2) do res=sqrt(a-b)

if(processor 3) do t=(l+e)(l-e)

if(processor 4) do m=a*b

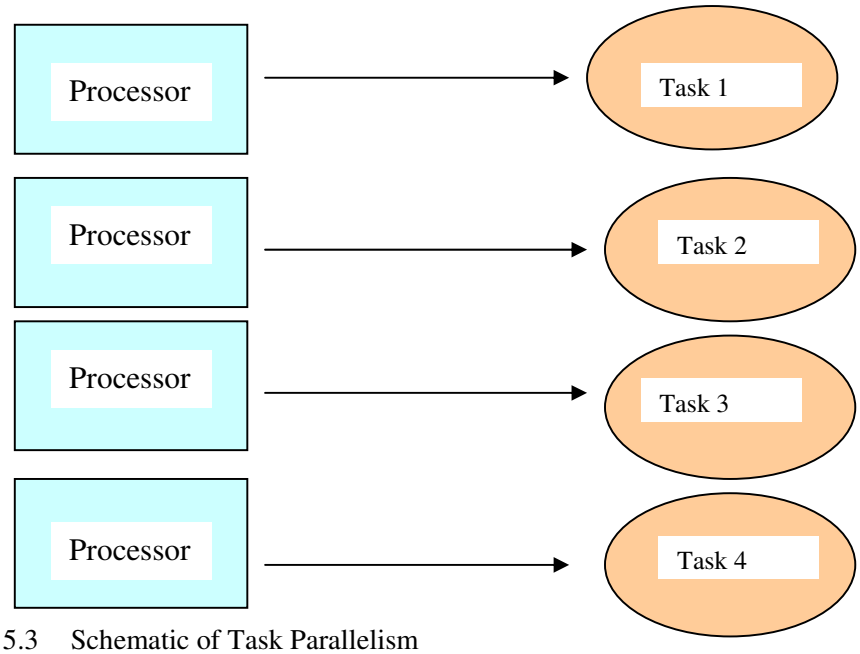


Figure 5.3 Schematic of Task Parallelism

If the processor 1 of the Quadcore performs grid generation task and processor 2 performs flow solution while processor 3 and 4 is engaged in performing post processing tasks then it is multi-tasking with different codes. Figure 5.3 shows the schematic of Task Parallelism. Task parallelism is able to handle Multiple Data Multiple Instructions (MIMD) by the cores which are typical of the CPU cores.

5.1.2 Data Parallelism

In Data Parallel computations, each processor executes same set of instructions on different pieces of data. This aspect is illustrated through an example given below.

```
int i;
double c[1000],a[1000],b[1000];d[1000];
for(i=0;i<1000;i++){
```

```
    c[i]=a[i]+b[i];
    d[i]=a[i]*b[i];
}
```

The above do loop can be executed say in 10 cores, simultaneously by assigning data of a[i] & b[i] corresponding to i=0 to 99 in core 1 and from 100-199 in core 2 and so on up to 10^{th} core in the form of 10 parallel threads. The operation c[i]=a[i]+b[i] is executed in each thread of all the 10 cores but with a different data corresponding to the a[i] and b[i]. Thus it is a single instruction, but multiple data corresponding to the respective arrays. Once this operation is completed, the next operation d[i]=a[i]+b[i] is taken up for execution by each thread. The schematic of the Data Parallel computation is shown in figure 5.4.

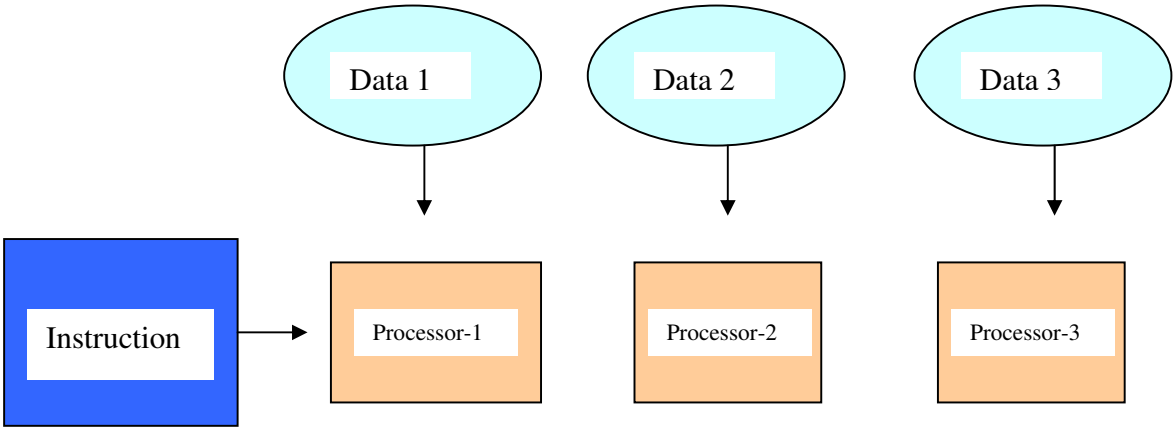


Figure 5.4 Schematic of Data Parallel computation

It is to be noted that in the above example, identical operation is conducted for each core through a single operation which does not have any branching statement. If there are branching statements, whose outcome is not always the same, then the performance of the Data Parallel computations would get affected as shown by the example given below.

```
    int i;
    double c[1000],a[1000],b[1000];
    for(i=0;i<1000;i++){
      if(a[i]>10) c[i]=a[i]+b[i];
    else c[i]=a[i]*b[i];
}
```

As in the previous example, the data is distributed among 10 cores and execution is carried out in 10 parallel threads. Considering, the first operation if(a[i]>10) c[i]=a[i]+b[i], only those threads having the data a[i]>10 can execute the statement. A GPU thread can move on to the next operation only after all the other GPU threads have finished executing the first operation. Thus those GPU threads which have finished executing the first operation have to wait, till all the other GPU threads have completed the first operation. However this is not the case with CPU threads which are more powerful to undertake flow control. Hence there is a considerable waiting or idling for GPU thread if the computations are not data parallel. This aspect has to be taken into account while designing the CFD algorithm to run on GPU threads.

Since the same instruction is executed for each data element, there is a lower requirement of sophisticated flow control. It is to be noted that GPU architecture possesses large memory bandwidth as compared to CPU but with lower memory access as the memory has to be shared with large number of GPU cores. Hence with operations involving high arithmetic intensity in a GPU, the penalty of memory access latency is not very much visible because of more number of computing load from large calculations which can give good performance without big data caches.

Applications which involve large data sets and that can use a data programming model would get good speed up from GPU computations. One of the examples is 3D rendering, wherein large sets of pixels and vertices are mapped to parallel threads. Similarly, image and media processing applications such as post processing of

rendered images, video encoding and decoding and pattern recognition that can map image blocks and pixels to parallel processing threads are some of the other examples suitable for GPU computations.

While the graphical applications mentioned above have a natural data parallelism, the use of GPU in other fields like CFD would pose a real challenge in making the program highly data parallel. Thus new data parallel algorithms have to be evolved for harnessing the real power of GPU for CFD applications.

5.2 GPU implementation using CUDA

The GPU processes data in Single-Instruction-Multiple-Thread (SIMT) fashion. The Compute Unified Device Architecture (CUDA) programming model of NVIDIA (2011) is used for GPU implementation. GPU accelerator used for the computations in the present work was TESLA M2070 which had 448 cores with 14 Streaming Multi-processing (SM) capability with each SM having 32 cores. The instruction that gets executed in a GPU is called the Kernel and is invoked from the host CPU. The CUDA programming model consists of grid and thread block. A grid consists of a number of thread blocks and each thread block contains a number of threads. The maximum number of threads that can be allotted to a block is 1024. The thread blocks can be one, two or three dimensional. Figure 5.5 shows the grid of thread blocks which are two dimensional with each block consisting of 12 threads. The number of blocks in the grid would depend on the number of threads allotted in a block. For example if there are 2000 cell computations of CFD allotted to be done in a GPU and if the number of threads allotted to a block is 100, then the grid would have 20 blocks with each block having 100 cells and each cell computed by a thread. Thread block computations are allotted to SM for computations and at each block, threads are organized into group of 32 threads called as Warp and computations performed in an SIMT (Single Instruction Multiple Thread) fashion for this Warp. Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory

access. For efficient cooperation, the shared memory is expected to be a low-latency memory near each processor core, just like an L1 cache.

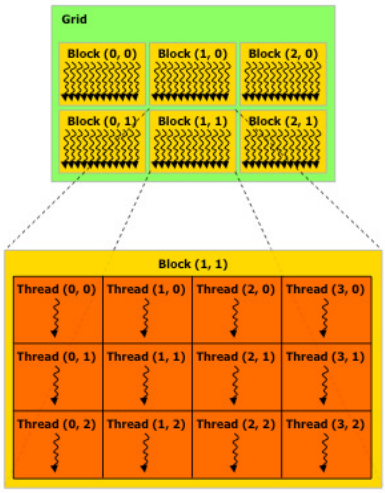


Figure 5.5 Grid of Thread Blocks (adapted from NVIDIA (2011))

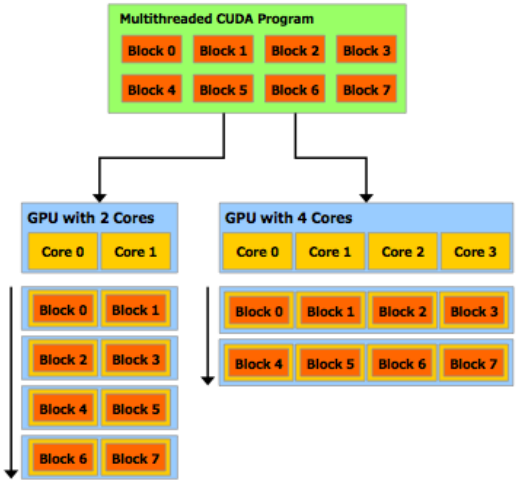


Figure 5.6 Data Parallelism in GPU (adapted from NVIDIA (2011))

Since all the threads of a block are expected to reside in the same GPU processor core, the number of threads per block is restricted by the limited memory resources of the GPU processor core. When a Kernel is called, the scheduler unit on the device will automatically assign a group of thread blocks to the number of

available GPU cores on the device. Once the GPU cores have completed the calculation, it will be assigned another block. Since, communication between thread blocks is not there, the GPU with more cores will perform calculations faster. Figure 5.6 represents the data parallelism in GPU.

An instruction given to a Thread Block is handled by the GPU which contains a number of GPU cores. All the threads within each block will be organized into groups of 32 threads called Warps which are executed in a SIMT manner as mentioned before. The main difference in data parallelism between Grid and Thread Block is that while there is a synchronization mechanism for all threads in a same block, it is not there for all the blocks in the grid. Hence it is important to ensure that there is no data dependency between Thread Blocks. The GPU implementation for a Cartesian mesh based CFD code was carried out and the various algorithmic steps to obtain good performance are given in the next section.

5.3 Parallel Computation of Cartesian Mesh solver

The parallel computation of the Cartesian mesh solver is done following the steps given below

Step -1	Domain decomposition
Step-2	Setting up communication links
Step-3	Grouping of cells for data parallelism
Step-4	Identifying computations to be done in CPU and GPU
Step-5	Performing computations in CPU and GPU
Step-6	Communicate the values of boundary cells after each iteration.
Step-7	Repeat 5 & 6 till convergence is attained.

5.3.1 Domain Decomposition

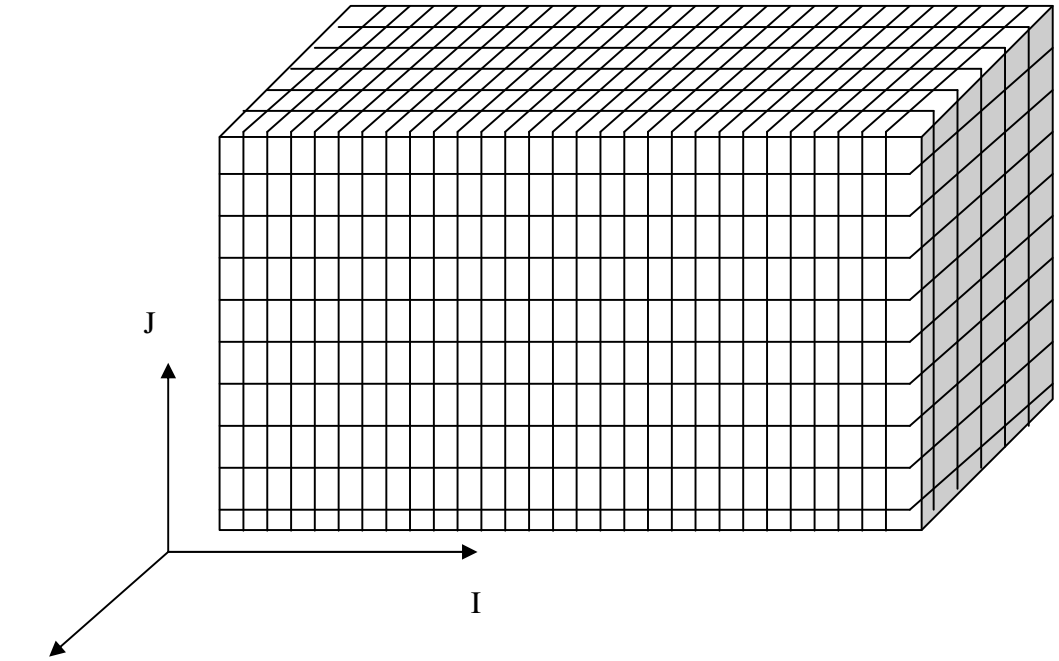


Figure 5.7 Computational domain of a typical Cartesian mesh

The computational domain for flow over a body with Cartesian mesh will be a rectangular Parallelepiped with cells in I, J and K directions as shown in figure 5.7. In order to solve for flow for this particular domain using cluster of "N" multi-core CPU machines with GPU accelerators, the first step is to divide this domain into nearly equal "N" sub-domains. The various algorithmic steps to achieve this for a cluster of "N" machines, say 21 for the purpose of illustration is given below.

Step-1

Compute the total computational load in the domain as per the following expression. Total load=1.0*number of gas cells+ 1.4*number of partial cells. The weight given for a partial cell is 1.4 times the gas cell as it has more computations involved by way

of implementation of body boundary condition. This additional time taken for partial cell was found out by checking the CPU time for partial cell computation. If a neighbour of the cell is split then unit load is added and if neighbour's neighbour is split then additional 10 units are added. These values have been arrived at based on time taken for actual computations with such type of cells. The body cells have zero weight as no computations are performed for these cells.

Step-2

Divide the total load distributed to 21 machines into two halves as given below

Load on first half= Total load*11/21

Load on the second half=Total load*10/21

March along the I direction and compute the total load for cells up to a particular I section. The parent cell can be identified by a three dimensional index (I,J,K) and for computing the total load up to I section say I=5 , one has to calculate the total load for all cells from cell (0,0,0) to (5,Nj,Nk) where Nj and Nk is total number of cells in J and K directions. For each parent cell, there could be many levels of split cells as shown in figure 5.8 which are recursively found out and the total cell load takes into account the computational load of all the split cells.

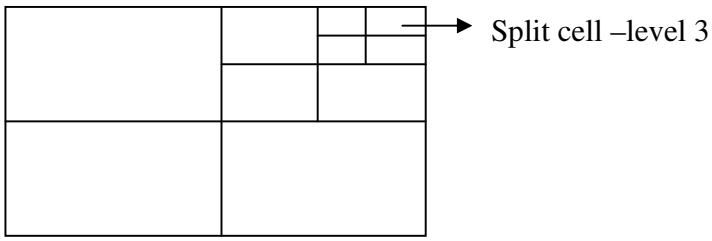


Figure 5.8 A Cartesian cell with three levels of division

If the load computed up to a particular I section is within 0.9 to 1.1 times the load on the first half, then the two halves are obtained by splitting the domain into two by cutting at that particular I section. Subsequently the same procedure is carried out along J and K directions and checked whether a better load balance is obtained as

compared to I section splitting. Out of the three ways of splitting into two halves, the split that gives the best load balance is taken for the splitting. Figure 5.9 shows the first level domain decomposition in which first half is allotted to 11 machines and the second half to 10 machines. Subsequently, the first half is further split into two halves in the same manner mentioned so as to obtain a splitting section that gives the best load balance giving rise to two sub-domains. The load on the first sub-domain of Domain (0) named as Domain (0, 0) will be distributed among 5 machines and the load of second sub-domain of Domain (0) denoted as Domain (0, 1) will be distributed among 6 Machines. Similarly Domain (0,0) is further split into two sub-domains in the same manner as shown in figure 5.7 as Domain (0,0,0) and Domain (0,0,1) and distributed among 3 and 2 machines respectively. This way the domain decomposition is carried out by consecutively splitting along the I, J or K section which gives the best load balance. It is to be noted that domain decomposition is done on a single machine and the output of domain decomposition is the starting and ending parent cell I, J, K values, the total number of cells and the machine to which they are allotted.

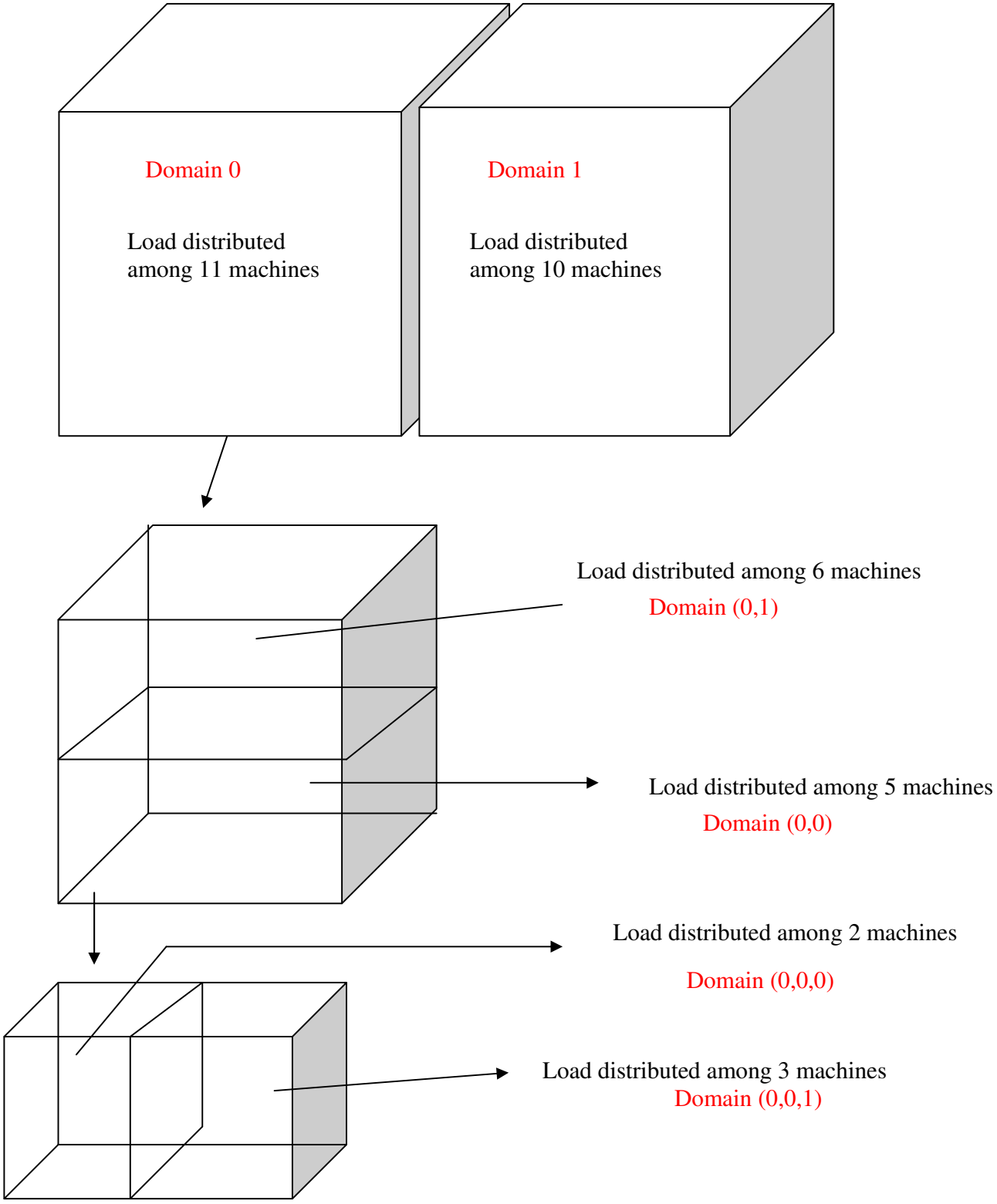


Figure 5.9 Domain decomposition by consecutive splitting

5.3.2 Setting up Communication Links

In this step, the cells which have to communicate to other machines are tagged and also provided with information as to the respective machine they need to communicate with. In order for the communication process to be done in minimum time, the communication information of cells from one machine to another particular machine is sent together as one packet and not individually. Also the packet of information that is sent from one machine to another machine falls in the proper location of the respective machine.

Figure 5.10 shows the schematic of the Cartesian cells without split cells that participate in communication for a domain which is split into two halves. Except for one face in each of the half, all the other faces are boundary faces for which boundary condition need to be applied. Machine 0 has Nx0 column of cells where Nx0 is the number of cells in I direction for Machine 0. Similarly, Machine 1 has Nx1 column of cells where Nx1 is the total number of cells in I direction for Machine 1. Even though the number of columns of cells is Nx0 for Machine 0, the computations are performed for cells of column 1 to column Nx0-2 only and these cells are called as active cells. The last two columns of cells namely $(Nx0- 1)^{th}$ column and $Nx0^{th}$ column of cells for which computations are not performed in Machine 0 are called dummy cells. These two dummy columns of cells are used to reconstruct the cell values at the last active column of cells. In the case of Machine 1 the computations are performed from the third column of cells to the last column. Thus the first two column of cells are the dummy columns of cells for Machine 1. The last four column of cells of Machine 0 and first four column of cells of Machine 1 are kept identical. The last two columns of dummy cells in the Machine 0 are kept same as third and fourth column of active cells of Machine 1 and the first two column of dummy cells of Machine 1 are kept same as the last two active column of cells of Machine 0 at Nx0-2 and Nx0-3.

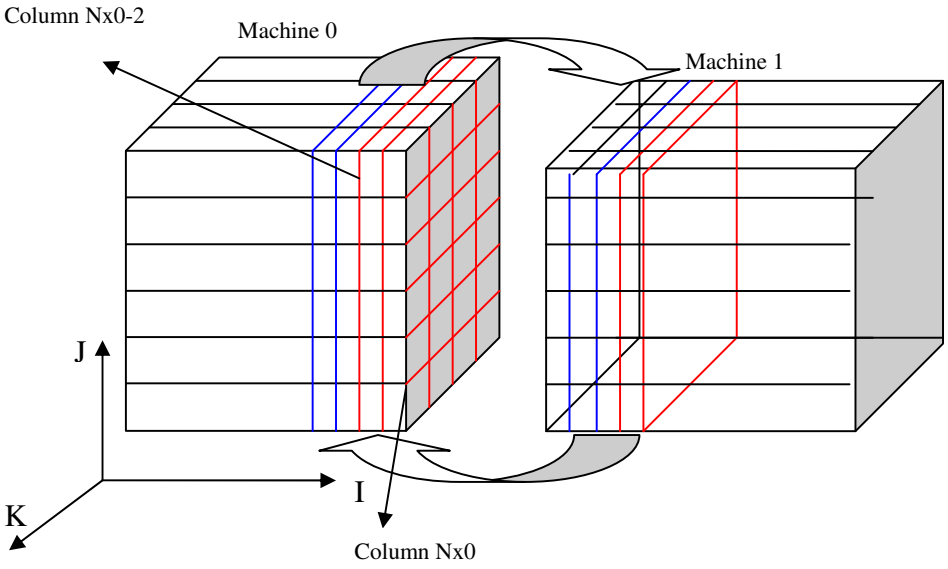


Figure 5.10 Schematic of communication

The computations in different machines are initiated through MPI. While the calculation takes place from column 1 to column Nx0-2 in Machine 0, the computations at Machine 1 takes place from column 2 to Column Nx1. If the computational load in the two machines is same, the computations get completed in the respective machines at the same time. Then after the first iteration, Machine 0 communicates the updated value of conserved variable vectors of last two active column of cells (Nx0-2,Nx0-3) to Machine 1 which will get allocated to the first two dummy column of cells (columns 0 and 1) of Machine 1. Similarly after the first iteration, Machine 1 sends the updated value of conserved variable vectors of third and fourth column of cells (first two active column of cells) to Machine 0 which will get allocated to the last two column of dummy cells (Nx0-1,Nx0) of Machine 0. By this process of two way communication, the values of the two columns of dummy cells get updated. Then the next iteration is again started simultaneously in the two machines by the MPI. Since the dummy column values are updated in both the machines, a proper estimation of fluxes in the last two active columns of cells in Machine 0 and first two columns of active cells in Machine 1 is possible in the next

iteration. This parallel process of computation followed by communication to update the dummy cell values are continued until the converged results are obtained. While communication is carried out, the complete information that is needed to be sent from one machine to another machine is sent as one packet containing certain number of bytes

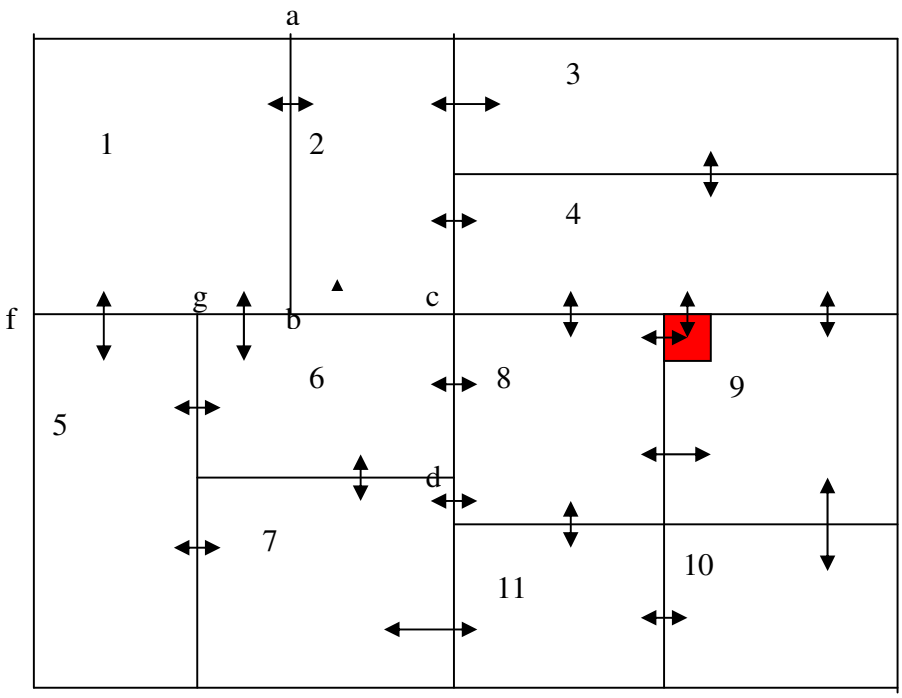


Figure 5.11 Communication links in the sub-domains

In the receiving machine the values get allocated to the appropriate cells by proper ordering of memory allocation of the dummy cells in the receiving machine. Once the domain is decomposed into as many numbers of sub-domains as the number of machines used for parallel computation, the conserved variable vector information of group of boundary cells that need to be send to another machine is identified. Figure 5.11 shows a domain split into 11 sub-domains with proper load balance and the two way communication links. It should be noted that a face sometimes needs to communicate with more than one machine. For example in Figure 5.11 it can be seen that for the bottom face, fgb of Machine 1, the cells along face fg need to

communicate with Machine 5 and cells along face gb with Machine 6. Also the cells at the corner as shown for Machine 9 in red colour have to communicate with more than one machine, like Machine 4 and Machine 8. All these aspects are taken into account while setting up the communication link.

5.3.3 Cell Grouping for Data Parallelism

Once the domain decomposition and the communication links have been set up, the next step is to group the cells having similar computations and allocate them to be computed to CPU and GPU. This grouping of cells is necessary, since the GPU has a SIMD architecture which demands same logic flow in the statements as in Data Parallel tasks. Figure 5.12 shows the Mach number flow field for a typical launch vehicle at supersonic Mach number and figure 5.13 shows the corresponding flow adapted Cartesian mesh. From the figure it can be clearly seen that at regions of large gradients the cells are divided. Also, it can be noticed from the figure that for some cells neighbour is split where as for some others the neighbour's neighbour is split. The different type of cells in a rectangular adaptive Cartesian mesh can be broadly classified to 8 types of groups as described below and shown in Figure 5.14

Group-1 Boundary partial cells
Group-2 Boundary air cells
Group-3 Partial cell whose neighbour's neighbour is split
Group-4 Gas cells whose neighbour's neighbour is split
Group-5 Partial cell whose neighbour is split
Group-6 Gas cell whose neighbour is split
Group-7 Partial cell whose neighbour and neighbour's neighbour is not split
Group-8 Gas cell whose neighbour and neighbour's neighbour is not split

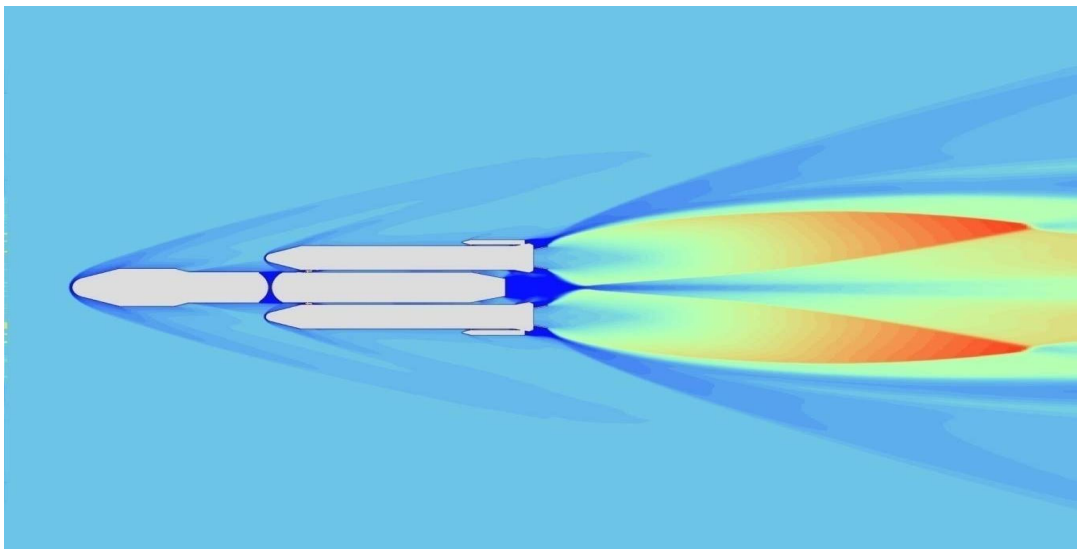


Figure 5.12 Mach number field in supersonic flow for a typical launch vehicle with jet-on condition.

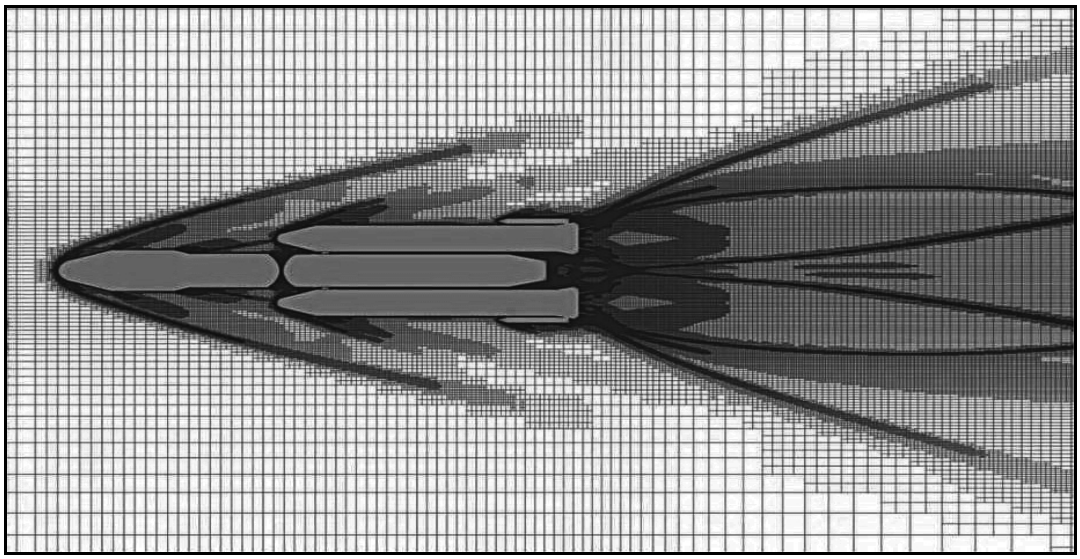


Figure 5.13 Flow-adapted Cartesian mesh for the flow field shown in Figure 5.12

.

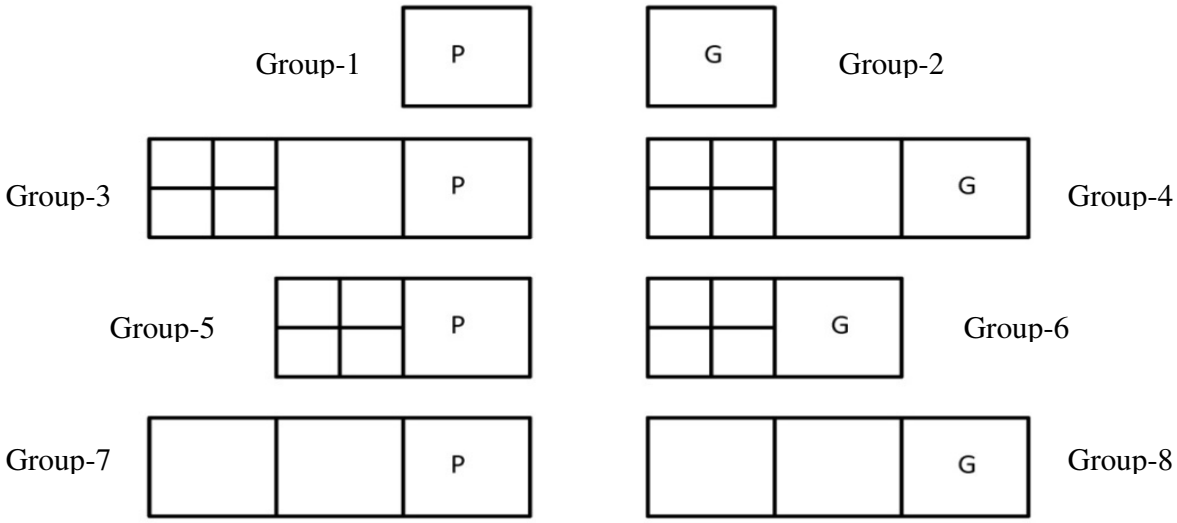


Figure 5.14 Schematic of 8 different cell groups

The data dependency of each cell could vary from 12 to 120. Figure 5.15 shows the schematic of typical cell marked as red with data dependency of 12 i.e. it needs 12 neighbouring cells information with two cells information adjacent to each cell face to compute the fluxes. Two neighbours adjacent to a face are needed for linear reconstruction of primitive variables at a face to obtain second order accuracy in the solution. If the neighbour to face is split and neighbour's neighbour is further divided as shown in figure for the cell denoted by blue colour, then 20 neighbours adjacent to each face of the cell is needed for linear reconstruction. This is the maximum data dependency needed for a Cartesian mesh with one hanging node for second order accurate solution.

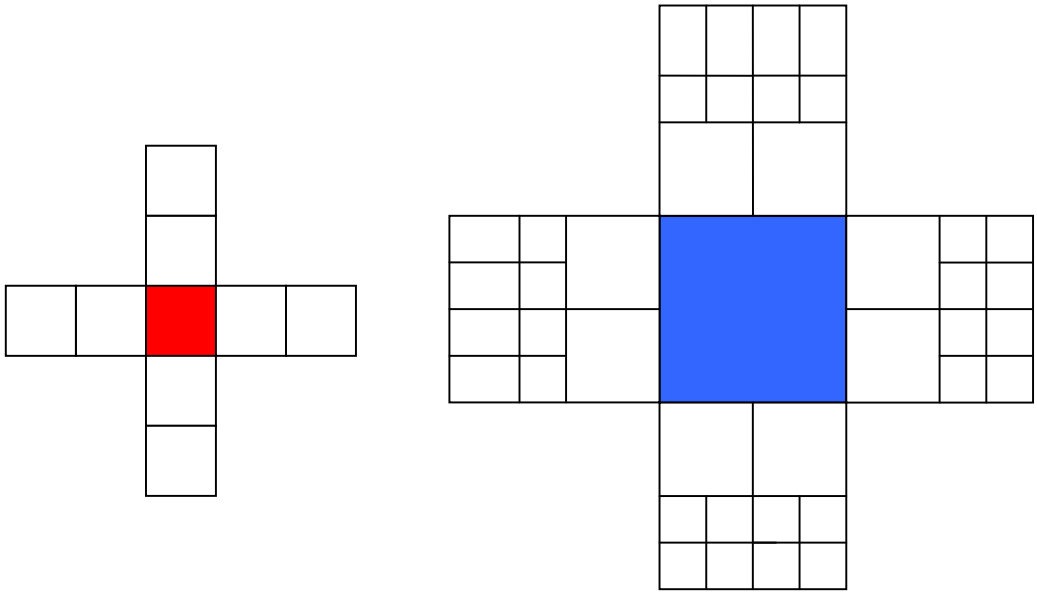


Figure 5.15 Schematic of cell with 12 and 120 cells data dependency

The eight cell group categories are arrived to have good efficiency in GPU computations. Each group of cells will be able to perform the computations in the GPU almost in identical fashion. Performing computations with a mix of different cell groups results in large performance loss in GPU accelerators. Analysis of the Cartesian grid for most of the flow problems like the one given in Figure 5.11 would show that most of the cells in the grid fall in the last group (cells which are not at the boundary and whose neighbour and neighbour's neighbour is not split), which are the most data parallel cells. Partial cells occur only near the body. The cell groups 4 and 6 where the gas cells neighbour or neighbour's neighbour is split occur only in the region of the grid where the level of the oct-tree changes. The number of cells in group 4 and 6 could increase as a result of the flow refinement, because this causes additional splitting of the gas cells. Once the grouping of cells in each Machine is completed, the next task is to allocate the number of cells to be computed in GPU and CPU in each Machine.

5.3.4 Load Sharing between CPU and GPU

Load balancing between the CPU and GPU is done using a parameter called core factor. This factor decides the ratio of computations between CPU and GPU. The cell groups are assigned in the increasing order of data parallelism and cells that are least data parallel are assigned to the CPUs first. The cell groups 1 & 2 are least data parallel and cell group 7 and 8 are highly data parallel. The cell groups 1 & 2 are boundary partial cells and boundary air cells respectively which have to implement the boundary condition additionally. This boundary condition implementation function is a lengthy code which has many branching statements and which are not well suited to GPU computations. It was seen that when this group of cells were allotted to GPU for computations, the performance drastically dropped and hence all the boundary cells are first allotted to CPU cores for computation. The computational load on CPU is estimated using the following expression

$$Pcpu = (ncores*Cfac)/(ncores*Cfac +ngpu) \quad (5.1)$$

$$Pgpu = ngpu/(ncores*Cfac +ngpu) \quad (5.2)$$

where

Pcpu -Portion of computation done by all the CPU cores in a computing node
Pgpu - Portion of computation done by all the GPUs in a computing node
Ncores - Number of processor cores available in the computing node
Ngpu- Number of GPUs available in the computing node
Cfac - The ratio of the computation load done by a CPU core to that of a GPU called core factor

In the above expression the core factor is the unknown and is arrived at based on trial and error method. It was found that a core factor of 0.015 (for every 1000 cells to be computed in GPU, 15 cells would be computed in CPU) was found suitable for the present GPU based code which gave good performance for wide range of geometry and flow conditions. While performing load balancing between CPU and GPU it is ensured that GPU which has a very large computing power would

never be idling and hence under situations when perfect load balancing is not possible to achieve between CPU and GPU, the GPU would be slightly overloaded.
For the load balancing to work effectively in the problems in which the number of less data parallel cells is more, an additional handling is introduced. If any of the cells other than groups 7 and 8 (partial and air cells which are not on the boundary and whose neighbour and neighbour's neighbour are not split) falls in GPU, the core factor is increased to a maximum value of 30%.

5.4. Programming and Algorithmic Aspects of GPU Computations for a Cartesian Mesh Solver

To obtain the best performance from the GPU, the programming and algorithms need to be tuned to the GPU architecture. In this regard, the handling of recursive data structures, typically used to traverse from the parent to the child of an oct-tree Cartesian cell, the memory management aspect of CPU to GPU copying and efficient use of memory hierarchy of GPU are some of the important aspects of computation with GPU accelerators.

5.4.1 Handling recursive data structure of Cartesian mesh

Cartesian grid is stored in a recursive data structure based on oct-trees. In-order to make it GPU capable, the cells need to be accessed in a non-recursive manner. The algorithm traverses each oct-tree and stores the pointer to the cells in a linear array. For each cell to remember its location within the tree a 32 bit integer was used. 3 bits are required per level for identification (One bit in each dimension). Additionally a bit is required to identify the leaf. Since trees up to a level 7 are used, 32 bits are sufficient to identify each cell. Additionally cell pointers to the neighbours are stored in each cell. Even though each cell could have 6 to 24 neighbours, only 6 pointers are used to store them. This is done by only storing the cell pointers with the same or lower level. For example if a cell is having 4 neighbours in one face, it will store the pointer to the parent cell of the neighbours. Figure 5.16 shows how a cell is identified

in its oct-tree. Each quad in the identifier represents the details of the cell in a particular level. First three bits represent X, Y and Z respectively and the fourth bit represents whether it has reached a leaf cell.

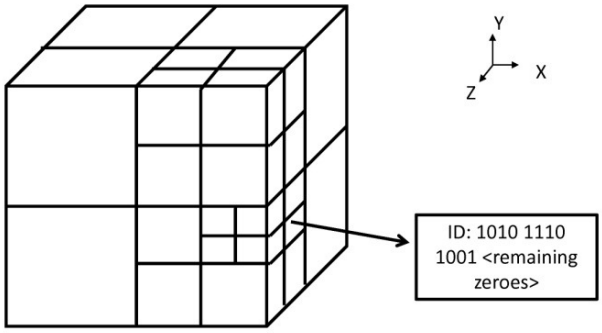


Figure 5.16 Schematic of oct-tree structure and bitwise identification of leaf cell

5.4.2 Programming Data Structure and Implementation for Parallel GPU Computation

After the domain decomposition, setting up communication links and grouping of cells with proper load identification in GPU and CPU, GPU and CPU threads are launched for computation. Number of CPU threads is equal to sum of number of CPU cores and number of GPU accelerators and number of GPU threads is equal to the number of GPU accelerators. In the case of a dual hex-core machine with 2 GPU accelerators, there would be 14 CPU threads and 2 GPU threads. The two extra CPU threads are meant to control the two GPU threads. The cell data structure used for the GPU based code and programming details is given in Appendix-1. After all the required information is copied from CPU to GPU, GPU kernels are launched which does computation in GPU cores for groups of cells. Computation in GPU is done on one dimensional grid with a certain number of thread blocks. Number of thread blocks is the number of cells of particular group divided by the number of threads in a block. For the present code, 128 threads in a thread block gave very good performance for a wide range of problems as compared to other values.

5.4.3 Thread Synchronization

After the CPU and GPU threads are launched, the first CPU thread becomes the master thread and initiates the computation of other CPU threads and the CPU threads that control the GPU threads which in turn initiates the GPU threads. Then the CPU thread 0 waits for the computation in all other threads to be over. Once the computation in a particular thread is over, the thread signals the completion to the master thread and falls in a lock position. Thus after the thread 0 gets the information that all other threads have completed their calculations, the communication process is carried out within and across machines. Then the next iteration starts after the master thread unlocks all the other threads by signaling to carry out the next iteration in the respective threads. This thread synchronization is carried out to ensure proper computations with proper updated values after each iteration and also to maintain the context of threads in GPU which would mean that the same thread would execute the functions for the next iteration without creating new threads. Creating new threads would mean repeated memory copy from CPU to GPU during each creation which is a time consuming operation and hence is avoided by maintaining the same GPU threads.

5.4.4 Memory Management Aspects

Before iteration can be started, updated value of the neighboring cells needs to be updated between CPU and GPU. For optimal performance, the communication should be done in minimum number of steps for the optimization of the process. To achieve this, the flow variables of cells are allocated in a contiguous memory. Each cell finds the location of its flow variables by storing the offset instead of the absolute address. As a result, the communication can be done between CPU and GPU in a single operation resulting in maximum performance. This also helps in maintaining single code for GPUs and CPUs. The whole array of trees is traversed and the group *id* is found out and marked. Then the cells are arranged in groups in the increasing order of their data parallelism into a linear array. Now the vector of flow variables is allocated and each cell stores the offset to its location within the flow vector.

Additionally only the necessary minimum is communicated between CPU and GPU. As a result of this, the communication between CPU and GPU is optimal both in quantity and the number of steps.

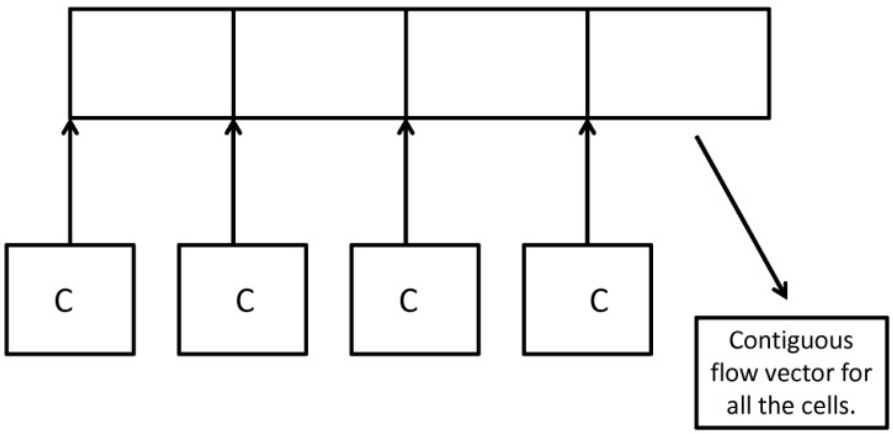


Figure 5.17 Schematic of memory layout

Figure 5.17 show the data layout of flow variables in the cells. Each cell stores the offset to the starting of its flow variable memory. The same memory layout is used in CPU as well as GPU. Hence the flow variable update between CPU and GPU can be done in a single update operation.

5.4.5 Effects of GPU Memory Hierarchy

GPU exposes its memory hierarchy to the programmers. There are three levels of memories namely registers, shared memory and global memory in the decreasing level of bandwidth. Shared memory in Tesla C2070 GPU is 64kB (Configurable as hardware managed or programmer managed). This was too low to be of any practical use as programmer managed. Hence for effectively using the memory bandwidth two activities were done. 48 kB of the shared memory was configured as hardware managed cache. It was found that the optimal usage of local variables is having

substantial effect in performance. In order to reduce memory access latency, the number of local variables was made to the minimum. As a result of this, there is a better chance for them to get accommodated in the registers or cache.

5.4.6 Solution Process Overview

Step-1 Domain decomposition is done by splitting the domain into rectangular sub-domains.

Step-2 Communication setup for distributed memory parallel computing via Message Passing Interface.

Step-3 Within each computing node, cells are arranged into groups with increasing level of data parallelism. The least data parallel cells are allotted to CPU which is computed using shared memory parallel computing by multi-threading. The remaining cells are allocated to GPU for groupwise computation of cells.

Step-4 All the necessary information required for computation is copied to GPUs.

Step-5 Start of computation in CPU and groupwise cell computation in GPU

Step-6 After the completion of each iteration of computations, the updated values are copied to the main memory.

Step-7 Communication is carried out between CPU and necessary information passed on to the corresponding GPU.

Step-8 The computation is again started in CPU and GPU as in step-5 and continued till convergence is attained.

5.5 Parallel Computing on a Cluster of GPU Machines

The parallel computing performance was tested for a typical flow problem on a cluster of dual quad core machines with 2 GPU accelerators in each machine. The cluster of GPU based machines named as SAGA (Super Computer for Aerospace with GPU Architecture) is a diskless cluster built with open source software components and in-house developments. The description of SAGA system is given in the next section.

5.5.1 Configuration of SAGA Supercomputer

SAGA supercomputing cluster consists of 368 diskless compute nodes of which 218 nodes have 2.4 GHz dual Quad core Intel processors and 2 numbers of GPU Nvidia-C-2070 accelerators and the rest 150 nodes have Intel Hex core processors with 2 numbers of GPU Nvidia-C-2090 accelerators each. The backbone of the cluster is a 40 Gbps Infiniband cluster interconnects. The theoretical peak computing performance is about 448 TFLOPS with each GPU accelerator of C-2070 type providing 515 GFLOPS peak and C-2090 type giving 570 GFLOPS peak. The Quad core CPU gives 38 GFLOPS and the Hex core deliver 55 GFLOPS peak performance. The peak power consumption of the cluster is about 253 kW. The cluster has a LINPACK performance of 188 TFLOPS from 320 nodes and ranked 86^{th} in the top 500 supercomputers of the world. Fig 5.18 gives the photograph of the Supercomputing cluster which is housed in the Sathish Dhawan Computing Centre of Vikram Sarabhai Space Centre at Trivandrum, India.

Figure 5.18 Photograph of SAGA supercomputing cluster (www.ISRO.org)

Figure 5.19 shows the layout of the SAGA supercomputer. There is a redundant fail safe Linux OS with DRBD (Distributed Relocated Block Device) and open source configured Heart Beat for the File Servers which are NFS (Network File System) over Infiniband. The compute node Linux Operating System is a tiny Linux Operating System occupying just 150 MB of RAM in the diskless node and has single image for all nodes.

The cluster resource manager running in the brain server is the key in-house software of the supercomputing cluster. The cluster resource manager allocates nodes on demand, powers it up on requirement, powers down when not required, and also manages the job queue. Daemon on the computing nodes spawns and terminates jobs on request from scheduler and reports job termination. The occupancy data is

maintained by the scheduler and the brain server is a failsafe server with redundancy. The biggest advantage of such a layout is the easiness to augment the computing facility to PetaFLOP scale by just adding more computing nodes and file servers as needed.

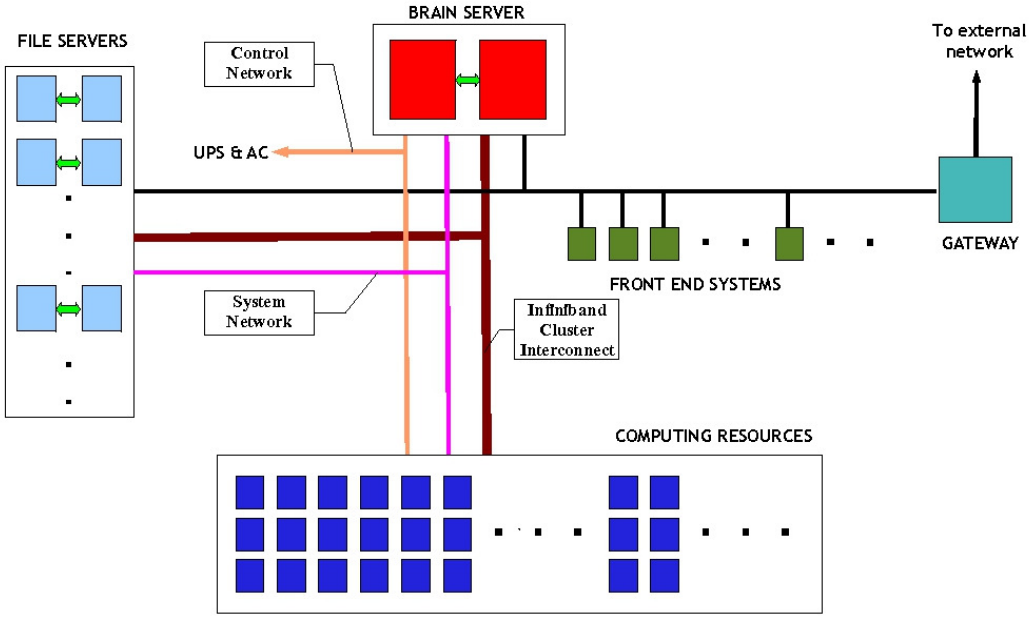


Figure 5.19 Layout of SAGA supercomputer (Sudhakaran et al. (2011))

5.5.2 Parallel computing performance with GPU accelerators

Parallel computing with tri-level parallelism, using MPI, Pthread and CUDA on multiple multi-core machines with GPU accelerators was carried out on SAGA for a typical launch vehicle configuration for perfect gas turbulent flow conditions. The problem was solved with three different grids. Figure 5.20 shows the speed up efficiency for the same problem on three different types of grids which have varying number of grids also varying number of split cells in the Cartesian mesh. The X axis shows the number of GPU based machine (each machine is dual quad core with 2 GPU accelerators of C-2070 type) used and the Y axis shows the speed up efficiency.

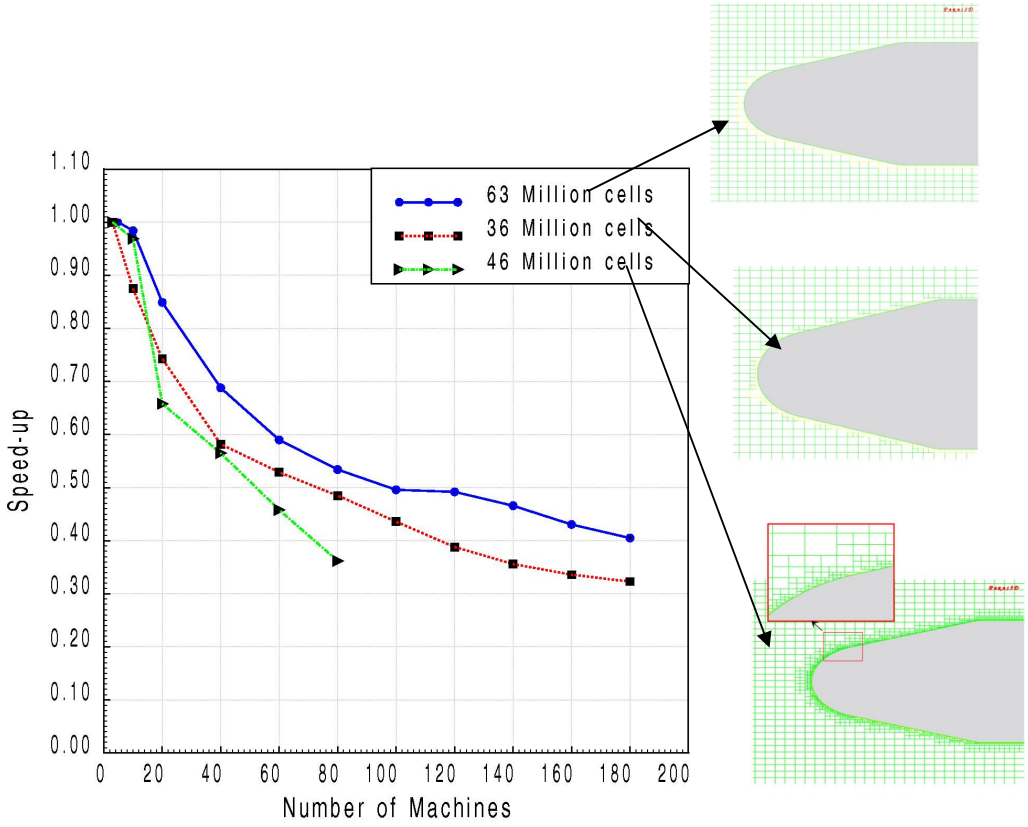


Figure 5.20 Speed up efficiency for a typical flow problem over a launch vehicle on cluster of GPU machines

Speed up efficiency for 'n' machines is the ratio of ideal time taken for computation for certain iterations (say 20 in the present case) in 'n' machines to the actual time taken in 'n' machines. Ideal time taken in 'n' machines is the actual time taken (for 20 iterations in the present case) in one machine divided by the number of machines. This can be expressed by the simple expression as given below

$$\eta_n = \frac{t_1}{n * t_n} \tag{5.3}$$

where η_n = Speed-up efficiency for 'n' machines

t_1 = Actual time taken for N iterations in 1 machine

t_n = Actual time for N iterations in 'n' machines.

The problem when solved with 63 million cells consisting of only basic Cartesian mesh without any split cells gave the best performance for more number of machines, since it has only 3 groups of cells, namely boundary air cells, partial cells and air cells whose neighbour and neighbour's neighbour is not split. This type of grid makes the computation highly data parallel. The second type of grid with 36 million cells had just one level of split Cartesian mesh. This has more types of cell groups as compared to the first type of grid. This gives lesser performance on more number of machines as compared to the first type of grid of 63 million cells because of lesser number of cells with less data parallel type of cells as compared to the first type of grid. The third type of grid used is 4 levels of split cells which would give rise to more number of cells other than group 7 and group 8 which are simple partial and air cells and thus less data parallel. It can be seen that, the performance rapidly falls after about 16 machines due to the fact that cells of non data parallel type had to be given to GPU for computations which gives poor performance. From this study it could bee seen that for a practical type of mesh as with the grid of 46 million cells, about 3 million cells per GPU machine gives performance above 85%.

Regarding the performance of the GPU accelerators for this problem, the problem speeded up by 4.5 times for the third type of grid with 46 million cells which is a practical type of mesh in one machine as compared to the case when GPU accelerators are not used. This speed up of 4.5X is against a theoretical maximum possible of 14X.

The performance of the parallel computation with GPU would depend on number of cells or the volume of computation and the data parallel nature of the cells. Even if the cells are highly data parallel, if the ratio of computation load to communication load is not large, the speed up would not be good as the communication would become a bottle neck. The present performance could further be improved by having more groups of cells which will make it more data parallel. This would mean that instead of neighbour being split as separate group as in the

present case, right side neighbour being split will be a separate group and left side neighbour being split would be yet another group and similarly top side neighbour split as one type and bottom side cell split as another type.

Another important feature associated with the present program is that the same code is used for computation on a pure CPU cluster and also for cluster of CPU machines with GPU accelerators. The program identifies whether GPU accelerator is present in the system and accordingly does the computation. The cluster of GPU based machines was used to compute tip-to-tail simulation of a typical Scramjet vehicle with combustion and is described in the next chapter.

CHAPTER-6

TIP-TO-TAIL SIMULATION OF FLOW OVER A TYPICAL SCRAMJET VEHICLE WITH COMBUSTION

Air-breathing engines have higher specific impulse as compared to other conventional propulsion systems like solid, liquid or cryogenic propulsion as it can utilize the atmospheric oxygen while operation and hence need to carry only the fuel which is either Hydrogen or Hydrocarbon based propellants. Scramjet mode of operation is the most suitable mode for air breathing vehicles operating at Mach numbers greater than 6 in order to get good propulsive power and efficiency. This is because, at Mach numbers greater than 6 if the velocities are brought down to subsonic conditions, as in the case of Ramjet engines, the temperature increase due to the large reduction in velocities from hypersonic to subsonic would be so large that the Hydrogen fuel injected would undergo dissociation instead of combustion. However, in the case of Scramjet mode of operations, the velocities after the air intake are maintained supersonic and hence the rise in temperature after air compression from the intake would be only large enough to cause ignition of the Hydrogen air mixture and not the dissociation of Hydrogen. For a Scramjet vehicle to deliver net thrust i.e. to obtain difference between the engine thrust and vehicle total drag greater than zero, the vehicle should have an airframe integrated Scramjet engine. In such type of vehicles, a portion of the air frame itself will act as intake as shown in figure 6.1. This type of vehicles with air frame integrated Scramjet engine experiments have been performed by X-43 flight experiment as reported by Voland et al. (2006). However such type of flight experiments would require special purpose launchers to put the air frame integrated scramjet vehicle at the required velocity and altitude which is quite involved and incurring huge costs. In order to evaluate the flight performance of a typical Scramjet engine, in a cost effective manner, sounding rocket experiments like that of Hyshot reported by Neal et al.(2005) is an alternative

although this would not necessarily yield net positive acceleration during the Scramjet phase of flight .

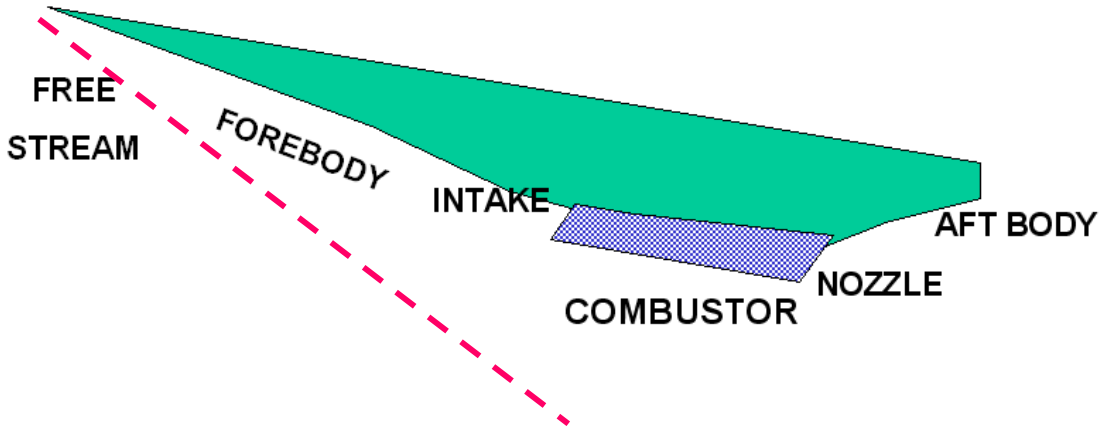


Figure 6.1 Schematic of air frame integrated Scramjet vehicle

In such type of flight experiments, the acceleration during the flight will be measured by the accelerometers. From the mass of the vehicle and change in the acceleration during the Scramjet phase of the flight; the thrust delivered by the Scramjet engines can be obtained.

The complete testing of the Scramjet engine in the ground is usually done in the open jet test facility and in the absence of this facility, only component level testing like the intake test and the combustor tests are possible. Hence computational fluid dynamics tool has to be used to predict the performance of the Scramjet vehicle before the flight, in the absence of open jet facility to obtain the performance of Scramjet engine. However CFD tool has to be validated against the results of component level tests like the air intake tests and the connected pipe mode tests of the combustor and with this confidence level, the tip-to-tail simulations with combustion in the Scramjet combustor of the Scramjet vehicle are performed so as to obtain the thrust delivered by the Scramjet vehicle with the external as well as internal flow.

Although full engine simulations have been reported in the literature by Kodera et al. (2003) and Gaitonde et al. (2010), the simulations performed are for body fitted meshes on conventional CPU machines. Such full engine simulations with Cartesian meshes employing the latest GPU based parallel computing platforms is not reported in the literature to the best of our knowledge. Considering the tremendous advantage of Cartesian meshes for automated grid generation and high cost effective through put obtained from the latest GPU based parallel computing, a full engine simulation of representative Scramjet vehicle is carried out with Cartesian mesh and are presented in this chapter.

6.1 Description of the problem

A Scramjet engine with three struts which has fuel injection from the base of the struts is chosen for the demonstration of full engine simulation. Figure 6.2 shows one half of the representative Scramjet vehicle with the engine. The figure actually represents the body view of the vehicle after Cartesian grid generation. The grid is generated by using a mesh of 180X120X120 with three levels of oct-tree refinement which has resulted in good capture of the body as seen from the figure. As seen in figure, the engine is attached to a representative cone cylinder forebody.

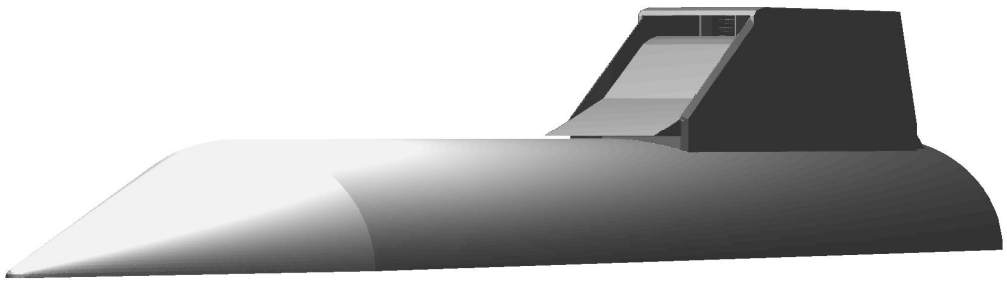


Figure 6.2 Typical Scramjet vehicle with Scramjet engine module

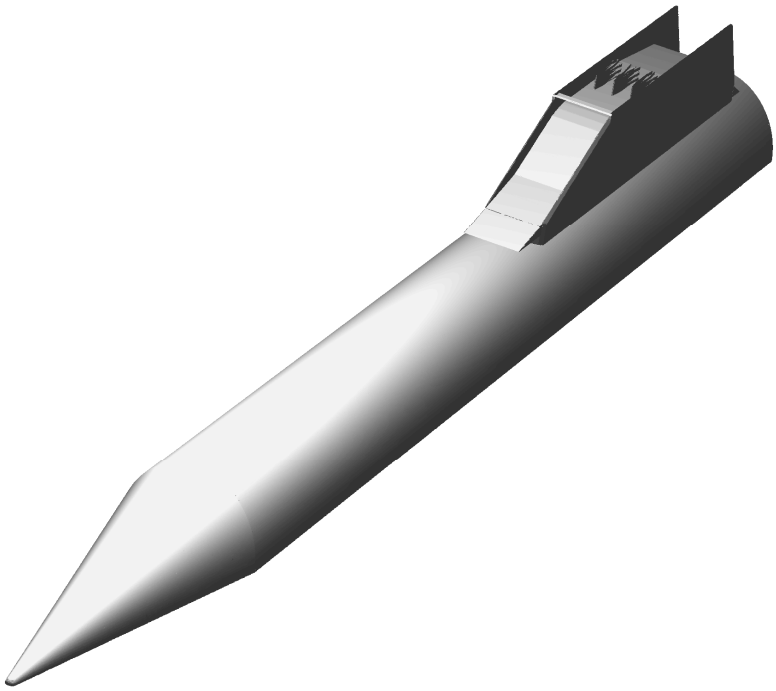


Figure 6.3 Representative Scramjet vehicle showing the engine module with three struts

Figure 6.3 shows the Scramjet vehicle with the top cover of the engine module removed in order to show the three struts in the engine. Figure 6.4 shows the view of section in the symmetry plane at z=0. Figure 6.5 shows the section at X=6.34 m. Figure 6.6 shows the section at Y=0.55 m which shows the injection ports at the end of struts. Computations are carried out for a free stream Mach number of 6.5 and free stream pressure of 2030 Pa at an altitude of 26 km at zero angle of attack. Fuel is injected from the strut base through holes of 6 mm diameter. Computations are carried out for an air fuel equivalence ratio (ER) of 0.6 and 1.0. Simulations are carried out to obtain pressure, temperature, water vapour mass fraction and Mach number distribution along the engine. Also the combustion efficiency and the thrust delivered by the Scramjet engine are also obtained. All the simulations are carried out on cluster of machines with GPU accelerators.

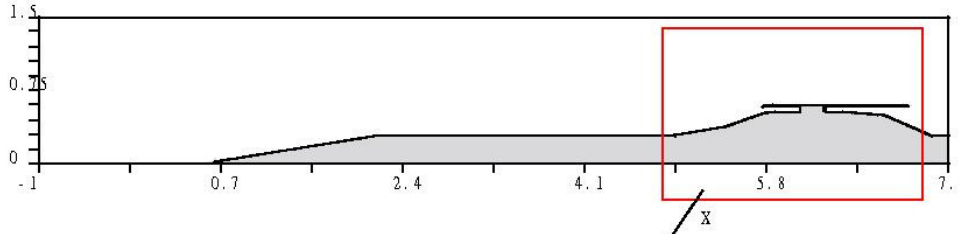


Figure 6.4 Body at section Z=0

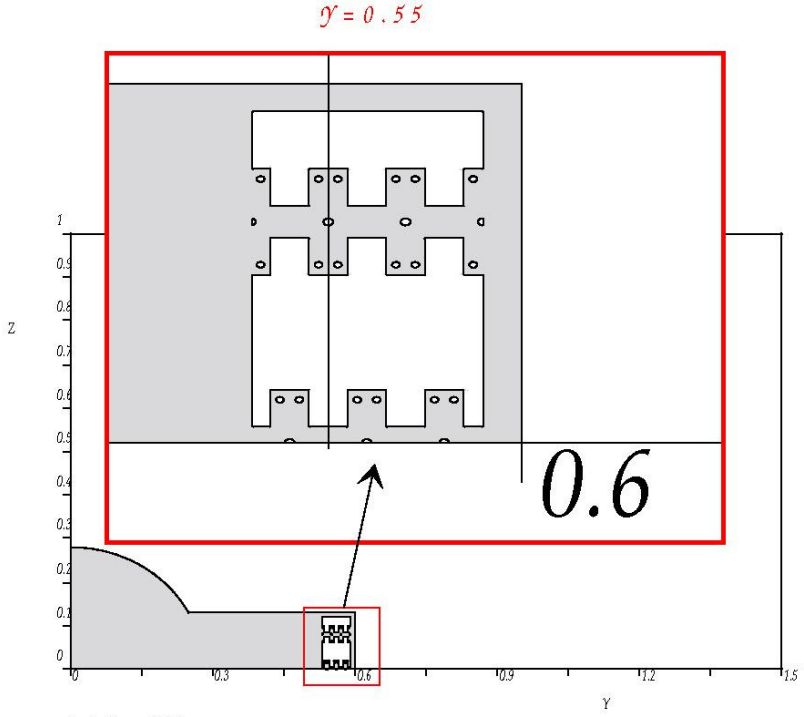


Figure 6.5 Section view at X=6.34 m with zoomed view in the inset

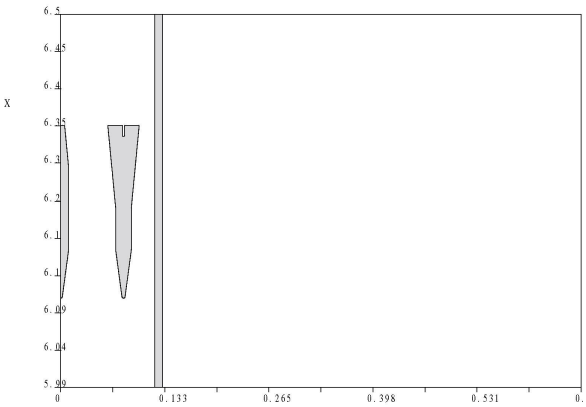


Figure 6.6 Body at section Y=0.55 m

6.2 Results and Discussion

A mesh of 180X120X120 with 3 levels of oct-tree splitting for the cells near the body is employed for proper body capture. The initial number of cells is 4.8million and the simulations are carried out at zero angle of attack for one fourth the body. At Ymin and Zmin boundary, symmetry conditions are imposed and supersonic inflow conditions imposed at the Xmin boundary. For all other outer boundaries the supersonic outflow conditions are imposed. The modified wall function approach described in Section 4.4.1 is used to get the wall effects due to turbulence on the flow

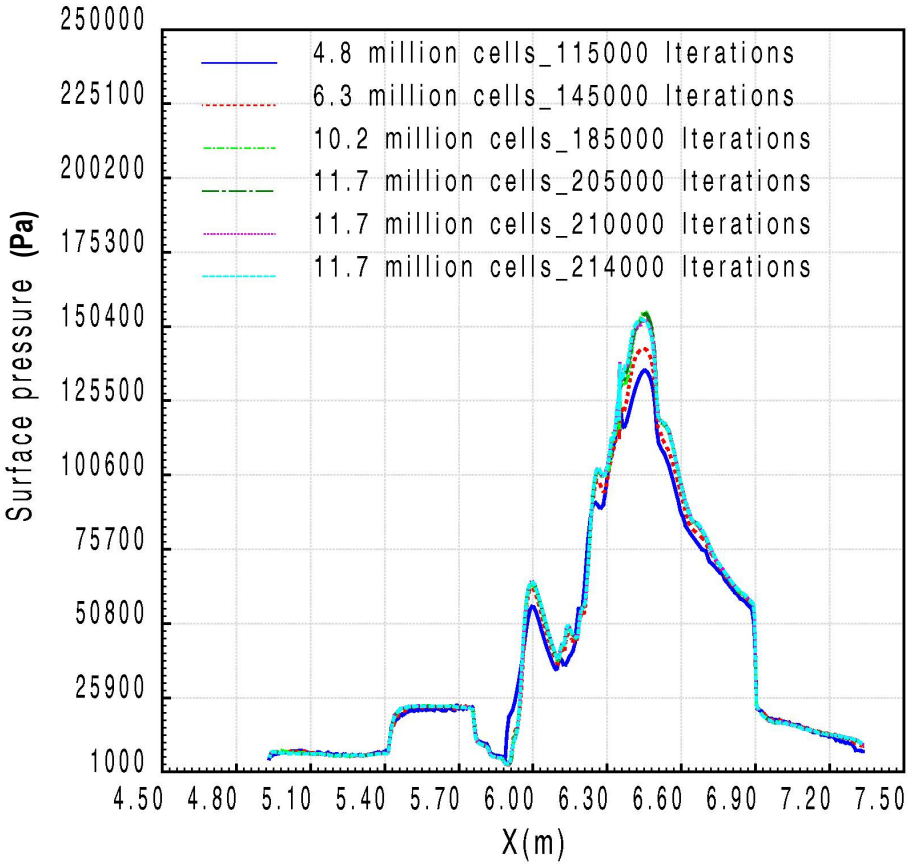


Figure 6.7 Grid independence plot for surface pressure along the centerline between two struts along bottom wall (ER=0.6)

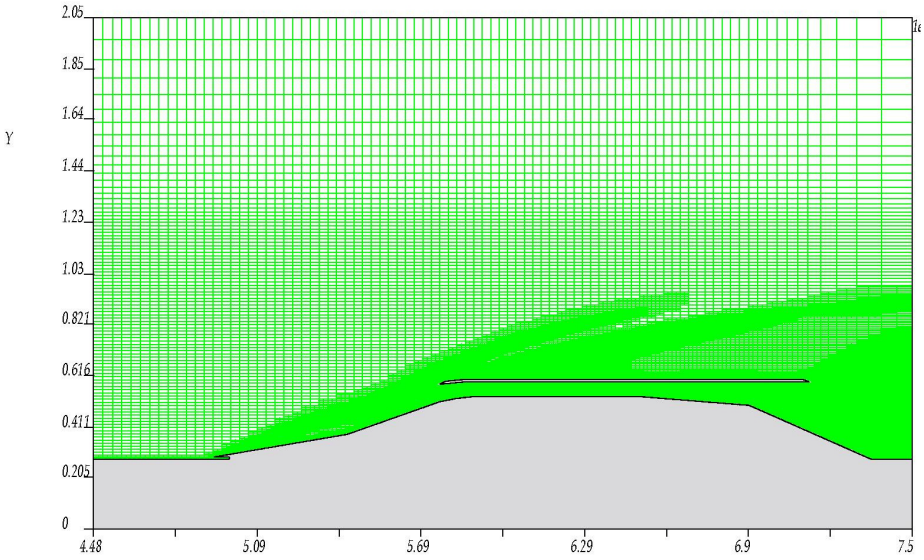


Figure 6.8 Final grid at section Z=0.0 in the Scramjet region after 3 levels of refinements

. Figure 6.7 shows the convergence and grid independence for the pressure plotted along the bottom wall of the Scramjet engine. Three levels of flow refinement is carried out based on the flow gradients and the total number of cells was increased from 4.8 million cells for the initial grid to 11.7 million for the final grid. It can be seen from figure 6.7 that the pressure distribution along the length of the Scramjet vehicle is independent of grid and iterations. Figure 6.8 shows the final grid in the region of Scramjet engine region which clearly shows the fine grid due to mesh refinement.

Figure 6.9 shows the Mach number plot over the complete vehicle at section Z=0.02 m (section in between the struts) from tip–to-tail simulation with combustion. All the features like the nose shock, shocks at the intake ramp and expansion at the nozzle are captured. Figure 6.10 shows enlarged view of the plot near the Scramjet engine region. It can be seen that at the intake, the two stage compression through first and second ramp takes place through the oblique shocks which almost touch the cowl lip causing very less spillage. Figure 6.11 shows the mass average total pressure

plotted along the length of the Scramjet engine. The mass averaged quantity, at an axial section of the engine, is the ratio of sum of the product of the mass flow rate across each cell and the quantity under consideration to the total mass flow rate across the section.

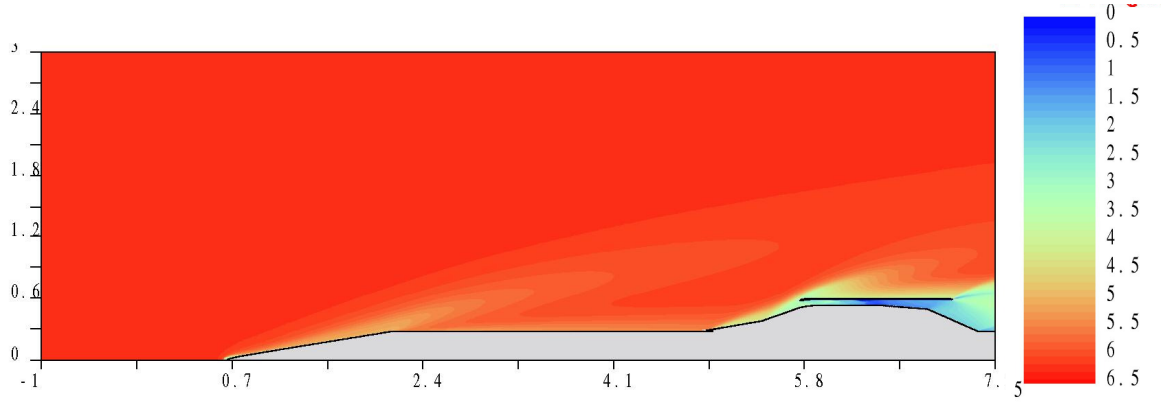


Figure 6.9 Mach number plot at section Z=0.02 m over complete vehicle from tip-to-tail simulation

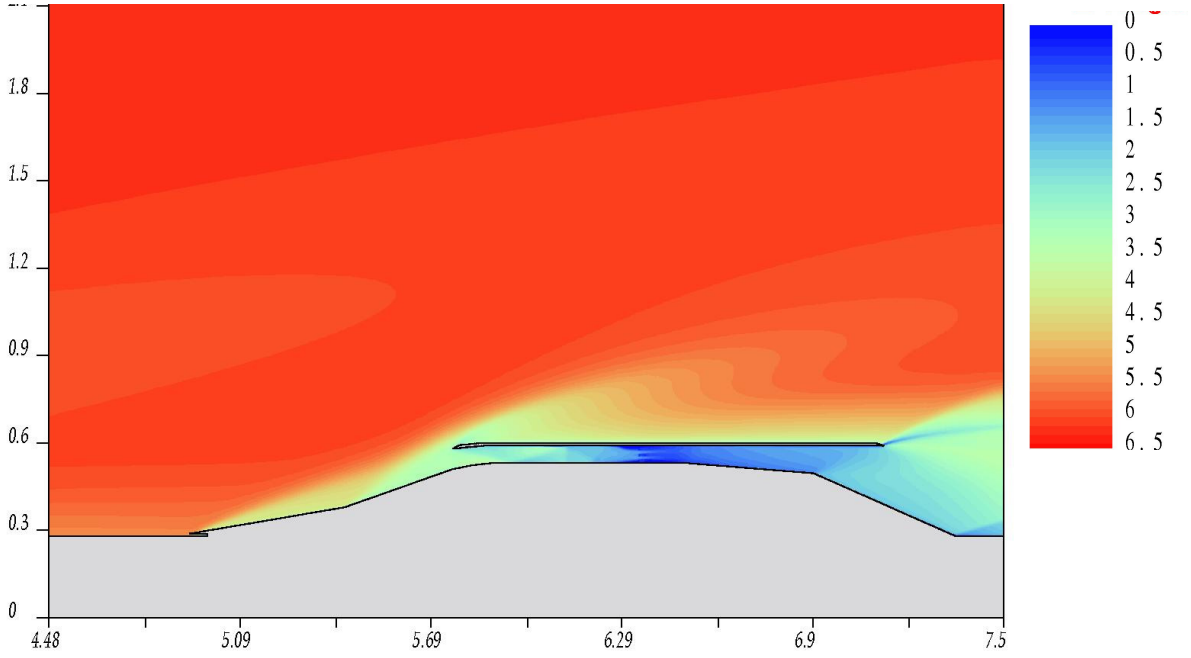


Figure 6.10 Mach number plot in the Scramjet engine region at section Z=0.02 m

The mass-averaged quantity of a flow variable Q at a section can be expressed by the following formula.

$$Q_{mass\_averaged} = \frac{\sum_{i=1}^{ncells} \dot{m}_i Q_i}{\sum_{i=1}^{ncells} \dot{m}_i} \qquad (6.1)$$

where $ncells$ is the number of cells in the section and $\dot{m}_i$ is the mass flow rate across the i^{th} cell. It can be seen that at the start of the intake the total pressure is around 30 bar and after the boundary layer splitter and two compressions from the two ramps the mass averaged total pressure drops to around 14 bar. Later the total pressure drop occurs due to the shock losses from the strut leading edge giving rise to total pressure at the combustor entry of around 5 bar. After the combustion process, the total pressure drops further by around 2.5 bar.

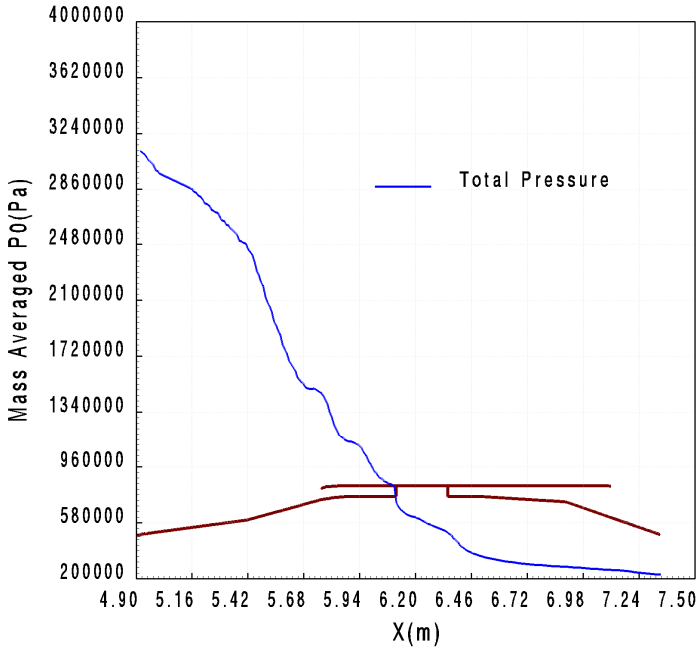


Figure 6.11 Mass-averaged total pressure along the length of the engine (ER=0.6)

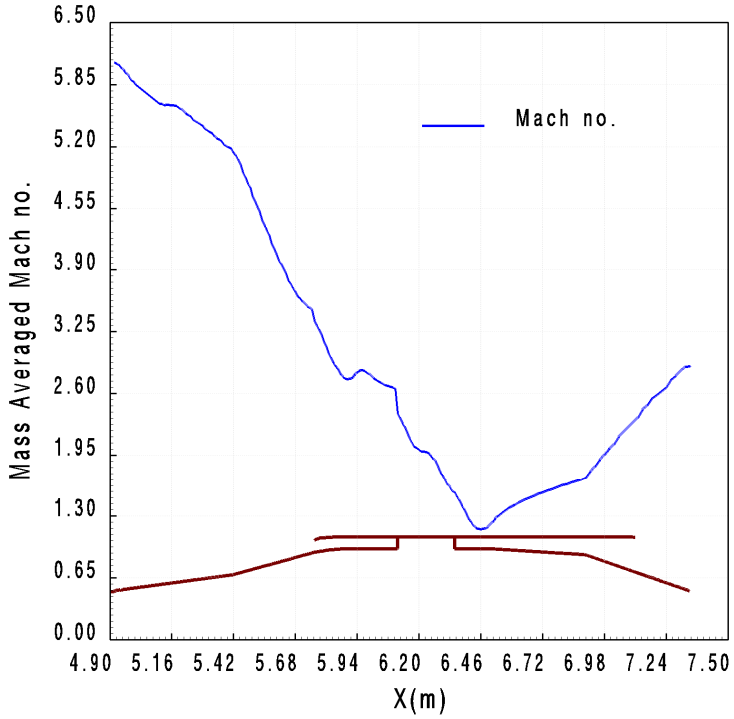


Figure 6.12 Mass-averaged Mach number along the engine (ER=0.6)

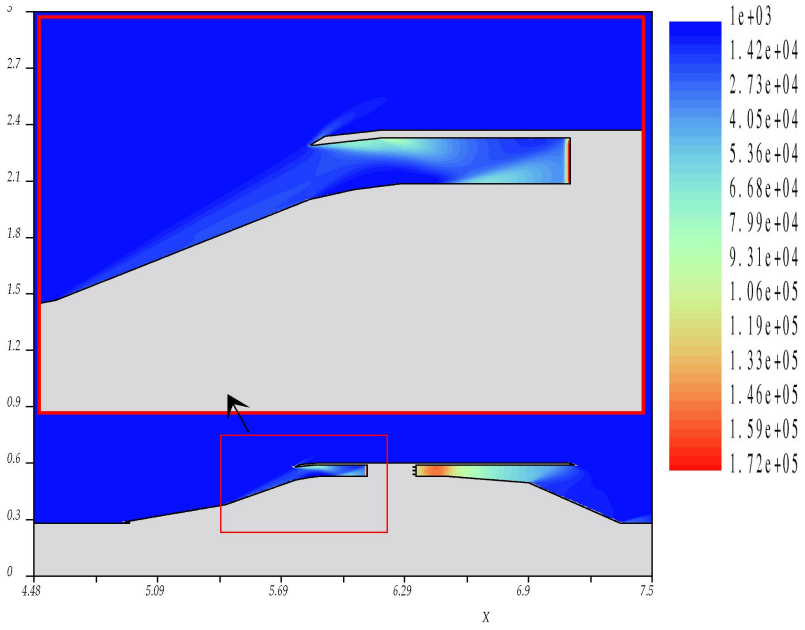


Figure 6.13 Pressure distribution in the Scramjet engine with combustion (ER=0.6) at section Z=0 showing the zoomed view of the intake shock system in the inset

Figure 6.12 shows the mass averaged Mach number along the Scramjet engine. The Mach number drops sharply due to the shock induced compression from the two ramps and the combustor entry Mach number is around 2. At the base of the strut, the combustion takes place with the mass averaged Mach number just above 1.0, indicating supersonic combustion. Later due to the divergent portion of the combustor and the nozzle, the Mach number increases.

Figure 6.13 shows the pressure distribution in the Scramjet engine at section Z=0 with zoomed portion of the intake showing shock reflection from cowl. The increase in pressure due to combustion at the downstream of the strut can be very clearly seen. No intake un-start is noticed due to the combustion as seen in the above figure. Figure 6.14 shows the water vapour mass fraction at section Z=0 and figure 6.15 shows the water vapour mass fraction at Y=0.559 m. It can be seen from Figure 6.15 that most of the combustion is in the axial region downstream of the strut. This indicates that there is still a region in between the strut in the combustor which can be further used for fuel injection and subsequent combustion which of course should not give rise to very large blockage or high pressure rise to cause intake un-start. Figure 6.16 shows mass flow rate of hydrogen along the combustor starting from the strut end. It can be seen that most of the hydrogen is consumed just downstream of the strut which is the mixing and combustion dominated zone. About 10% of the hydrogen is remaining un-burnt towards the combustor end.

Figure 6.17 shows the cumulative combustion efficiency which is the ratio of total amount of water vapour formed up to the axial location to the total ideal amount of water-vapour that would be formed assuming full combustion. Most of the combustion takes place just downstream of the strut and later the combustion efficiency remains almost constant, indicating that further combustion of small quantities of left over hydrogen is not taking place due to very high velocities involved. Figure 6.18a shows the pressure distribution along the centerline between two struts of the bottom wall i.e. at section of z-symmetry plane of the engine for

equivalence ratio of 0.6. The free stream Mach number for the Scramjet vehicle is 6.5 and after the bow shock ,the flow field that approaches the first ramp of the intake after the cone cylinder fore body portion is about Mach number 6 and undergoes the first compression over the 10.5 degree ramp followed by second compression in the second ramp which also has 10.5 degree wedge angle as shown in Fig. 6.13. Thus the flow from the horizontal has turned by 21 degrees. The shock from the second ramp impinges on the cowl lip and at the cowl the flow initially turns by 15 degrees and then subsequently by another 6 degrees at the end of drooping portion of the cowl. The turning of the flow by 15 degrees at the cowl tip will create a shock and further turning by 6 degrees introduces another weak shock. The reflected shock from the cowl lip impinges on the bottom wall causing the pressure rise at 6m location. It is to be noted that there is also expansion waves emanating from the expansion corner after the second ramp which interact with reflected shock and forms complex wave systems. The reason for the sharp pressure jump at 6m location is due to the reflected shock from the cowl lip. After the pressure jump due to the reflected shock, the pressure reduces as it tries to recover and then at little downstream it again rises due to the shock-shock interaction caused by the two strut leading edges as shown in figure 6.18b. Subsequently the strut geometry causes pressure rise and further pressure rise occurring near the strut base is due to combustion. The high pressure flow due to combustion later expands through the divergent portion of the combustor and the nozzle as shown in Fig.6.18a. Figure 6.19 shows the mass averaged static temperature which clearly shows the rise in temperature due to compression in first ramp and second ramp. This is followed by further increase due to shock from strut leading edge, which will make the temperature at the base of the strut more than the ignition temperature of hydrogen-oxygen mixture.

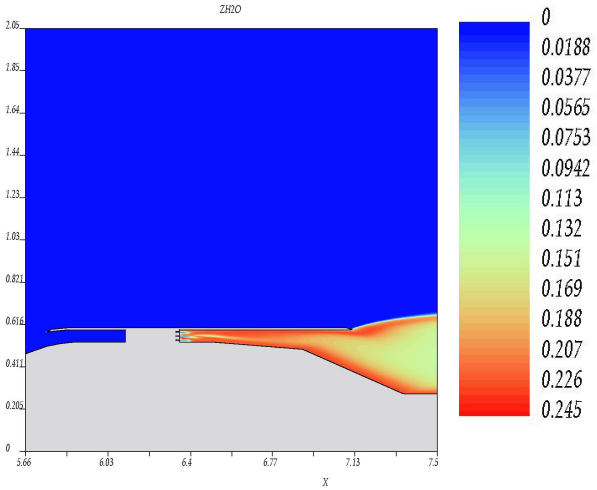


Figure 6.14 Water vapour mass fraction at section Z=0. (ER=0.6)

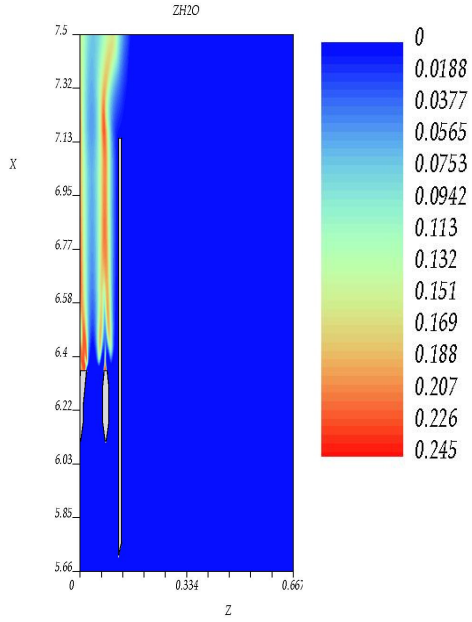


Figure 6.15 Water vapour mass fraction at section Y=0.559 m (ER=0.6)

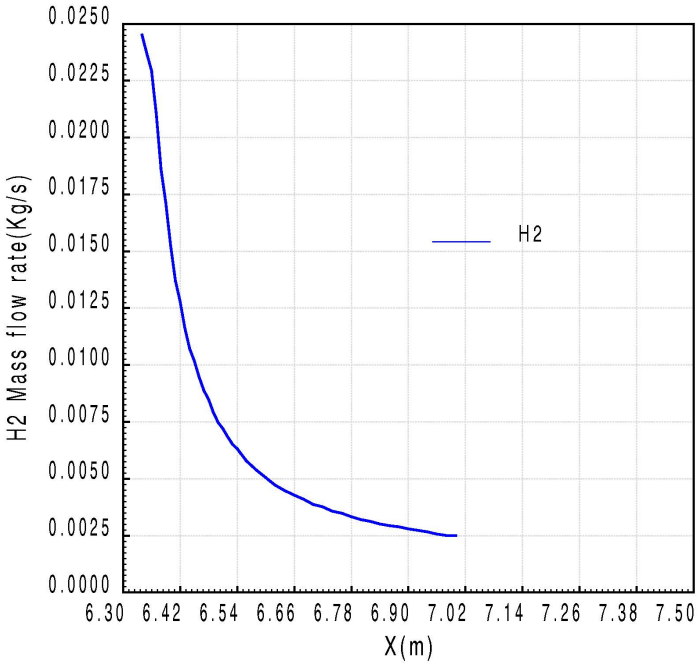


Figure 6.16 Mass flow rate of hydrogen along the engine after the strut base (ER=0.6)

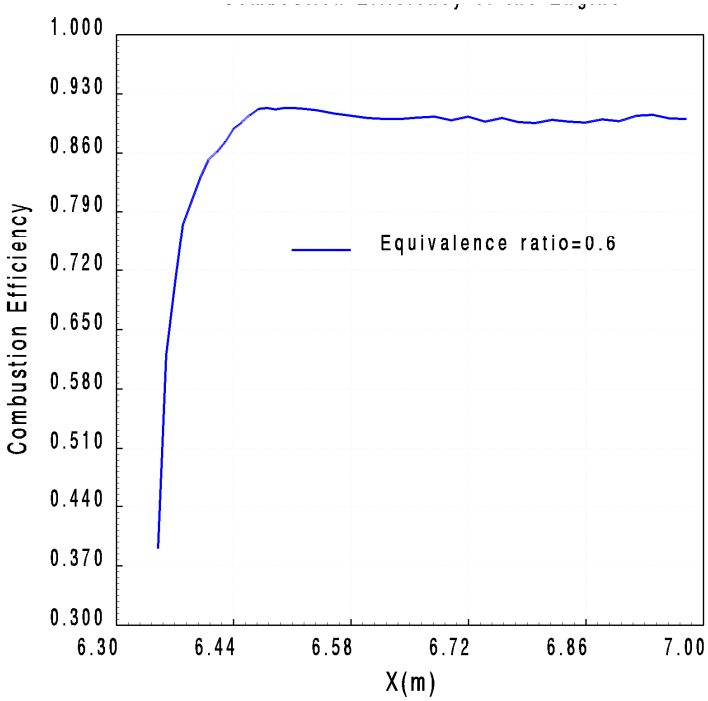


Figure 6.17 Combustion efficiency along the combustor for equivalence ratio of 0.6

(Pa)

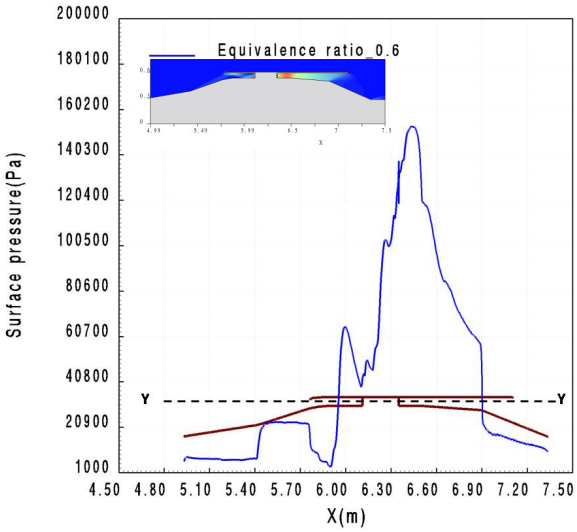


Figure 6.18a)Pressure distribution along the centerline between two struts of bottom wall for equivalence ratio of 0.6

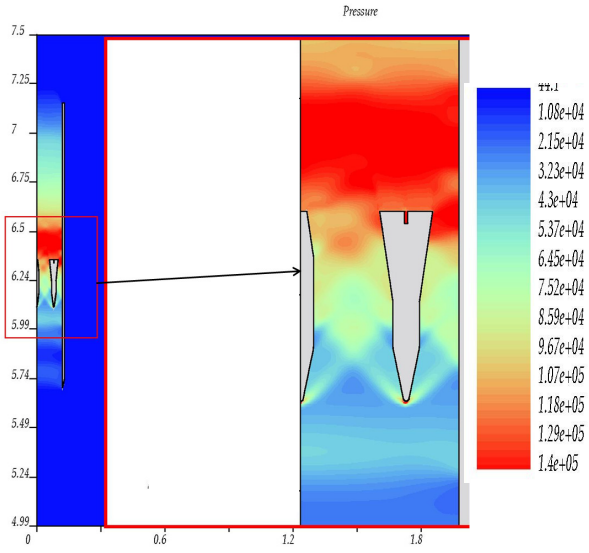


Fig 6.18b) Pressure at Section YY=0.55 m

.

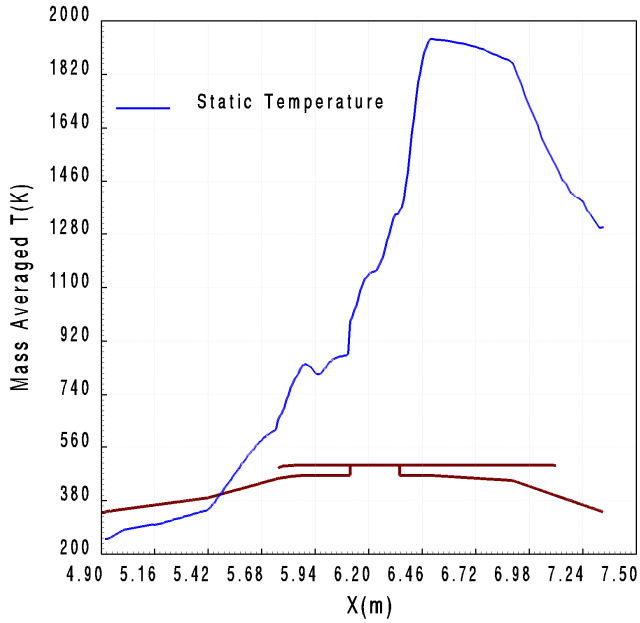


Figure 6.19 Mass-averaged static temperature along the engine for equivalence ratio of 0.6

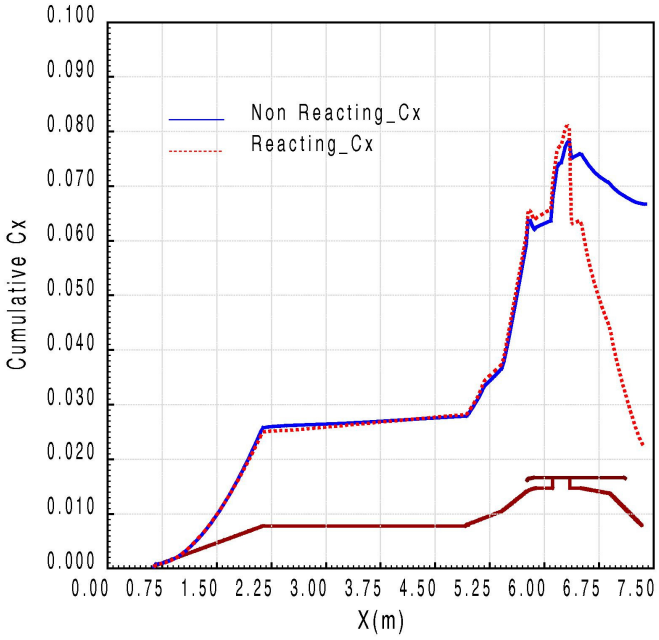


Figure 6.20 Cumulative axial force along the length of the Scramjet vehicle for equivalence ratio of 0.6

Sharp rise in temperature is noticed due to combustion which then reduces due to expansion in the nozzle. Figure 6.20 shows the cumulative axial force coefficient (C_X= Axial Force /(0.5*$\rho V^2 S$)) for an equivalence ratio of 0.6 compared with that of the non-reacting case. The behaviour of the curve is very much on the expected lines, with the reacting and non-reacting curves behaving the same way till the start of the strut and further after the strut, the reacting case gives a sharp drop in cumulative axial force, indicating thrust being delivered by the Scramjet engine. However the overall axial force with Scramjet operation is still a small positive number indicating that the vehicle would not have a net positive thrust and hence would be decelerating.

Figure 6.21 shows the mass-averaged Mach number along the engine length for equivalence ratio 1.0. The Mach number keeps falling through the intake with a small increase at the end of the second ramp because of the expansion and then keeps further falling due to the presence of struts. At the strut base, the Mach number is

minimum because of low recirculating flow which is the active zone for combustion. Increase of Mach number downstream of strut is due to the divergence of the combustor followed by expansion through the nozzle. Figure 6.22 shows the mass-averaged temperature along the engine for equivalence ratio 1.0 which is almost the same as that of equivalence ratio 0.6 except that the peak temperature extends for slightly larger region because higher fuel flow rate. Figure 6.23 shows the total pressure plot along the engine and figure 6.24 shows the pressure along the centre line of the engine. As expected, the higher equivalence ratio gives a larger pressure rise. Figure 6.25 shows the mass flow rate of hydrogen downstream of the strut which clearly shows 50% consumption of the injected mass flow within about 15 cm from the strut base.

Figure 6.26 shows the body pressure palette of the engine showing all the features like pressure rise in second ramp, footprint of the leading edge shock of the strut and rise in pressure due to combustion at the rear of the strut. Figure 6.27 shows the centerline pressure distribution between two struts for reacting and non-reacting cases. As expected, the pressure for reacting and non-reacting cases are same up to the region near the strut base, after which the rise in pressure due to combustion is clearly visible and higher pressure rise noticed for higher equivalence ratio.

Figure 6.28 shows the cumulative axial force distribution for equivalence ratio 0.6 and 1.0 compared with non-reacting case. The equivalence ratio of 1.0 gives more engine thrust as compared to equivalence ratio 0.6, due to more fuel injected. It is to be noted that, for ER 1.0, the plot shows a small net positive cumulative axial force coefficient indicating that the vehicle will still be in a deceleration mode. The net axial force from simulation (Thrust-Drag) for equivalence ratio 0.6 is -620 N for equivalence ratio 1.0 the value is -324 N. The change in deceleration can be measured in flight and can also be theoretically estimated by dividing the computed thrust of the Scramjet engine by the total mass of the Scramjet vehicle. It is to be noted that the present simulations are for a representative Scramjet vehicle without fins which

otherwise would usually be needed for providing static stability to the vehicle and this would also impart additional drag to the vehicle.

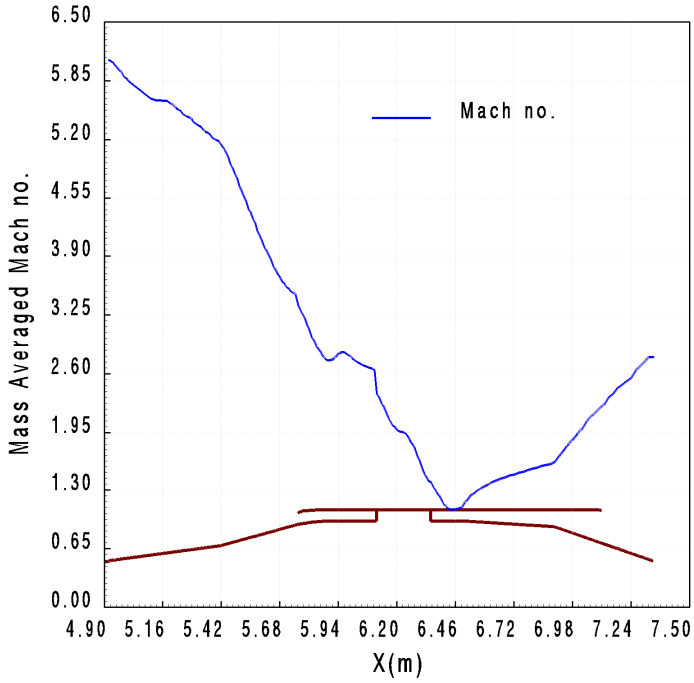


Figure 6.21 Mass-averaged Mach number along the engine for equivalence ratio of 1.0

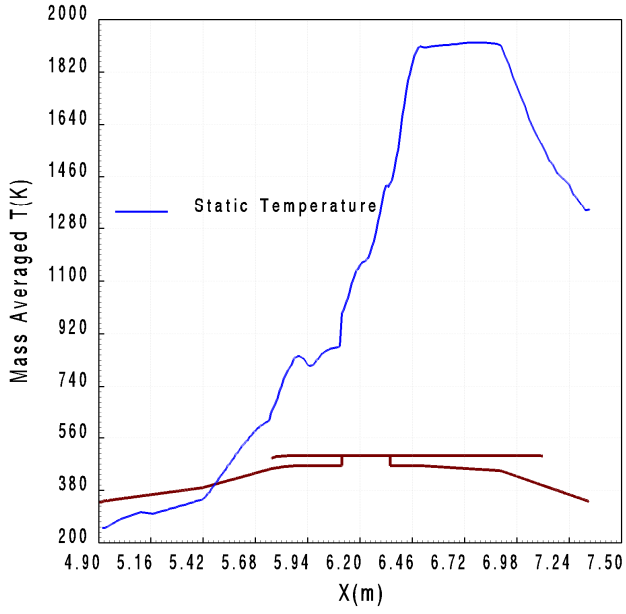


Figure 6.22 Mass-averaged static temperature along the engine for equivalence ratio of 1.0

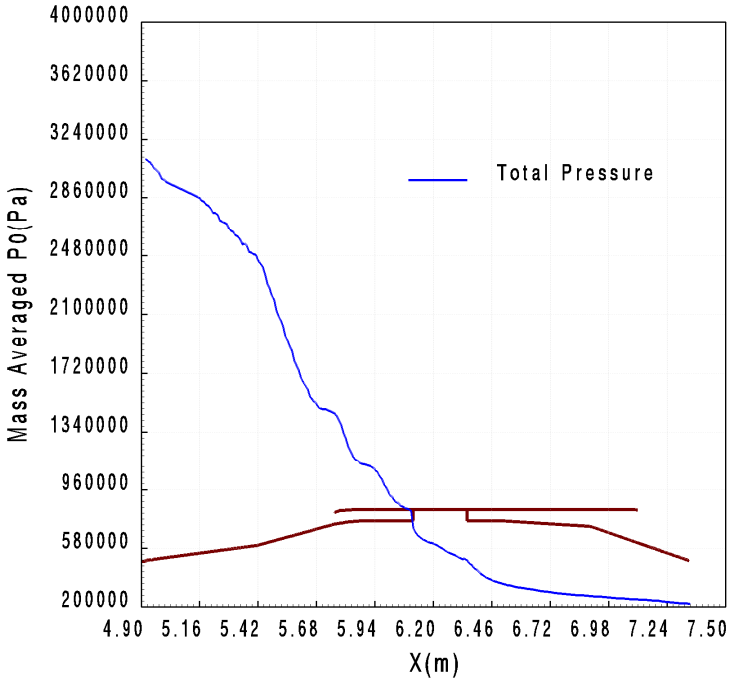


Figure 6.23 Mass-averaged total pressure along the engine for equivalence ratio of 1.0

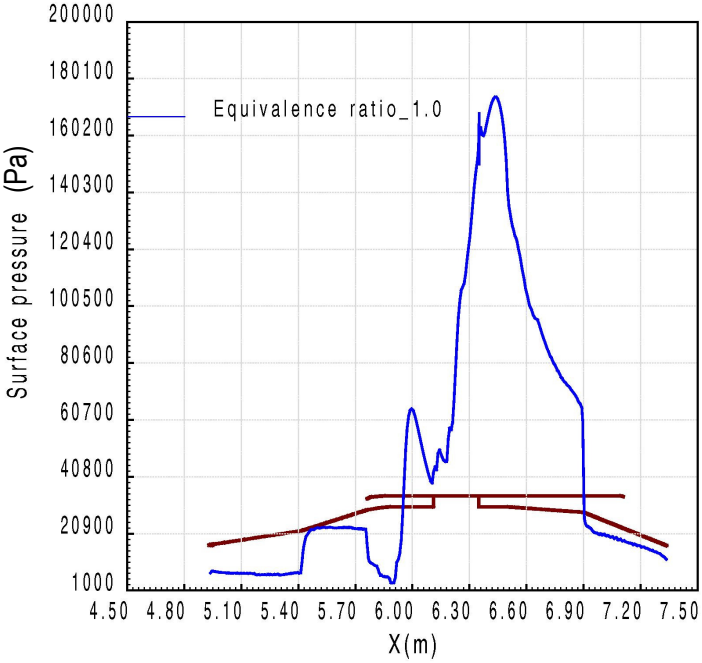


Figure 6.24 Pressure distribution along the centerline between two struts of bottom wall for equivalence ratio of 1.0

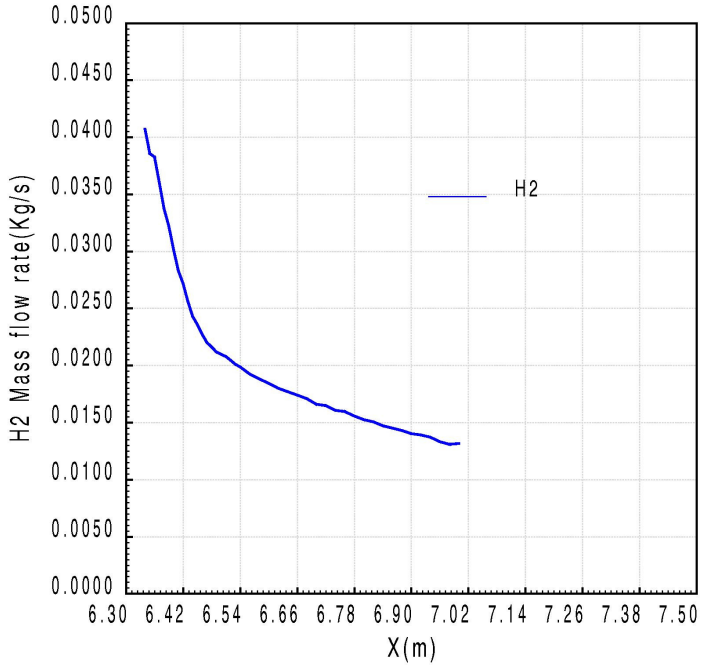


Figure 6.25 Hydrogen mass flow rate downstream of strut for equivalence ratio 1.0

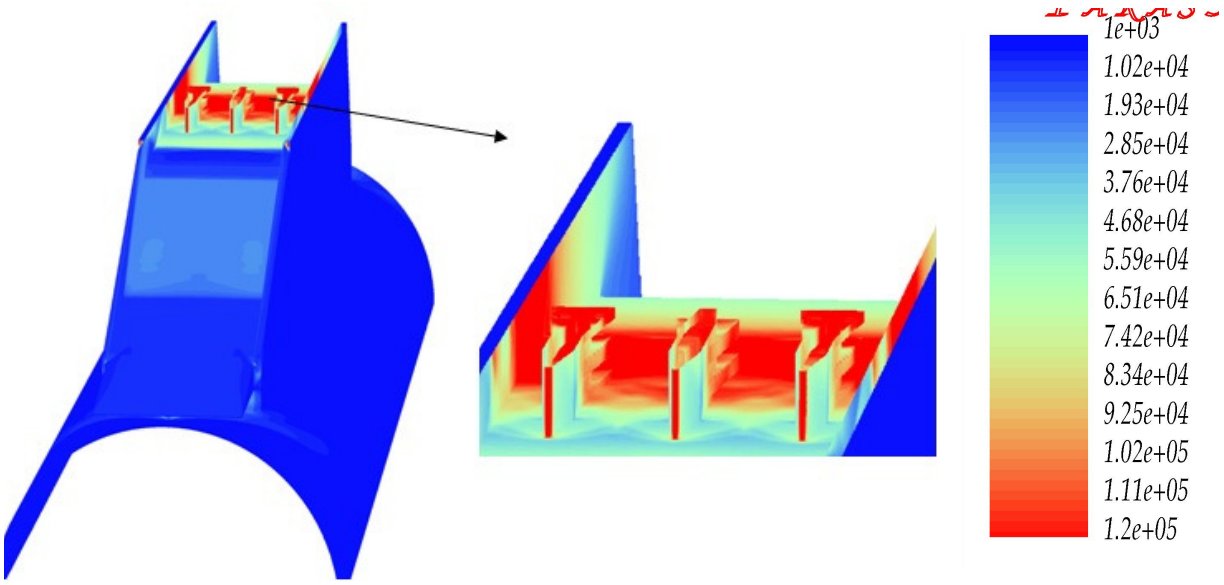


Figure 6.26 Body pressure of the Scramjet engine with three struts with enlarged view of the strut region for equivalence ratio =1.0

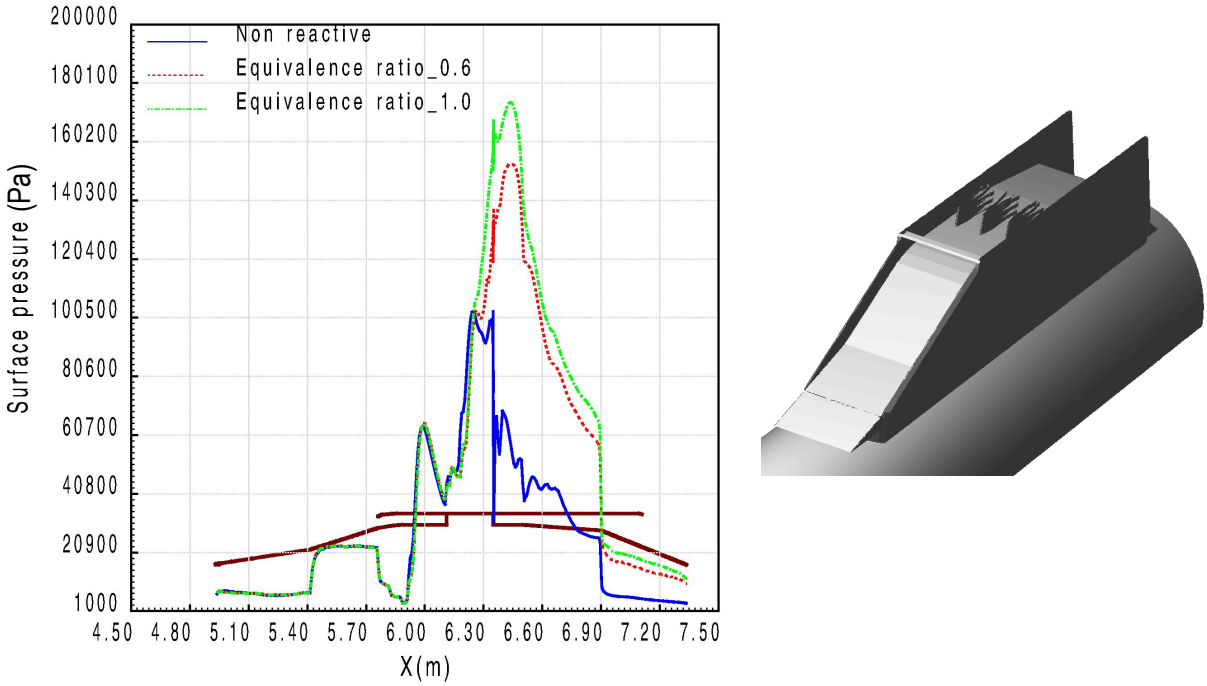


Figure 6.27 Surface pressure along the centerline between two struts for reacting and non-reacting cases

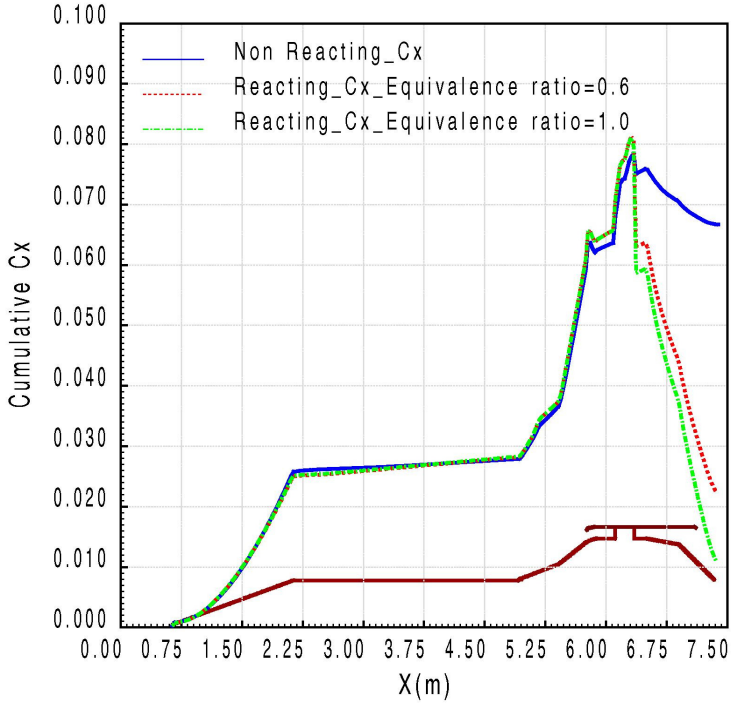


Figure 6.28 Cumulative axial force coefficient for equivalence ratio 0.6 and 1.0 compared with non-reacting case

6.3 Parallel Computing Performance of Tip-to-Tail Flow Simulations on GPU cluster

Parallel computing was carried out on a cluster of dual quad core machines with each machine consisting of 2 GPU accelerators having 512 cores each. Thus each machine has 8 CPU cores and 1024 GPU cores. In order to have good performance from the GPU cluster, the computational load has to be shared between GPU and CPU cores in such a way that GPU cores are allotted tasks which are highly data parallel. As described in section 5.3, the cells are classified into 8 groups which need to perform similar type of computations and computations are performed group wise.

For the present computations, the total number of cells after 3 levels of flow refinement are 11.81 million, out of which 6.56 million cells are gas cells and 1.77 million cells are partial cells and the rest are body cells for which no computations are performed and hence is not counted for computational load. Figure 6.29 shows different cell groups used for GPU computations.

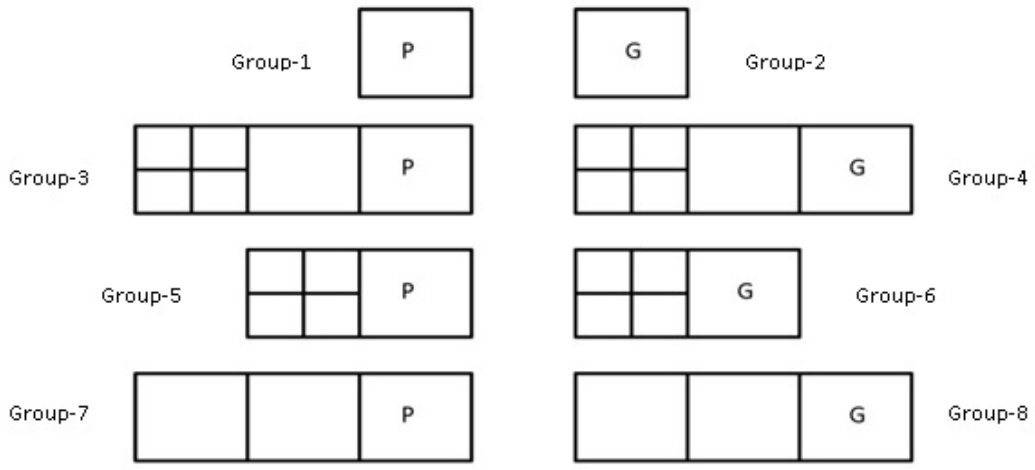


Figure 6.29 Different cell groups for GPU computation

Group-7 and Group-8 are groups of partial cells and air cells whose neighbours are not split and are highly data parallel groups. Most of the cells in the domain will fall in this group and are allotted to GPU for computation with maximum priority and on the other hand these cells are allotted to CPU with least priority. Group-3 and Group-4 cells are groups of partial cells and air cells whose neighbour's neighbour is split and are allotted to GPU with second level of priority. Group-5 and Group-6 are groups of partial cells and air cells whose neighbour is split. Such groups are allotted to CPU with a high level of priority and allotted to GPU only if the compute load in GPU is so less that GPU could complete the computation before CPU. Normally in GPU computation, it is always advisable to avoid idling of GPU as its computing power is much larger than that for CPU. Hence the computational load is distributed in such a way that if at all the load cannot be evenly distributed, always the GPU will be allotted the extra computational load to avoid idling of GPU. The last two cell groups are Group-1 and Group-2 cells which are the boundary partial cells and air cells and are the least data parallel groups and are allotted to CPU with maximum priority. Table 6.1 shows the cell group distribution between CPU and GPU for 2 machines and Table 6.2 shows the distribution in 4 machines.

Table 6.1 Distribution of cell groups between CPU and GPU in 2 machines

Machine Number	Type	Group-1	Group-2	Group-3	Group-4	Group-5	Group-6	Group-7	Group-8
1	CPU 386672	73450	20372	29253	34906	2906	42425	0	0
	GPU 3681816	0	0	142794	176433	0	0	908911	2453678
2	CPU 365219	18685	97621	48769	116431	2087	68734	0	12892
	GPU 3895048	0	0	0	0	0	0	538393	3356655

Table 6.2 Distribution of cell groups between CPU and GPU in 4 machines

Machine number	Type	Group-1	Group-2	Group-3	Group-4	Group-5	Group-6	Group-7	Group-8
1	CPU 205594	6696	122558	21378	24108	2410	28444	0	0
	GPU 1813403	0	0	92932	112265	0	0	554534	1053672
2	CPU 189295	5787	54940	18685	56204	1435	52244	0	0
	GPU 1944863	0	0	5380	13334	0	0	296315	1629834
3	CPU 190280	66754	81174	12670	15205	496	13981	0	0
	GPU 1859211	0	0	45067	59761	0	0	354377	1400006
4	CPU 190982	12898	42681	24704	46893	652	16490	0	46664
	GPU 1935127	0	0	0	0	0	0	242078	1693049

It can be very clearly seen that boundary partial cells and boundary air cells are Group-1 and Group-2 cells and are entirely allotted to CPU since these are least data parallel cells due to the additional code length and branching that occurs due to implementation of boundary conditions. On the contrary, the partial cells and the air cells which do not have any split neighbour i.e Group-7 and Group-8 are entirely allotted to GPU as they are highly data parallel. The Group-7 cells which are partial are less data parallel as compared to the Group-8 cells which are air cells due to the additional overhead of implementation of body boundary condition with wall function. It can be seen that the GPU load is about 10 times the CPU load which is also the ratio of the maximum theoretical speed of GPU to CPU.

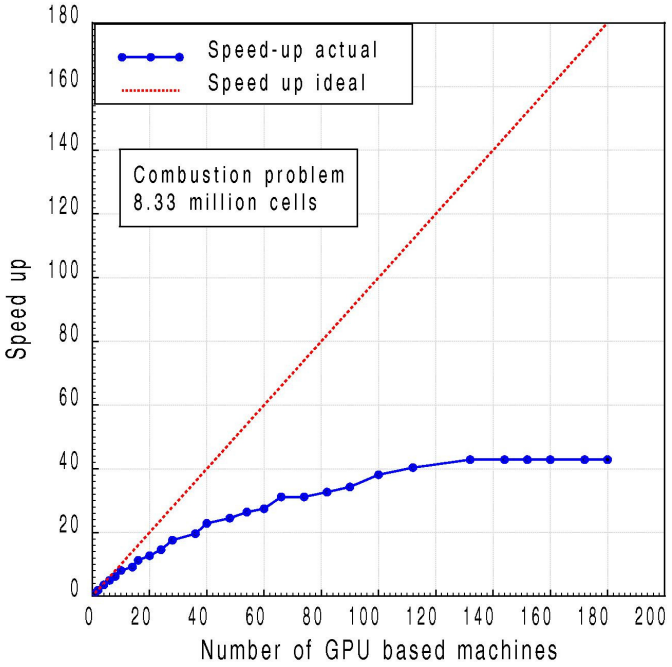


Figure 6.30 Speed up performance on 180 node GPU cluster

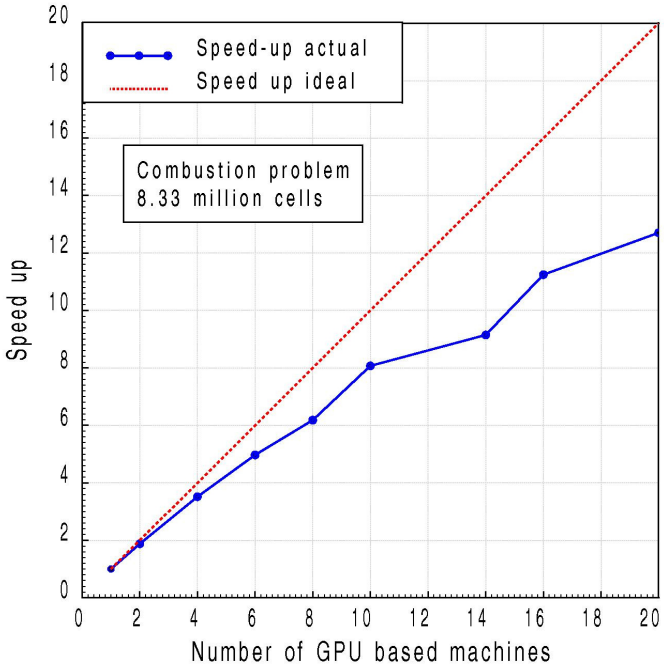


Figure 6.31 Speed up performance up to 20 GPU machines

Figure 6.30 shows the speed up performance up to 180 machines as compared to ideal speed up and Figure 6.31 shows the same figure plotted for speed-up obtained up to 20 machines for better clarity. It can be seen that for the present problem size of 8.33 million cells consisting of air cells and partial cells which would take part in the computation, the speed up efficiency beyond the use of 10 machines (each machine has 8 CPU cores and 2 GPU accelerator) is less than 80%. This means that for 10 machines, the ideal speed up is 10 times the single machine speed or in other words the time taken would be $1/10^{th}$ the time taken in a single machine for 100% efficiency. However we notice that the speed up for 10 machines is about 8 which means the speed up efficiency is 80%. It is to be noted that each machine has a theoretical peak speed of 1.2 TFLOP and the speed up efficiency of 80% is obtained for 12 TFLOP peak speed which is very good for the above 8.33 million size problem. The present problem could be completed with 214000 iterations in 36 hours on 10 machines. It is seen that beyond certain number of machines, the speed up remains almost constant. This is because, as number of machines is increased, the computational load on GPU reduces but the communication overhead for GPU to CPU and CPU to GPU copy process remains constant and hence no gain is obtained in speed up. With regard to speed up on a single dual code core machine with 2 GPU accelerators as against computation on dual code core without GPU accelerators, the speed up value obtained was 7.3X. as against the theoretical value of about 14X.

It can be inferred from the speed-up performance that for the above class of combustion problems, about 2 million cells per GPU machine would give a performance of more than 90% and for 1 million cells per machine the speed up efficiency will be more than 80%. Considering the space, cost and power advantage of GPU as compared to the CPU cluster, the computations using GPU cluster are highly beneficial and are ideal candidates for high performance computations.

CHAPTER-7
CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

Cartesian grid based approach to solution of high speed flow problems as applied to reentry type of vehicles and Scramjet combustion which are essential technologies for low cost access to space has been addressed in this work. The solution of reentry type of problems to obtain heat flux with a Cartesian mesh based approach is noted to be not reported in literature. The present approach followed is through near wall viscous resolution by a combination of unstructured prism layer solution near the wall and Cartesian mesh solution away from the wall. With regard to application of Cartesian mesh approach to Scramjet engine flows with combustion, along with high performance computing with GPU based systems which is also noted to be not reported in literature, the problem is addressed through pure Cartesian mesh with an available wall function approach. The highly compute intensive part is addressed through development of new methodologies and algorithms for high performance computing with GPU accelerators. We have also demonstrated its utility in the context of tip-to-tail flow computations for a typical Scramjet vehicle with combustion.

7.1.1 Near-Wall Viscous Resolution with Hybrid Method for Laminar Hypersonic Flow over Re-entry Capsules

Cartesian mesh based approach is used to estimate near wall quantities like heat flux for laminar hypersonic flows over axi-symmetric bodies typical of reentry capsules. To achieve this, firstly, prism layers are constructed from the background Cartesian mesh panels formed from intersection of Cartesian mesh with the body. This is done by extrusion of prism layer in stretched fashion from the background Cartesian mesh panels up to a certain user defined height. The extruded prism layer is then stitched to the outer Cartesian mesh. Subsequently, laminar Navier-Stokes code

solution is carried out for the unstructured prism layer near the wall and the Cartesian mesh away from the wall. The developed code was validated against available experimental heat flux results for a typical sphere-cone-cylinder-flare geometry and bulbous heat shield geometry at hypersonic Mach numbers under non-reacting conditions. For the finite rate chemically reacting flow, the code was validated for a wedge, sphere and hemisphere-cylinder against other CFD code solutions from structured mesh and limited experimental results. The above hybrid solution methodology is demonstrated for axi-symmetric flows.

For three-dimensional flows, since the extruded prism layers from the Cartesian mesh panels formed due to the intersection of the body are not stitched to the outer Cartesian mesh, the solution is carried out in two steps. In the first step, an Euler solution is obtained for the pure Cartesian mesh. Later near wall prism layers are constructed by extrusion from the background Cartesian mesh panels but not stitched to the outer Cartesian mesh. Subsequently, the Cartesian mesh Euler solution is mapped on to unstructured prism layer. In the next step, the laminar Navier-Stokes solution is performed for the prism layer alone using the Euler solution as the outer boundary condition for the unstructured prism layer. The solution methodology is validated against experimental heat flux values of three dimensional flow for a typical sphere-cone-cylinder-flare configuration. This methodology, does not take into account the interaction between the near wall unstructured prism layer solution to the outer Cartesian mesh solution. However this deficiency was overcome by extending the prism layer to sufficient height beyond the interaction zone.

7.1.2 Scramjet Combustion Simulation with Cartesian Mesh Solver

In the area of Scramjet combustion, we have explored the Cartesian grid solver for computing turbulent finite-rate chemically reacting Hydrogen-air combustion. This is particularly because Scramjet engines have complex geometries for which Cartesian mesh generation can be completely automated, leading to significantly

reduced turnaround time from geometry to Cartesian mesh solution. This is a very essential aspect in the design cycle of a Scramjet engine. This was achieved by developing a finite-rate chemically reacting Hydrogen-air combustion code from existing Cartesian mesh perfect gas turbulent flow with the available wall function approach. In the present computations, turbulence chemistry interaction is not taken into account. However, based on the match seen with the current computations with experimental results, it does not seem to have influenced the results. The developed code was validated against available experimental pressure and total temperature measurements for a typical Scramjet combustor test in connected pipe mode.

Since the combustor tests carried out are in connected pipe mode which is not identical to the flight condition, numerical studies were carried out to bring the effects of vitiation and inlet pressure on the combustor performance. Through the above mentioned work we could demonstrate that one could obtain good estimates of pressure and combustion efficiency with a Cartesian mesh solver for Scramjet combustion simulation and thus can be used as an effective tool to evaluate various candidate Scramjet engine configurations in the design phase.

7.1.3 Parallel Computation of Scramjet Combustion on Adaptive Cartesian Mesh with GPU Accelerators

The computation of finite rate chemically reacting flow with Hydrogen-air combustion for Scramjet vehicle is very compute intensive and clearly needs a high performance computing support to analyze a large number of candidate configurations in the design space. In this context, we have explored the use of the latest parallel computing technology of a cluster of CPU machines with GPU accelerators. In order to enable computations on the GPU cluster a tri-level parallelism approach with Pthread, MPI and CUDA was adopted. While the Pthread enables the use of multiple CPU cores of each machine sharing a common memory, the MPI enables the use of multiple machines and CUDA handles the computation in the GPU accelerator. The real challenge posed was to extract good performance from

GPU accelerators for Cartesian grid solvers with hanging nodes which exhibit poor data parallelism. This challenge was addressed by developing suitable data parallel algorithms for the adaptive Cartesian mesh and implementing good memory management techniques in the code. Data parallelism for the Cartesian mesh solver was achieved by grouping the cells into 8 different groups having similar type of computations and allocating the least data parallel cells to the CPU and highly data parallel cells to GPU for computation. The new data parallel algorithm developed by grouping the cells into different groups and launching the computations groupwise is one of the main contributions of the present work.

Tip-to-tail flow simulation of a typical Scramjet vehicle with a three strut Scramjet engine was carried out for two equivalence ratios. The performance of the Scramjet engine in terms of various quantities like total pressure, Mach number, combustion efficiency and thrust are brought out. The parallel computing performance for the above simulation on a cluster of GPU machines is also brought out. It is found that a computational load of about 1 million cells per GPU machine gives a parallel computing performance of more than 80%.

Based on the above studies, we conclude that high performance GPU cluster based Cartesian mesh solutions for Scramjet vehicle flow simulations with combustion involving very complex geometry and flow is a very good option for obtaining fast and useful solutions for a large number of candidate configurations in a typical design environment.

7.2 Future Work

In the area of near-wall viscous resolution with a combination of unstructured prism layer stitched with outer Cartesian mesh, the method is demonstrated for two dimensional and axi-symmetric flows. For three dimensional flows, the stitching of the prism layer with the outer Cartesian mesh is not done in the present work. Hence

the future work is to develop methodology to stitch the extruded polyhedral prism layers from the Cartesian mesh body panels to the outer Cartesian mesh for general three dimensional geometries. This will greatly enhance the utility of the hybrid solution methodology. Also grid adaptation based on flow gradients for the prism layer cells is another future task.

As for the Scramjet combustion, the present combustion simulations were done with laminar chemistry i.e. without turbulence-chemistry interaction. Although good match in the wall pressure was obtained for a particular Scramjet combustor test condition, this may not be true for all types of geometries and flow conditions. Hence, the inclusion of turbulence-chemistry interactions is one of the future tasks identified. Also the present computations are done with κ-ε turbulence model and the mixing of fuel with air behind the struts, being a very important process, it will be useful to investigate with other turbulence models also, and even with high fidelity models like Large Eddy Simulations. Another important task is to obtain the near wall resolution by creating very large number of cells near the wall and directly solve to wall for the pure Cartesian mesh without resorting to wall function. This could be also made possible by anisotropic Cartesian mesh cell division along with high performance computing platforms to solve extremely large number of cells that will be mostly confined to the near-wall region.

The future work envisaged on parallel computing with GPU accelerators is to enhance the performance of the adaptive Cartesian mesh solver by having better memory management by way of reduced use of global memory and better use of local memory and registers. This can reduce the memory congestion during the global memory access leading to substantial improvement in the computing performance.

REFERENCES

1. Alavilli Venkata Siva Prasad (1997). Numerical simulation of hypersonic flows and associated systems in chemical and thermal nonequilibrium. Ph.D dissertation, Dept. of Fluid Mechanics, Virje Universteit Brussel, Brussels, Belgium

2. Anderson, J.D., (1989). *Hypersonic and high temperature gas dynamics.* McGrawHill

3. Ashok,V. and Thomas C. Babu (1999). Parallelisation of Navier-Stokes code on a cluster of workstations. *Lecture notes in Computer Science*-1745, Springer Verlag edition, pp.349-353.

4. Berger, M.J. and LeVeque, R.J. (1989). An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA-89-1930-CP

5. Blottner, F.G. (1971). Chemically reacting viscous flow program for multi-component gas mixtures. *Sandia Report SC-RR-70-754*, 1971.

6. Bussing, T.R.A. and Murman, E.M. (1988). A finite volume method for the calculation of compressible chemically reacting flows. *AIAA Journal,* Vol.26,No.9

7. Candler, G.V. (1989). On computation of shock shapes in non-equilibrium hypersonic flows. AIAA-89-0312

8. Candler, G.V. (1991). Computation of weakly ionized hypersonic flows in thermochemical nonequilibrium. *Journal of Thermophysics and Heat Transfer,* Vol.5,No.3 pp-266-273

9. Chakarborty Debasis, P.J.Paul, H.S.Mukunda (2000)Evaluation of combustion models for high speed H_2/air confined mixing layer using DNS data. *Combustion and Flame*, 121:195-209.

10. Chakraborty Debasis, Roychowdhury, A.P., Ashok.V and Pradeep Kumar (2003) Numerical investigation of staged transverse sonic injection in Mach 2 stream in confined environment. *Aeronautical Journal*, 2003, 107(1078),719-729.

11. Chiang, Y.L., Van Leer, B. and Powell, K.G. (1992). Simulation of unsteady inviscid flow on an adaptively refined Cartesian grid. AIAA-92-0443

12. Corier, W.J. (1994). An adaptively refined, Cartesian, cell-based scheme for the Euler and Navier-Stokes equations. *NASA-TM 106754*, October-1994

13. Crumpton, P.I., Moiner, P. and Giles, M.B. (1997). An unstructured algorithm for high Reynolds number flows on highly stretched grids. 10th *International Conference on Numerical Methods for Laminar and Turbulent Flows,* Swansea, England, July 21-25, 1997

14. Curran,E.T. and Murthy, S.N.B. (2000). Scramjet propulsion. Vol.189, *Progress in Astronautics and Aeronautics*,Reston,Virginia:American Institute of Aeronautics and Astronautics.Inc,2000

15. Dana A. Jacobsen and Inanc Senocak (2011). A full-depth amalgamated parallel 3d geometric multigrid solver for GPU clusters. AIAA 2011-946

16. De. Zeeuw, D. and Powell, K.G. (1991). An adaptively refined Cartesian mesh solver for the Euler equations. AIAA-91-1542-CP

17. Dmitry Davidenko, Iskender Gokalp and Emmanuel Dufour (2003). Numerical simulation of hydrogen supersonic combustion and validation of computational approach. AIAA-2003-7033

18. Dunn, M.G., and Kang, S.W (1973). Theoretical and experimental studies of reentry plasmas. *NASA CR-2232*.

19. Epstein, B., Luntz, A.L. and Nachshon, A. (1992). Cartesian Euler method for arbitrary aircraft configurations. *AIAA Journal*, 30(3):679-687, March 1992

20. Evgeny V.Timofeev., Rabi B.Tahir, and Sannu Molder (2008). On recent developments related to flow starting in hypersonic air intakes. AIAA-2008-2512

21. Everett H. Phillips, Roger L. Davis and John D. Owens (2010). Unsteady turbulent simulations on a cluster of graphics processors. AIAA 2010-5036

22. Fry Ronald, S. (2004). A century of ramjet propulsion technology evolution. *Journal of Propulsion and Power*, Volume 20,No.1, January-February 2004:27-58

23. Frymier, P.D., Jr., Hassan, H.A. and Salas, M.D. (1988). Navier-Stokes calculations using Cartesian grids. *AIAA Journal*,26(10):1181-1188, October 1988

24. Gaffney, R.L., Hassan, H.A. and Salas, M.D. (1987). Euler calculations for wings using Cartesian grids. AIAA-87-0536

25. Gaitonde, V., Malo-Molina, F.J., Risha,D. and Ebrahimi,H. (2009). Integrated analysis of scramjet flow path with innovative inlets. *DoD High Performance Computing Modernization Program Users Group Conference* (HPCMP-UGC), San-Diego,CA,2009, pp-81-87

26. Georgi Kalitzin and Gianluca Iaccarino (2003). Towards immersed boundary simulation of high Reynolds number flows. *Centre for Turbulence Research, Annual Briefs*.

27. Gerlinger, P., Brugemann,D. and Algermissen,J. (1994). Numerical simulation of supersonic mixing and combustion. *Proceedings of 25th Symposium (International) on Combustion*, Irvine (CA/USA).

28. Gerlinger, P., Kasal, P., Boltz,J. and Brugemann,D. (1998). Numerical investigation of hydrogen strut injections into supersonic air flows. AIAA-98-3424.

29. Gerlinger, P., Mobus, H. and Brugemann,D. (2001). An implicit multigrid method for turbulent combustion. *Journal of Computational Physics,* Volume 167, Issue 2, pp 247-276.

30. Gerlinger, P., Kasal, P., Schneider, F., von Wolfersdorf, J., Weigand, B., and Aigner, M. (2005). Experimental and numerical investgation of lobed strut injectors for supersonic combustion. *Basic Research and Technologies for Two-Stage-to-Orbit Vehicles, Collaborative Research Centres. D.Jacob, G.Sachs, S.Wagner (Editors), 365-382, Wiley 2005*

31. Gerlinger, P., Nold, K. and Aigner, M. (2008). Investigation of hydrogen-air reaction mechanisms for supersonic combustion. AIAA-2008-4682.

32. Gerlinger, P., Nold, K. and Aigner, M. (2010). Influence of reaction mechanism, grid spacing, and inflow conditions on numerical simulations of lifted supersonic flames. *International Journal for Numerical Methods in Fluids,* 62, 1357-1380.

33. Gerlinger, P. (2012). Multi-dimensional limiting for high-order schemes including turbulence and combustion. Journal of Computational Physics, 231 (5), pp 2199-2228. .

.

34. Ghislain Tchuen., Yves Burtschelb, and David E. Zeitounb (2008). Computation of non-equilibrium hypersonic flow with artificially upstream

flux vector splitting (AUFS) schemes. *International Journal of Computational Fluid Dynamics*. Vol.22,No.4 April-May 2008,209-220

35. Gnanasekar, S., Ashok, V., Dipankar Das, and Lazar T. Chitilappily (2009). Effect of connected pipe test conditions on scramjet combustor performance. *International Journal of Aerospace Innovations*, Multi-Science Publishing, Volume 1, Number 4, December 2009, pp.159-173.

36. Gnoffo,P.A. (1989). A code calibration program in support of the aeroassist flight experiment. AIAA –89-1673, *AIAA-24th Thermophysics Conference*

37. Gnoffo,P.A. (1999). Planetary-entry gas dynamics. *Annual review of Fluid Mechanics* 31:451-494

38. Goyne,C.P., Krauss,R.H.,McDaniel J.C and Whitehurst, W.B (2007). Test gas vitiation effects in a Dual-mode Scramjet Combustor. *Journal of Propulsion and Power,* Vol.23, No.3,

39. Gupta, R.N., Yos, J.M., Thompson, R.A. and Lee, K.P. (1990). A review of reaction rates and thermodynamic and transport properties for 11 species air model for chemical and thermal nonequilibrium up to 30000K. *NASA Reference Publication, 1232*

40. Hagemann, C., Schley, C.A., Odintsov, E. and Sobatchkine, A. (1996). Nozzle flowfield analysis with particular regard to 3d-plug-cluster configurations. AIAA 96-2954. *32nd AIAA Joint Propulsion Conference*, July 1-3, Lake Buena Vista, Florida

41. Hai P.Le and Jean-Luc Cambier (2012). Development of a flow solver with complex kinetics on the graphic processing units. AIAA 2012-0721

42. Hirch, CH., Lacor, C., Rizzi, A., Eliasson, P., Lindblad, I and Haeuser, J (1991). A multiblock/multigrid code for the efficient solution of complex 3D Navier-Stokes flows. *First European Symposium on Aerothermodynamics for Space Vehicles*. ESA (ESTEC), Noordjwijk. Netherlands.

43. Huang Wei, Wang ZenGhuo and Liu Jun (2011). Parametric effects on the combustion flow field of a typical strut based scramjet combustor. Chinese *Science Bulletin,* December 2011, Vol.56, No.35, pp-3871-3877

44. Jae-doo Lee. (2006). "Development of an efficient viscous approach in a cartesian grid framework and application to rotor-fuselage interaction." Ph.D dissertation, Georgia Institute of Technology, August-2006.

45. Jeong –Yeol choi, Fuhua Ma and Vigor Yang (2005). Dynamic combustion characteristics in scramjet combustors with transverse fuel injection. AIAA-2005-4428

46. Jin Wook Lee (2007). "Parallelised Cartesian grid methodology for non-equilibrium hypersonic flow analysis of ballutes". Ph.D dissertation, Georgia Institute of Technology, August-2007

47. Jin Wook Lee, Orsini, A. and Ruffin, S.M. (2010). Unstructured Cartesian-grid methodology for non-equilibrium hypersonic flows. *Journal of Thermophysics and Heat Transfer,* Vol.24,No1, Jan-Mar-2010

48. Julien.C.Thibault and Inanc Senocak (2009). CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. AIAA 2009-758

49. Karman, S.L. Jr. (1995). SPLITFLOW: A 3D unstructured Cartesian/prismatic grid CFD code for complex geometries. AIAA-95-0343

50. Katz, A. and Jameson, A. (2009). A multi-solver scheme for viscous flows using adaptive Cartesian grids and meshless grid communication. AIAA-2009-768

51. Kee, R.J., Dixon-Lewis, G., Warnatz, J., Coltrin, M.E. and Miller, J.A.(1986). A Fortran computer code package for the evaluation of gas-phase multicomponent transport properties, *Tech. Rep. SAND 86-8246, UC-401*, Sandia National Laboratories..

52. Kee, R.J., Rupley, F.M. and Miller, J.A. (1992). The CHEMKIN thermodynamic data base, *Tech. Rep. SAND-8215B, UC-4,* Sandia National Laboratories, 1992.

53. Kim, K.H. and Kim, C. (2005). Accurate, efficient, and monotonic numerical methods for multi-dimensional compressible flows, part ii: Multi-dimensional limiting process, *Journal of Computational Physics*, 208, pp 570-615.

54. Kindler, M., Lempke, M., Gerlinger, P. and Aigner, M. (2011). TASCOM3D: A scientific code for compressible reactive flows. *14th Teraflop Workshop at the High Performance Computing Center*, Stuttgart.

55. Kodera, M., Sunami, T., and Itoh, K. (2003). Numerical simulation of scramjet engine for JAXA's flight experiment using Hyshot. AIAA-2003

56. Kuchi-Ishi,S., Watanabe,S., Nakakita, K. and Koyama,T. (2005). Comparative force/heat-flux measurements between JAXA hypersonic test facilities using standard model HB-2 (Part-1: 1.27m Hypersonic Wind Tunnel Results). *JAXA-RR-04-035E*- March-2005

57. Launder, B.E. and Sharma, B.I. (1974). Application of the energy dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer,* Vol.1, No.2, pp. 131-138.

58. Lehr, H.F. (1972). Experiments on shock-induced combustion. *Astronautica Acta,* Vol.17, pp. 589-597

59. Liou M.S. and Stefen,C.J.Jr. (1993). A new flux splitting scheme.. *Journal of Computational Physics*, 107, pp 23-39

60. Lobb, R.K. (1964). Experimental measurements of shock detachment distance on spheres fired in air at hypervelocities. *Proceedings of AGARD NATO Specialists Meeting, Fluid Dynamics Panel*

61. Luo Feiteng, Song Wenyan, Zhang Zhiqiang, Li Weiqiang and Li Jianping (2012). Experimental and numerical studies of vitiated air effects on hydrogen-fueled supersonic combustor performance. *Chinese Journal of Aeronautics* (2012) 164-172

62. Manokaran, K., Vidya, G., and Goyal, V. K. (2003) CFD simulation of flow field over a large protuberance on a flat plate at high supersonic mach number. *41st Aerospace Sciences Meeting and Exhibit*, Reno, Nevada, USA, 6th –9th January 2003, AIAA 2003 – 1253

63. Martin Krause., Reintarz, B., Ballmann, J. (2006). Numerical computation for designing a scramjet intake. ICAS-2006. *25th International Congress of the Aeronautical Sciences*

64. Martin Krause and Josef Ballmann (2007). Numerical simulations and design of a scramjet intake using two different RANS solvers. AIAA-2007-5423

65. M.D.de Tullio et.al. (2007). An immersed boundary method for compressible flow using local grid refinement. *Journal of Computational Physics*,225,pp-2098-2117

66. Melton, J.E., Berger, M.J., Aftosmis M.J.,and Wong W.D. (1995). 3D applications of a Cartesian grid euler method. AIAA-95-0853

67. Michael Emory, Vincent Terrapon, Rene Pecnick and Gianluca Iacarrino (2011). Characterising the operability limits of the hyshot-II scramjet through RANS simulations. AIAA-2011-2282

68. Moss, J.N. (1974) Reacting Viscous-Shock layer solutions with multicomponent diffusion and mass injection. *NASA TR-R-411*, June.

69. Munikrishna, N. and Balakrishnan, N. (2011) Turbulent flow computations on a hybrid Cartesian point distribution using meshless solver LSFD-U. *Computers and Fluids*, 48, 128-138.

70. Neal, E. Hass, Michael K.Smart and Allan Paull (2005) Flight data analysis of Hyshot 2. *13th AIAA/CIRA. International Space Planes and Hypersonic Systems and Technologies Conference.*

71. NVIDIA (2011). NVIDIA CUDA C Programming Guide 4.1. November.

72. Palmer, G. (1989). The development of an explicit thermochemical nonequilibrium algorithm and its application to compute three dimensional AFE flowfields. AIAA-89-1701

73. Park, C. (1987). Assessment of Two-Temperature kinetic model for ionizing air. AIAA-87-1574.

74. Partha.Mondal, Munikrishna,N. and Balakrishnan, N.(2007). Cartesian like grids using a novel grid stitching algorithm for viscous flow computations *Journal of Aircraft*,44 (5):1598-1609

75. Pellet,G.L., Claudio Bruno and Chinitz,W (2002). Review of air vitiation effects on Scramjet ignition and flameholding combustion processes. *38th Joint Propulsion Conference and Exhibit,* July, AIAA-2002-3880.

76. Peskin, C.S. (1977). Numerical analysis of blood flow in the heart. *Journal of Computational Physics*, 25:220-252

77. Rahul Ingle and Debasis Chakraborty: Numerical Simulation of Dual Mode Scramjet Combustor with significant upstream interaction, International Journal of Manufacturing, Materials, and Mechanical Engineering, Vol 2, No. 3, July-September, 2012, pp 60-74.

78. Rainer M Kirchhartz, David J Mee, Raymond J. Stalker, Perter A. Jacobs and Michael A. Smart (2010). Supersonic boundary-layer combustion: effects of upstream entropy and shear-layer thickness. *Journal of Propulsion and Power*, Vol.26, No.1, January-Febraury,2010,pp.57-66

79. Rajat Mittal, and Gianluca Iaccarino (2005). Immersed boundary methods. *Annual Review of Fluid Mechanics*,37:239

80. Rey DeLeon and Inanc Senocak (2012). GPU-accelerated large-eddy simulation of turbulent channel flows. AIAA 2012-0722

81. Scherrer, D., Dessornes,O., Montmayeur,N. and Ferrandon, O. (1995). Injection studies in the French hypersonic technology program. 6th *International Aerospace Planes and Hypersonic Technologies Conference*, April 1995, AIAA 95-6096

82. Sebastian Karl, Kalus Hannemann, Johann Steelant and Andreas Mack (2006). CFD analysis of Hyshot supersonic combustion flight experiment configuration. AIAA-2006-8041

83. Shuang-Zhang Tu. and Stephen M. Ruffin, (2002). Calculation of nonequilibrium flows using a solution adaptive, Unstructured Cartesian-Grid Methodology. AIAA-2002-3098

84. Singh, A. K., Dipankar Das, Ashok, V., Jadav, V., and Babu, C. (2009). Computational studies on the effect of blockage on starting and un-starting of hypersonic air intake. *Symposium on Applied Aerodynamics-2009*, Bangalore.

85. Soumyajit Saha and Deabsis Chakraborty, (2011). Reacting flow computation of staged supersonic combustor with strut injection. *Journal of Aerospace Sciences and Technologies*, Vol 63, No.4 , Nov, 2011, pp289-298.

86. Sudhakaran, G., Thomas.C.Babu and Ashok, V. (2012) A GPU computing platform (SAGA) and a CFD code on GPU for aerospace applications. *ATIP/ACRC Workshop on Accelerator Technologies for High Performance Computing*, May, Singapore, pp 117-121.

87. Srinivasa, P (1991). Experimental investigation of hypersonic flow over a bulbous heat shield at Mach number 6. Ph.D thesis, IISc, Bangalore.

88. Svehla, R. A. (1962). Estimated viscosities and thermal conductivities of gases at high temperatures. *NASA TR-R-132,* 1962.

89. Tani, K., Kanda,T. and Kudou, K. (2001). Effect of side spillage from airframe on scramjet engines. *Journal of Propulsion and Power,* Vol.17, No 1

90. Tani, K., Takeshi Kanda and Kenji Kudou. (2006). Aerodynamics performance of scramjet inlet models with a single strut. *Journal of Propulsion and Power*, Vol.22,No.4,July-August 2006

91. Townend, L.H.(2001). Domain of Scramjet. *Journal of Propulsion and Power*, Volume-17,No.6,Nov-Dec-2001,1205-1213

92. Venkatakrishnan V. (1995). Convergence to steady-state solutions of the euler equations on unstructured grids with limiters. *Journal of Computational Physics,* 118, pp.120-130

93. Vijayan, P. and Kallinderis, Y. (1994). A 3D finite volume scheme for the Euler equations on adaptive tetrahedral grids. *Journal of Computational Physics,* 113, pp. 249-267

94. Voland, R.T., Huebner,L.D. and McClinton,C.R.(2006). X-43A hypersonic vehicle technology development, *Acta Astronautica*, 59:181-191

95. Wang, Z.J.(1996). A fast nested multi-grid viscous flow solver for adaptive cartesian/quad grids. AIAA-96-2091

96. Wang, Z.J. (1998). A quadtree-based adaptive Cartesian/quad grid flow solver for Navier-Stokes equations. *Computers and Fluids*,27(4):529-549

97. Xiangying Chen, and Ge-Cheng Zha.(2009). A hybrid Cartesian-body fitted grid approach for simulation of flows in complex geometries. AIAA-2009-3880

98. Ya'eer Kidron, Yair Mor-Yossef, and Yuval Levy (2010). Robust cartesian grid solver for high Reynolds number turbulent simulations. *AIAA Journal*, Vol.48, No.6, June.

99. Yang, G., Causon, D.M., Ingram, D.M., Saunders, R. and Batten, P.(1997a). A cartesian cut cell method for compressible flows. Part A: Static body problems. *Aeronautical Journal*, 101(2):47-56,February 1997

100. Yang, G., Causon, D.M., Ingram, D.M., Saunders, R. and Batten, P.(1997b). A cartesian cut cell method for compressible flows. Part B: Moving body problems. *Aeronautical Journal*, 101(2):57-65, February 1997

APPENDIX-1

The cell data structure used for the GPU programming and important parts of the GPU program with brief explanation is given

```
typedef union cell {
  CLEAF item;
  CNODE attr;
} CELL;

typedef struct {
   char level;   char ncel;   unsigned char load;   unsigned char celllev;   unsigned
short i;   unsigned short j;   unsigned short k;   unsigned int lev; unsigned int marker;
real *U;   real *Us
typedef struct {
   char st;   char ncel;     unsigned char freeze;   unsigned char load;   unsigned char
celllev;
   unsigned short i;   unsigned short j;   unsigned short k;   unsigned int lev;
unsigned int marker; ;   real* Ub;   struct PTCL  *pcp;  union cell *Nb[6];
} CLEAF;

   typedef struct{
   real *U;   real *Us;   real *Ub;   struct PTCL  *pcp;   union cell *next;  union
cell *Nb[6];
} CNODE;

typedef struct PTCL {
  real Pt[P_DIM];
}PCELL;                    /* partial cell parameters structure */
```

The “CELL” is a union of two structures “CLEAF “and “CNODE “.A node cell is a cell which has further children. A leaf cell is the cell which has no further divisions and has the following information.

1) One character (1 byte) to represent as to how many levels of division it has undergone char level

2) One character to represent whether the leaf cell is gas cell, partial cell or body cell char ncel

3) The I,J,K of the parent cell is stored as unsigned short (2 bytes) unsigned short i, unsigned short j, unsigned short k

4) The position of the leaf cell in the parent cell is obtained by 32 bits of information (7 levels of division and each direction represented by one bit) and is represented by unsigned integer (4 bytes)unsigned int lev

5) Conserved variable is stored in double precision which is 8 bytes (defined as real). Pointer to the conserved variable vector, both the previous time step as well as the updated one is .stored in cell data structure and later suitable memory allocation is done depending upon the type of problem (for chemically reacting flow with combustion vector of 15 conserved variables is used) real *Us; real *U;

6) Pointer to the boundary cell conserved variable vector real *Ub

7) Computation load for the cell. For gas cell the load is unity and for partial cell the load is 1.4. For each neighbour being split, the load is augmented by unity and for neighbour's neighbour being split it is augmented by 10 unsigned char load

8) Pointer to the partial cell structure (used if the cell is partial) struct PTCL *pcp

9) The partial cell has 10 values (Pt[P_DIM]) each of which are represented in double precision. These are 6 partial fluxing areas of faces, 3 direction cosines of the normal to the wall and distance from the cell center to the wall real Pt[P_DIM]

10) Pointer to the 6 neighbouring cells, one cell adjacent to each face is represented union cell *Nb[6];

11) If the cell is divided then it has the information about 8 of its children represented by union cell *next

Each CPU and GPU thread has the entire information about the cells in the Machine and is represented by the following Celllinks data structure

```
typedef struct Celllinks_ {
  int count,gcount,pcount;  int Maxgcount,Maxpcount;
  int ggrp0,Maxggrp0, ggrp1,Maxggrp1, int ggrp2,Maxggrp2;
  int ggrp3,Maxggrp3, ggrp4,Maxggrp4,  pgrp0,Maxpgrp0;
  int pgrp1,Maxpgrp1, pgrp2,Maxpgrp2, pgrp3,Maxpgrp3;
  int Tstep,StabControl;  int Nx,Ny,Nz;
```

```
int TRANSIENT,NAVIE,ROE,AUSM;   char RFILE[128];
int Bound[6], Maxus,Nus; unsigned long Cell_offset,U_offset,Pcp_offset;
  real *Us, *GpuUs;
real Init_sum,Stab, a[320], *Hx,*Hy,*Hz;
real FsM,FsT,FsP,JetM,JetT,JetP;
     CELL *C_arr,*d_C_arr, **cinfo,  **pcell, **gcinfo;
CELL **gpcel,**g0cell, **g1cell,**g2cell, **g3cell, **p0cell, **p1cell, **p2cell;
      CELL **p3cell;   CELL **gg0cell;
CELL **gg1cel, **gg2cell, **gg3cell, **gp0cell, **gp1cell;
CELL **gp2cell, **gp3cell, **CApart **Bcell;
struct Celllinks_ *GPUCelllink;
int Gpu;   real *Uptr;   real *GPUUptr;   PCELL *GPUPcpptr;
CELL *GPUCellptr;   PSYNC mlock;   PSYNC glock;   real Uindata[U_DIM];
   int Gpuubcount;   int Gpuubcountmax;
int transloc;   real * Ub;   real * GpuUb;

} CELLLINKS;
```

The CELLLINK structure has all the information needed for computation of which the important ones are given below

1) Number of gas cells in the particular CPU or GPU Celllinks- int gcount
2) Number of partial cells in the Celllinks int pcount
3) Total number of partial cells in the sub-domain int Maxpcount
4) Total number of gas cells in the sub-domain int Maxgcount
5) Total number of cells in X,Y & Z direction in the sub-domain int Nx,Ny,Nz
6) Number of partial cells and gas cells of group0 to group3 in the thread denoted through Celllink data structure int pgrp0, pgrp1, pgrp2, pgrp3, ggrp0, ggrp1, ggrp2, ggrp3
7) Total number of partial cells and gas cells of group0 to group3 in the sub-domain int Maxpgrp0,Maxpgrp1,Maxpgrp2,Maxpgrp3,Maxggrp0, Maxggrp1, Maxggrp2, Maxggrp3
8) All the flow variable inputs and the flow properties and the domain size represented by double precision real [320]
9) Pointers to the parent cell vertices real *Hx,*Hy,*Hz
10) Pointer to conserved variable vector of cells in present CPU thread real *Us

11) Pointer to conserved variable vector of cells in GPU thread real *GPuUs

12) Pointer to array of partial and air cell groups CELL**gpcel,**g0cell, **g1cell,**g2cell, **g3cell, **p0cell, **p1cell, **p2cell,**p3cell

13) Pointer to array of partial and air cell groups in GPU CELL**g0cell, **g1cell,**g2cell, **g3cell, **p0cell, **p1cell, **p2cell,**p3cell

14) Pointer to GPU celllinks struct Celllinks_ *GPUCelllink

15) Initial guess values of the conserved variable vector real Uindata[U_DIM]

16) Pointer to boundary cell conserved variable vectors real * Ub

17) Pointer to boundary cell conserved variable vectors in GPU real * Ub

The program below gives the SetupLinks function to generate the Celllink data structure which would have the information as given above. The full list of the function with all statements, variable declaration, and include files are avoided to make the presentation short and concise. The number of celllink data structure is the equal to sum of total CPU cores and GPU accelerators. The celllink contains all the essential information needed for calculation of each cell.

```
int SetupLinks(int cores){
        Nthreads = Ncore + Ngpu;
        Celllinks = (CELLLINKS *)malloc(sizeof(CELLLINKS)*Nthreads);
      Pth = (pthread_t *)malloc(sizeof(pthread_t)*(Nthreads+Ngpu));
        for (i=0;i<Nthreads;i++) {
              Celllinks[i].Hx = Hx;
              Celllinks[i].Hy = Hy;
              Celllinks[i].Hz = Hz;
              memcpy( Celllinks[i].a,a,320*sizeof(real));
              memcpy( Celllinks[i].Bound,Bound,6*sizeof(int));
              Celllinks[i].Nx = Nx;
              Celllinks[i].Ny = Ny;
              Celllinks[i].Nz = Nz;
              Celllinks[i].d_C_arr = GPUCellptr;
               Celllinks[i].C_arr = Cellptr;
               Celllinks[i].Cell_offset = Cell_offset;
              Initdata(Celllinks[i].Uindata);
              count = Celllinks[i].count;
              Celllinks[i].pcount = pcellcount(Celllinks[i].clist);
               Celllinks[i].pgrp0 = pgrp0count(Celllinks[i].clist);
              Celllinks[i].pgrp1 = pgrp1count(Celllinks[i].clist);
```

```
            Celllinks[i].pgrp2 = pgrp2count(Celllinks[i].clist);
            Celllinks[i].pgrp3 = pgrp3count(Celllinks[i].clist);
            Celllinks[i].ggrp0 = ggrp0count(Celllinks[i].clist);
            Celllinks[i].ggrp1 = ggrp1count(Celllinks[i].clist);
            Celllinks[i].ggrp2 = ggrp2count(Celllinks[i].clist);
            Celllinks[i].ggrp3 = ggrp3count(Celllinks[i].clist);
            Celllinks[i.gcount = count - Celllinks[i].pcount;
        }
}
```

The program statements below denote the launching of CPU threads.

```
for(i=0;i < (Nthreads); i++) {
      pthread_create(Pth+i,NULL,CpuThread,(void*)(Celllinks+i));
}
```

For dual quad core machine with 2 GPU accelerators, Nthreads is 10.(2*4+2). The extra CPU threads which are equal to number of GPU accelerators are meant to control the GPU threads. The pthread_create function for CPU threads executes the CpuThread function with argument Celllinks+i. The CpuThread function does the computation of all the cells that are represented in Celllinks[i] of i^{th} CPU core.

The program, statements to create GPU threads is given below

```
for(i=Nthreads;i < (Nthreads+Ngpu); i++) {
pthread_create(Pth+i,NULL,GpuThread,(void*)(Celllinks+i-Ngpu));
 }
```

. The function GpuThread with corresponding GPU celllink pointer containing all the information about the groups of the cells that need to be computed is passed as the argument. In the GPU thread function all the information in celllink data structure of the GPU thread celllink residing in CPU is copied to the GPU through the copygpumemory function which uses the cudaMemcpy utility function. After all the information of celllinks is copied to GPU, the cudamain function is called which launches the GPU kernels involving computation in GPU cores in groups of cells. Computation in GPU is done on one dimensional grid with a certain number of thread blocks. Number of thread blocks is the number of cells of particular group divided by the number of threads in a block. The number of threads in a block denoted by Ngridg in the program statements below is a user defined value. For the present

code, 128 threads in a thread block gave very good performance for a wide range of problems as compared to other values.

```
Ng = clink->ggrp0;
  MaxThds=128;
  if(Ng > 0) {
  Ngridg = Ng/MaxThds;
  dim3 grids (Ngridg,1);
```

Once the grid with number of thread blocks and number of threads per block is identified (dim3 grids (Ngridg,1);) as given by the program statements above, the next step is to convert the shared memory of the GPU Streaming Multi-Processor (SM) to L1 cache for faster memory access. This is done through CUDA utility statement as given below.

```
cutilSafeCall(cudaFuncSetCacheConfig(Gpucalcg0,cudaFuncCachePreferL1));
```

After this, the important task of launching GPU Kernels for computation in GPU cores through CUDA function call is done for different groups of cells one after another. The following program statement below gives the GPU Kernel launch for group 0 type cells.

```
Gpucalcg0<<<grids,gthrds>>>
(clink->GPUCelllink,MaxThds,Ng,SEGSIZE,stabloc);
```

The above statement gives the command to launch the GPU Kernel to calculate the group 0 type cells of Ng in number with grid containing the certain number of thread blocks and each thread block containing MaxThds number of threads. The pointer to the Cellinks structure in GPU which contains the entire information needed for flow computation, clink->GPUCelllink is passed as the argument and stabloc is the convergence factor which is to be obtained from the calculation. Once the GPU Kernel is launched, then the calculation of group 0 cells of the function d_calc_grp0gcell(n*SEGSIZE,count,cinfo,clink->GpuUs+Stabloc)is done in GPU. In GPU this function gets executed in clusters of 32 cells at a time.

The computer program with important functions and only important statements are listed below for brevity.

```
void *GpuThread(void *dummy) {
        int i=0,size,size1,size2,Gpu;
  CELLLINKS *clink;
  clink = (CELLLINKS *)dummy;
  real totcpu,prevcpu,cpu,stcpu,upt,maint,copyt;
  pthread_mutex_lock(&mulock);
  psync_slave_init(&(clink->glock));
    Gpu = clink->Gpu-1;
  SetGpuDevice(Gpu);
  cudaSetDeviceFlags(cudaDeviceBlockingSync);
  setupgputhread(dummy);
  size = (sizeof(real)*((U_DIM+EXMEM)*clink->Nus));
  size1 = (sizeof(real)*((U_DIM)*clink->Nus));
  size2 = (sizeof(real)*(clink->Nus));
    if(size != 0) {
      cutilSafeCall(cudaMemcpy(clink->GPUUptr,clink->Us,size,
                               cudaMemcpyHostToDevice));
  }
  cudainit(clink);
  Gpuready++;
  pthread_mutex_unlock(&mulock);
// Wait here for trigger
  while (1) {
    wait_master(&(clink->glock));
    psync_wait_master(&(clink->glock));
       UpdateGpu(clink);
       cudamain(clink);
    cutilSafeCall(cudaThreadSynchronize());
       psync_signal_master(&(clink->glock));
    cudaCopyUgrps(clink);
    cutilSafeCall(cudaThreadSynchronize());
  }


void * setupgputhread(void *dummy) {
 allocgpuallmemory(dummy);
 allocgpumemory(dummy);
 copygpumemory(dummy);
 return NULL;
}
void * copygpumemory(void *dummy) {
 CELL *el;
 int j,i;
 CELLLINKS *clink,*GPUCelllink;
 clink = (CELLLINKS *)dummy;
```

```
   GPUCelllink = clink->GPUCelllink;
   cutilSafeCall(cudaMemcpy(clink->Hx,Hx,
        (clink->Nx+1)*sizeof(real),cudaMemcpyHostToDevice));
   cutilSafeCall(cudaMemcpy(clink->Hy,Hy,
        (clink->Ny+1)*sizeof(real),cudaMemcpyHostToDevice));
   cutilSafeCall(cudaMemcpy(clink->Hz,Hz,
        (clink->Nz+1)*sizeof(real),cudaMemcpyHostToDevice));
   for(i=0;i<clink->ggrp0;i++) {
     clink->g0cell[i]=
       (CELL *)((unsigned long)(clink->g0cell[i])+clink->Cell_offset);
   }
        cutilSafeCall(cudaMemcpy(clink->gg0cell,clink->g0cell,
sizeof(CELL *)*clink->ggrp0,cudaMemcpyHostToDevice));
}

int cudamain(CELLLINKS *clink) {
  int Ng,Np,stabloc;
  int Ngridg,Ngridp;
  int MaxThds=MAXTHDS;
  float sttime,etime;
  dim3 gthrds(MaxThds,1);
  dim3 pthrds(MaxThds,1);
  int SEGSIZE=1;
  int Blks;
  Blks = MaxThds*16*14*10;
  stabloc = clink->Nus*U_DIM;
  Ng = clink->ggrp0;
  SEGSIZE = 1;
  if(Ng > 0) {
    Ngridg = (Ng+ MaxThds*SEGSIZE-1)/(MaxThds*SEGSIZE);
    dim3 grids (Ngridg,1);
    cutilSafeCall(cudaFuncSetCacheConfig(Gpucalcg0,cudaFuncCachePreferL1));
    Gpucalcg0<<<grids,gthrds>>>
(clink->GPUCelllink,MaxThds,Ng,SEGSIZE,stabloc);
  }
stabloc +=Ng;
  Ng = clink->ggrp1;
  SEGSIZE = (Ng+Blks-1)/Blks;
  SEGSIZE = 1;
  if(Ng > 0) {
    Ngridg = (Ng+ MaxThds*SEGSIZE-1)/(MaxThds*SEGSIZE);
    dim3 grids (Ngridg,1);
    cutilSafeCall(cudaFuncSetCacheConfig(Gpucalcg1,cudaFuncCachePreferL1));
    Gpucalcg1<<<grids,gthrds>>> (clink-
>GPUCelllink,MaxThds,Ng,SEGSIZE,stabloc);
```

```
  }

}


__global__ void Gpucalcg0(CELLLINKS *clink,int MaxThds,int size,int
SEGSIZE,int Stabloc) {
   int i,j,k,l,count;
   int n;
   CELL **cinfo;
   i= threadIdx.x;
   l= blockIdx.x;
   n = i+ MaxThds*l;
   count = SEGSIZE;
   if( (n*SEGSIZE+count) > size ) count = size - (n*SEGSIZE);
   if( count > 0 ) {
     cinfo =  clink->gg0cell;
     d_calc_grp0gcell(n*SEGSIZE,count,cinfo,clink->GpuUs+Stabloc);
   }
}
__global__ void Gpucalcg1(CELLLINKS *clink,int MaxThds,int size,int
SEGSIZE,int Stabloc) {
   int i,j,k,l,count;
   int n;
   CELL **cinfo;
   i= threadIdx.x;
   l= blockIdx.x;
   n = i+ MaxThds*l;
   count = SEGSIZE;
   if( (n*SEGSIZE+count) > size ) count = size - (n*SEGSIZE);
   if( count > 0 ) {
     cinfo =  clink->gg1cell;
     d_calc_gcell(n*SEGSIZE,count,cinfo,clink->GpuUs+Stabloc);
   }
}
```

PUBLICATIONS BASED ON THE THESIS

1. Ashok, V., Adimurthy, V. and George Joseph (2012). Computation of heat flux in hypersonic flow with a Cartesian mesh using near-wall resolution. **Paper accepted for publication in *Journal of Aerospace Sciences and Technologies.***

2. Ashok,V., Adimurthy, V. and George Joseph (2013). Computation of non-equilibrium chemically reacting hypersonic flow from a Cartesian mesh with near wall viscous resolution. **Paper accepted and to be published in *Journal of Applied Fluid Mechanics,* Volume 7, Number 2,April 2014.**

3. Ashok, V., Harichand, M.V., Thomas.C.Babu, Adimurthy, V. and George Joseph (2013). Acceleration of an adaptive Cartesian grid based solver for hypersonic chemically reacting flows on a cluster of GPU based systems. Paper under review by *Journal of Parallel Computing*.

4. Ashok, V., Adimurthy, V. and George Joseph (2013). Heat flux computation in hypersonic flow with Cartesian mesh using hybrid solution methodology**. Paper accepted for publication in *International Review of Aerospace Engineering (IREASE).***