Accelerating a Hypersonic CO_2 Reaction Solver

Undergraduate Thesis

submitted in Partial fulfillment of the requirements for the degree of

Bachelors of Technology

in the

Department of Mechanical Engineering

By

G. Vinay Ram
ID No. AP17110030007
&
D. Dinesh Sai
ID No. AP17110030009

Under the supervision of:

Dr. Satya Pramod Jammy

SRM University-AP

May, 2021

Certificate

This B.Tech report entitled " Accelerating a Hypersonic CO_2 Reaction Solver" by G. Vinay Ram (AP17110030007) and D. Dinesh Sai (AP17110030009) is prepared and submitted as partial fulfillment of the requirements for the degree of Bachelors of Technology in Department of Mechanical Engineering.

Supervisor
Dr. Satya Pramod Jammy
Associate Professor
Department of Mechanical Engineering
SRM University-AP

Head of Department (HOD)
Dr. Prakash Jadhav
Associate Professor
Department of Mechanical Engineering
SRM University-AP

Examiner

Date: 24 - 05 - 2021
Place: SRM University-AP

"If human life were long enough to find the ultimate theory, everything would have been solved by previous generations. Nothing would be left to be discovered."

Stephen Hawking

Abstract

Atmospheric entry is one of the greatest challenges faced during interplanetary missions. Aerodynamic heating, equilibrium or non-equilibrium gas chemistry and strong shocks are some of the major obstacles encountered by the spacecraft at hypersonic speeds during re-entry into the atmosphere. The spacecraft should be designed efficiently to endure these repercussions. Computational fluid dynamics has emerged as a useful tool for achieving this goal. A 2-D unstructured finite volume method based solver capable of simulating hypersonic flows in CO_2 rich Martian atmosphere with chemical kinetics for eight species consisting of $N_2, O_2, NO, O, N, CO_2, CO$ and C incorporated in it is developed. It solves 2-D Navier-Stokes equations along with the species continuity equations. The solver uses AUSM delta schemes for evaluating convective flux terms and node gradients for calculating viscous flux terms. A simple first order explicit Euler scheme is utilized for temporal discretization. The solver is parallelized to run on single or multiple CPUs and GPUs in order to attain greater computational speeds using OP2 application programming interface. The kernel files and parallel loops for OP2 API are generated using semi-automatic code generation technique which uses symbolic Python package SymPy. The OP2 generates kernels and executable parallel loops for high level scripts which can be implemented using multiple computational architectures like OpenMP, MPI, OpenCL, CUDA, etc,. The developed solver is validated for canonical ramp in a channel test case with published data and assessed the speedup of the solver using different architectures. The solver has a speedup of 13x using OpenMP compared to the sequential solver for the considered inviscid test cases.

Keywords: Mars solver, FVM, GPU, auto parallel.

Acknowledgements

First and foremost, praises and thanks to our parents for their immense love and blessings throughout our project.

We are tremendously grateful for our mentor **Dr. Satya Pramod Jammy** for his constant support and encouragement throughout the project. His dynamism, sincerity, vision and motivation have deeply inspired us. We will forever be indebted to him for this opportunity.

We feel short of words to express our heartfelt thanks to all the Professors of Mechanical Department, SRM University-AP and to all those who have directly or indirectly helped us during our journey. This would have been an impossible task without the support from our families and friends.

Contents

List of Figures

List of Tables

Abbreviations

API	-	Application Programming Interface
AUSM	-	Advection Upstream Splitting Method
CFD	-	Computational fluid dynamics
CFL	-	Courant-Friedrichs-Lewy
CPU	-	Central Processing Unit
CUDA	-	Compute Unified Device Architecture
FDM	-	Finite Difference Method
FEM	-	Finite Element Method
FVM	-	Finite Volume Method
GPU	-	Graphics Processing Unit
MPI	-	Message Passing Interface
OP2	-	Oxford Parallel library for Unstructured mesh solvers
OpenCL	-	Open Computing Language
OpenMP	-	Open Multi-Processing
SWBLI	-	Shock Wave Boundary Layer Interaction

Nomenclature

C_i	-	Mass concentration of species i (kg/m^3)
C_p	-	Specific heat at constant pressure (J/kgK)
C_v	-	Specific heat at constant volume (J/kgK)
C_{pi}	-	Specific heat of species i at constant pressure ($J/kmolK$)
e	-	Internal energy (J/kg)
e_i	-	Internal energy of species i (J/mol)
E	-	Total energy (J/kg)
E_v	-	Viscous flux vector in x direction
E_1	-	Convective flux vector in x direction
F_c	-	Convective Flux vector
F_v	-	Viscous flux vector in y direction
F_1	-	Convective flux vector in y direction
h_i	-	Specific enthalpy of species i (J/kg)
h^0_{fi}	-	Heat of formation of species i (J/mol)
k	-	Conductivity ($W/(mK)$)
M	-	Mach number
MW_i	-	Molecular weight of species i
N	-	Total number of species
p	-	Pressure (N/m^2)
q_x	-	Heat flux in x direction (W/m^2)

q_y	-	Heat flux in y direction (W/m^2)
R_u	-	Universal gas constant ($J/kmolK$)
S_I	-	Inviscid axisymmetric source term
S_i	-	Species production rate (kg/m^3s)
S	-	Source term vector
S_v	-	Viscous axisymmetric source term
t	-	Time (s)
T	-	Temperature (K)
U	-	Conserved variable vector
u	-	Velocity in x direction (m/s)
v	-	Velocity in y direction (m/s)
X	-	X-coordinate (m)
Y	-	Y-coordinate (m)
Y_i	-	Mass fraction of species i
α	-	Constant ($\alpha = 1$ for 2-D axisymmetric, $\alpha = 0$ for 2-D problem)
β	-	Shock Wave angle
ρ	-	Density (kg/m^3)
μ	-	Viscosity (Ns/m^2)
θ	-	Angle ($degree$)
$\tau_{xx}, \tau_{xy}, \tau_{yy}, \tau_{\theta\theta}$	-	Shear stress components (N/m^2)

Dedicated to our Parents . . .

Chapter 1

Introduction

1.1 Motivation

Ever since man conquered the lunar surface, it has been a distant dream to set foot on Mars. There are dozens of compelling reasons for us to explore Mars, some of them include the search for Martian existence, understanding the planet's surface, and to colonize human life, etc. Certain aspects can be undoubtedly noticed through observational satellites which revolve around the planet, these observational satellites tremendously help in analysing and accurately reporting the specific information found there and not all key things can be recognized by them. For exploring things like the rock composition or collecting samples of soil and performing experiments, etc., we have to land onto the surface. This brings us to the concept of atmospheric entry.

Movement of any object from outer space entering into the atmosphere or through the gases of the atmosphere is termed as atmospheric entry. During the atmospheric entry, hypersonic vehicles are subjected to strong shocks, equilibrium or non-equilibrium gas chemistry, large heat fluxes and as a consequence, significantly high temperatures are reached on the structure [38] and vibrations are increased. To design the spacecraft we would require a seamless understanding of the technical and fundamentals of multiple fluid flow regimes like subsonic, transonic, supersonic, and hypersonic.

The fig. 1.1 is an example for formation of bow shock at hypersonic speed. There is a wake region formed behind the re-entry vehicle which separates the flow and its interaction with

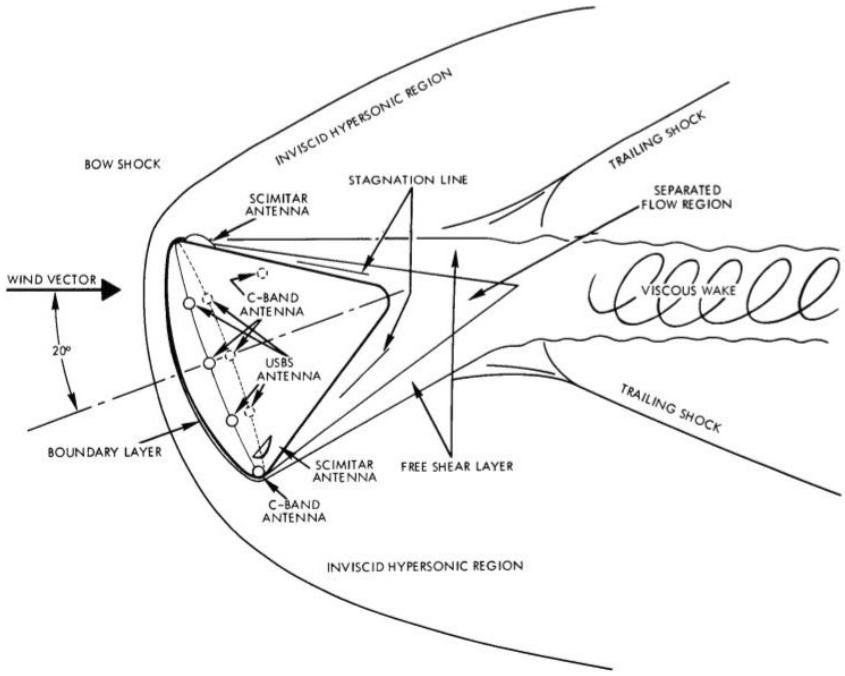


FIGURE 1.1: Re-entry flow regions of the Apollo Command Module [12]

free shear layer forms a trailing shock. As the re-entry vehicle enters into the atmosphere at hypersonic speeds, we can observe elevated temperatures that are more than 800K [11]. As a consequence, the shock layer results in the occurrence of various physical phenomena like vibrational relaxation, chemical reactions, ionization, radiation, etc. Aerodynamic heating is one of the most significant problems encountered in hypersonic flights. Albeit a variety of factors such as shock wave boundary layer interaction, excess drag force, heat effects, etc. come into view.

Mars's atmosphere is about 1% dense of Earth's atmosphere. Martian atmosphere composition is: 95% by volume of carbon dioxide (CO_2), 2.6% molecular nitrogen (N_2), 1.9% argon (Ar), 0.16% molecular oxygen (O_2), and 0.06% carbon monoxide (CO) at the surface. It is primarily composed of carbon dioxide (CO_2) [40]; and due to the aerodynamic heating the molecules of gas dissociate near the surface, as the result of this phenomena the gas around the re-entry vehicle behave abnormally. So, while designing the spacecraft we should consider all of these characteristics.

In order to simulate the flight conditions of re-entry vehicles we have to develop a solver with the required chemical kinetics for Mars's atmosphere. Developing a solver that can run sequentially on the Central Processing Unit (CPU) takes enormous amount of

computational time. For example, to converge a 3D Navier-Stokes flow case using CPU with more than millions of degrees of freedom takes thousands of computational hours. To overcome this parallelization is the ideal approach. Solving the same example flow case using high performance parallel computing [2] reduces the time.

In this work we developed a Parallelized Hypersonic CO_2 based reaction solver using OP2 [16] framework which can be run using single or multiple CPUs and GPUs. This is used in solving the numerical problems for hypersonic re-entry vehicles in the Martian atmosphere.

1.2 Thesis Overview

Chapter 2 discusses the literature review on the high temperatures effects, Shock wave boundary layer interaction and also addresses the objectives of the work. Chapter 3 demonstrates the governing equations and numerical formulation of in-house developed non-equilibrium flow solver. Chapter 4 focuses on parallelization strategy and background of semi-automatic code generation. Chapter 5 deals with validation of the solver followed by the speedup comparison for both inviscid and viscous test cases. Finally, the thesis ends with conclusions and future work in chapter 6.

Chapter 2

Background and Literature review

The relevance of physical effects and its characteristics for high-speed flows was highlighted in the previous chapter. In view of this, the current section addresses high temperature effects, shock wave boundary layer interaction (SWBLI), a literature survey on parallel computation using OP2 framework, and finally, problem statement and objectives.

2.1 High temperature effects

High temperature effects have a significant impact on the flow field modifications and aerodynamics of high-speed vehicles. High temperatures of over 800K can be observed in hypersonic flows such as space vehicles. Due to flow deceleration in the shock layer, the kinetic energy is converted into heat energy (internal energy) resulting in huge temperature difference post-shock. This causes vibrational excitation in gas molecules which inturn causes collisions of atoms. This also leads to ionization and dissociation reactions in which molecules break down into new atoms. It signifies that fluid need time to complete the reactions and achieve a final chemical composition in the equilibrium state. Over which, the thermophysical properties like specific heat, viscosity, and thermal conductance get disrupted. Moreover, the presence of free electrons due to ionized air can result in a blackout, where spacecraft communications become difficult. These phenomena are referred as "real gas effects". The temperature ranges of vibrational excitation, dissociation, and ionization reactions for air at 1 atm pressure are shown in fig. 2.1. O_2 begins to dissociate at 2500K and it completely gets dissociated at 4000K. Range of dissociation is

from 2500K to 9000K and the range of ionization begins from 9000K.

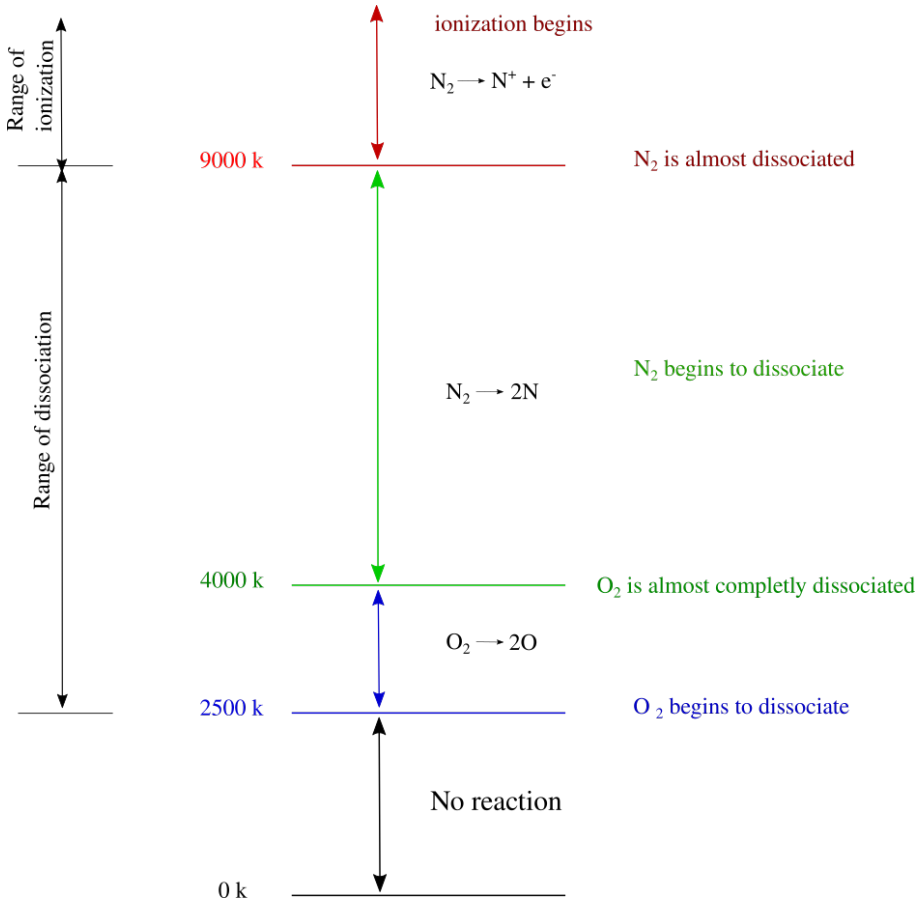


FIGURE 2.1: Ranges of various high temperature aspects of air (1 atm)

It is important to understand high temperature effects. Lobb [28] did the most groundbreaking research on the impact of high temperatures and also conducted several experiments to find out the shock stand-off distance on spheres driven with hypervelocity. Hornung [22] demonstrated the dissociation of nitrogen over a blunt body in an experiment. Followed by this, a number of numerical studies to obtain accurate aerodynamic analysis were published [42]. Then after various flux computation schemes like Van Leer and Roe were derived for simulation purpose. Using the Direct Simulation Monte Carlo method (DSMC) with a chemistry model of five species, a simulation of transitional flow for hypersonic reentry conditions around the nose of a space shuttle is presented by Moss and Bird [30]. For low altitude free stream conditions the results are very good with experimental observations. In addition, at higher altitudes, no changes were seen in the composition of free stream gas behind the shock wave.

Many researchers did a vigorous amount of work to simulate and understand the chemical equilibrium flow. An algorithm was developed to obtain the chemical equilibrium of air, based on a equilibrium constant method by Pimentel and Alberto Rocha [35] to achieve the equilibrium composition for a wide range of temperatures and pressures, they used both five and seven species model. It was found to be simple to integrate with any solver with the proposed algorithm.

In previous studies [33, 32, 13] various chemical models were presented. Tchuen and Zeitoun [43] investigated the impact of chemical kinetic models in multiple areas, along with a wide range of mach numbers. The main focus of these studies is backward reaction rates. Later, for analysing surface heat flux on hypersonic vehicles, Wang [48] carried out similar numerical evaluations and presented the comparison of numerical results to experimental data by taking three test cases, ELECTRE vehicle, Space Shuttle orbiter and Apollo command module for mach numbers between 13 and 20.5, and results were consistent.

In recent times, there has been a growing curiosity in studying the aerodynamics of Martian atmosphere. Candler [8] presented a chemical kinetics model for CO_2 - N_2 mixture which is for the martian atmosphere integrated in a 2-D Computational Fluid Dynamics (CFD) model and also highlighted that thermal radiation is important because it can be used for flow cooling. Mitcheltree and Gnoffo [29] gave a computational approach used to describe the aerothermodynamics of hyper-velocity vehicles in mars atmosphere by considering maximum heating and maximum deceleration points. They developed ablative and non-ablative boundary conditions to audit its impact on surface heating. Later, Sharma and Swantek [39] published numerical and experimental works of ongoing efforts to study high-enthalpy carbon dioxide flows. The experiments are carried out on a hyper-velocity expansion tube for aero-shell to do various calculations. Predictions have been made for heat transfer and shock stand-off distance calculations with conceptual and numerical estimations for three different ramp angles. Hao et al. [20] numerically investigated effects of two different models for transport properties on Mars atmospheric entry vehicles. Both models were observed to have identical translational-rotational and vibrational heat fluxes. To estimate the chemical diffusion flux, the collision integral model should be used. Furthermore, the transport models had no impact on the wake structure. This would provide great insight into high temperature effects and how it plays

a prominent role in designing a robust and reliable flight.

2.2 Shock wave boundary layer interaction (SWBLI)

Shock wave boundary layer interaction (SWBLI) has a prominent role in flight performance. Shock waves are commonly encountered in supersonic and hypersonic flows near wing-body junction, turbines, nozzle, missiles, helicopter blades, reentry vehicles, etc., which may impact its efficiency. The majority of flow properties like pressure, temperature, heat flux, transition to turbulence, etc., are interrupted by shock waves which are very dangerous to overall safety.

Numerical and experimental investigations have been carried out to study the behaviour of SWBLI [5] and to take immediate steps to reduce its adverse effects. The most pioneering work to study SWBLI was carried by Ackeret et al. [1] in the late 1940s. The study's aim was to investigate the effects of shock and boundary layer in a wind tunnel. They observed unusual shock patterns, which they theorized were caused by frictional impacts. For the first time in the history of laminar flow events, these complex shock geometries are captured. Subsequently, a series of experiments were performed by Gadd et al. [15] to investigate the impact of wide range parameters like free stream Reynolds number, Mach number and shock strength on shock impingement and ramp induced SWBLI. Three different forms of boundary layers were investigated in this study.

Further, the hypersonic viscous flows for test cases like wing-fuselage and wing-flap junction have been analysed by F. Grasso, M. Marini [17]. Different effects on the flow field were studied in those experiments, including leading edge shape, viscous interaction parameter and deflection angles. Theoretical and computational work is used to develop additional scaling laws for peak heating, upstream effect and aerodynamic coefficients. The fig. 2.2 shows Typical configuration of a Scramjet intake and the associated flow features. The flow is from left to right. The free-stream Mach number and the unit Reynolds number are 7.7 and $4.1 \times 10^6 m^{-1}$, respectively. A supersonic combustion Ramjet's (Scramjet intake) consists primarily of multiple external compression ramps, followed by an internal part. The incoming flow is compressed by the oblique shock waves produced by the ramps and the cowl lip. It is an application of SWBLI which can be simulated by a ramp in a channel

test case.

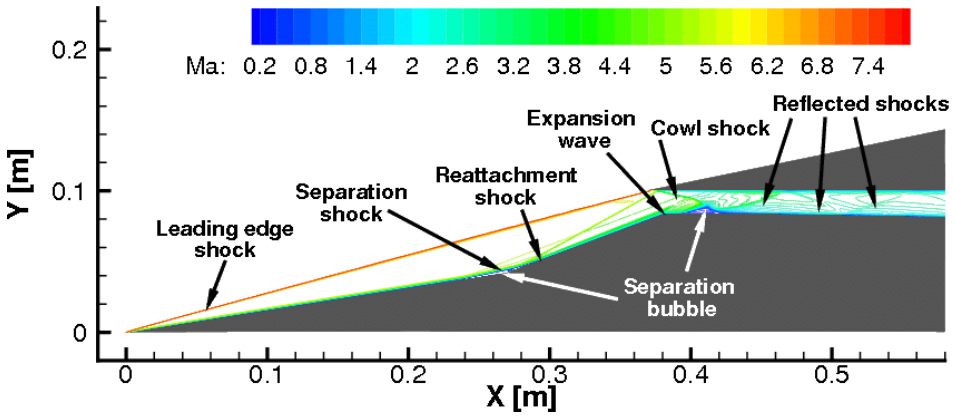


FIGURE 2.2: Typical configuration of a Scramjet intake and the associated flow features [14].

Since SWBLI was discovered to be dangerous, several studies have been conducted to reduce it. There are several control mechanisms available in the literature, such as the use of a blunt leading edge, mass injection or suction, vortex generator energy deposition pressure feedback channel, and so on. Townsend's [45] paper was regarded as one of the first attempts at SWBLI control. It presented effects of leading edge blunt nose for flow separation characteristics on a compression ramp, with the implementation of bluntness, a decrease in separation bubble size and plateau pressure was observed. Furthermore, the SWBLI was found to be significantly affected by changes in upstream local mach numbers and Reynolds numbers. The experimental investigations by Holden and Micheal [21] presented for low Reynolds number and high mach number flow over a flat-plate wedge. These tests were done on sharp and blunt leading-edge structures. As a result, larger separation for well separated flow with increase in bluntness and in contrast decrease in separation was observed. This research also included associations that could be used to estimate the plateau pressure and minimum angle needed to produce separation. The studies carried out by Gray and Rhudy [18] found experimental data on various Mach numbers, Reynolds numbers and wall to free stream total temperature ratios. The impact of bluntness and cooling of walls on supersonic laminar flow separation was examined. For the intermediate blunt radius, the upstream extent was higher than other categories. In addition, wall cooling decreases the ramp inducing area of viscous interaction irrespective of transition location.

Besides that, an experimental investigation of the boundary layer separation associated with the compression corner was performed by Lewis [26] for cooling effect in the wind

tunnel. For both laminar and transitional interactions, the surface pressure distribution was found to be dependent on Reynolds number. Chapman et al. [9] proposed that the principle of free interactions is empirically evaluated and also showed that the adiabatic structure was quite good approximated but it failed to associate the cool case with the adiabatic wall. Inger and Zee [23] presented the effect of mass transfer by suction and blowing for transonic shock wave and turbulent boundary layer. It concludes that a greater fraction of separation reduction was observed in both the techniques. Lately, Pasquariello et al. [34] studied a passive flow-control technique for shock wave, and uses a control configuration which consists of local suction and injection through a pressure feedback duct. Numerical analysis was carried out for free stream mach number 2.3 resulting in a impinging oblique shock wave generated by an 8.8° wedge interacting with a turbulent boundary layer. The suction angle was changed while the blowing position remained the same. This results in reduced size of separation zone. By using this technique, turbulence amplification was significantly reduced at the interaction region.

2.3 Parallel Computation

2.3.1 OP2

OP2 [16] is an Application Programming Interface (API) with associated libraries for solving Unstructured Mesh based algorithms on multi-core CPUs and clusters of GPUs. The application uses source-source translation and compilation to generate appropriate back-end code for the various target platforms e.g. OpenMP [4], MPI [3], CUDA [31], OpenCL [19], etc., for execution on different back-end hardware. OP2 is the successor of OPLUS (Oxford Parallel Library for Unstructured Solvers) which was developed by University of Oxford in 1933 for a research project [7]. OP2 draws on its predecessor's features but develops a "active" code-generation approach to concurrent many / multi-core architectures.

2.4 Problem statements and Objectives

Over the years, many industries and academic communities are using only multi-core CPUs for the numerical analysis and Computation Fluid Dynamics (CFD) simulation. Graphic Processing Unit (GPUs) have gained popularity in recent years for CFD simulations. In comparison to sequential code, GPU can reduce the computational run time by an order of magnitude. Interestingly, there are different architectures available to develop a solver to run on GPUs, they are growing quickly and it is a difficult job to develop several codes for different architectures.

Dipankar Das et al. [10] developed a solver for evaluating the energy deposition based drag reduction technique for Earth and Mars atmosphere. This solver is implemented to run on CPUs which increases the computational time. The present work is mainly focused on developing a CO_2 reaction based solver for hypersonic flows and to parallelise it using OP2 API which then runs on GPUs as well as CPUs.

2.4.1 Objectives

The main purpose of the present work is to simulate the high temperature effects for hyper-velocity vehicles. In order to have wide applicability, the presented solver must be capable of simulating low and high enthalpy flow rates. Moreover, in future it can be enhanced by implementing various schemes, higher order accuracy and different physical characteristics like drag coefficient, energy deposition etc. Therefore, the main objectives of the present work are listed below:

1. Development of an unstructured finite volume Navier-Stokes reacting gas flow solver encapsulated with AUSM δ scheme and viscous fluxes using OP2 framework.
2. Validation for the in-house developed solver against published data.
3. Comparison run time between sequential and parallel solver.

Chapter 3

Numerical Methodology

3.1 Governing Equations

The Governing Equations for this solver contain Navier-stokes and Euler equations. These calculations are based on the principles of gas dynamic equations like conservation of mass, conservation of momentum and conservation of energy. In addition, to the Navier-Stokes equations, species continuity equations must be solved. The source term of these added equations is responsible for the production rate of species that is negligible in the low non-reactive gas flow. The following are the vector forms of the coupled Navier-Stokes and species continuity equations for 2-D axisymmetric laminar viscous compressible flows:

$$\frac{\partial U}{\partial t} + \frac{\partial E_1}{\partial x} + \frac{\partial F_1}{\partial y} + S + \alpha \left(S_I - S_v \right) = \frac{\partial E_v}{\partial x} + \frac{\partial F_v}{\partial y} \tag{3.1}$$

Here,

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ C_1 \\ . \\ . \\ C_{N-1} \end{pmatrix}, E_I = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho E + p)u \\ uC_1 \\ . \\ . \\ uC_{N-1} \end{pmatrix}, F_I = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho E + p)v \\ vC_1 \\ . \\ . \\ vC_{N-1} \end{pmatrix} \tag{3.2}$$

$$E_v = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} - q_x - \sum_{i=1}^{N} h_i C_i \bar{u}_i \\ -C_1 \bar{u}_1 \\ . \\ . \\ -C_{N-1}\bar{u}_{N-1} \end{pmatrix}, S = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ S_1 \\ . \\ . \\ S_{N-1} \end{pmatrix}.$$

$$F_v = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} - q_y - \sum_{i=1}^{N} h_i C_i \bar{v}_i \\ -C_1 \bar{v}_1 \\ . \\ . \\ -C_{N-1}\bar{v}_{N-1} \end{pmatrix}, S_I = \frac{1}{y} \begin{pmatrix} \rho u \\ \rho uv \\ \rho v^2 \\ (\rho E + p)v \\ vC_1 \\ . \\ . \\ vC_{N-1} \end{pmatrix}$$

$$S_v = \frac{1}{y}\begin{pmatrix} 0 \\ \tau_{xy} - \frac{2}{3}y\frac{\partial(\mu v/y)}{\partial x} \\ \tau_{yy} - \tau_{\theta\theta} - \frac{2}{3}\mu\frac{v}{y} - \frac{2}{3}y\frac{\partial(\mu v/y)}{\partial y} \\ u\tau_{xy} + v\tau_{yy} - q_y - \frac{2}{3}\frac{\mu v^2}{y} - \frac{2}{3}y\frac{\partial(\mu uv/y)}{\partial x} - \sum_{i=1}^{N} h_i C_i \bar{v}_i \\ -C_1\frac{\bar{v}_1}{y} \\ . \\ . \\ -C_{N-1}\frac{\bar{v}_{N-1}}{y} \end{pmatrix}$$

Where U is the conserved variable vector, E_1 and F_1 are convective flux vectors in x and y directions and S is source term vector. The inviscid and viscous axisymmetric source terms are represented as S_I and S_v respectively, α is a constant ($\alpha = 1$ for 2- D axisymmetric, 0 otherwise), E_v and F_v are the viscous flux vectors in x and y directions. Further, the physical variables are density (ρ), velocity (u) in x direction, velocity (v) in y direction, pressure (p), temperature (T), internal energy ($e = \sum_{i=1}^{N} e_i \frac{C_i}{\rho MW_i}$) and total energy ($E = e + \frac{1}{2}\rho\,(u^2 + v^2)$). The total number of species is N and $e_i = h_{fi}^0 + \int_{T_R}^{T} C_{pi}\mathrm{dT} - R_u T$ is the molar energy of i^{th} species of C_i, MW_i, h_{fi}^0 and C_{pi} are the mass concentration, molecular weight, heat of formation, and specific heat at constant pressure, respectively. Here, R_u is the universal gas constant and T_R is the reference temperature. The shear stress components are given by,

$$\tau_{xx} = \mu\left(\frac{4}{3}\frac{\partial u}{\partial x} - \frac{2}{3}\frac{\partial v}{\partial y}\right) \tag{3.3}$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right),\ \tau_{yy} = \mu\left(\frac{4}{3}\frac{\partial v}{\partial y} - \frac{2}{3}\frac{\partial u}{\partial x}\right)$$

$$\text{and } \tau_{\theta\theta} = \left[\frac{-2}{3}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) + \frac{4}{3}\frac{v}{y}\right]$$

Where, q_x and q_y are heat dissipation in x and y directions, τ_{xx}, τ_{xy} are shear stress components and $\tau_{\theta\theta}$ is the shear stress component for axi-symmetry.

The mixture pressure is predetermined as summation of partial pressure of individual species.

$$p = R_u T \sum_{i=1}^{N} \frac{C_i}{MW_i} \tag{3.4}$$

The above equation is based on the iterative use of Newton-Raphson method for generating temperature from internal energy.

$$\rho e = \sum_{i=1}^{N} \frac{C_i}{MW_i}(h_{fi}^0 + \int_{T_R}^{T} C_{pi} dT + h_{fi}^0) - p \tag{3.5}$$

3.2 Chemical Kinetics

To model the flight conditions of the Martian atmosphere, a separate finite volume related inviscid non-equilibrium flow solver has been designed specifically for the chemical reaction of carbon dioxide flow. A chemistry model of eight species (N_2, O_2, N, O, NO, CO_2, CO, and C) containing ten elementary reactions is implemented in all the cases discussed in following chapters. In general, a series of N_R elementary reversible reactions involving N species can be described as follows:

$$\sum_{j=1}^{N} v'_{ij} n_j \rightleftharpoons \sum_{j=1}^{N} v''_{ij} n_j \tag{3.6}$$

Here, i=1,2 ..., N_R, v'_{ij} and v''_{ij} are the stoichiometric coefficients for species j appearing as a reactant in the ith forward and backward reactions respectively and molar concentration for j species is $n_j = \frac{C_j}{MW_j}$. For the rate constant of a reaction i, the Arrhenius rate expression is being used, which is defined as,

$$k_i = A_i T^{m_i} e^{\frac{-E_i}{R_u T}} \tag{3.7}$$

Where, A_i, m_i are constants and E_i is activation energy. After that,the difference in molar concentration of species j is summarized as,

$$S_j = MW_j \sum_{i=1}^{N_R} (v''_{ij} - v'_{ij})(k_{fi} \prod_{l=1}^{N} n_l^{v'_{il}} - k_{bi} \prod_{l=1}^{N} n_l^{v''_{il}}) \tag{3.8}$$

Where, forward and backward rate constants for reaction i are represented as kfi and kbi respectively. In the table 3.1 gives the details of chemical reactions and corresponding

specific reaction-rate constants.

S.NO	**Forward reaction**	**Kfi (cm3/ mole sec)**	**Kbi (cm3/ mole sec)**
1	O2 + M - > 2O + M	$9.1 \times 10^{18} T^{1.0} e^{-5.937 \times \frac{10^4}{T}}$	$9.1 \times 10^{15} T^{-0.5}$
2	N2 + M - > 2N + M	$2.5 \times 10^{19} T^{-1.0} e^{-1.132 \times \frac{10^5}{T}}$	$1.5 \times 10^{18} T^{-1.0}$
3	NO + M - > N + O +M	$4.1 \times 10^{18} T^{-1.0} e^{-7.533 \times \frac{10^4}{T}}$	$3.5 \times 10^{18} T^{-1.0}$
4	CO + M- > C + O + M	$4.5 \times 10^{19} T^{-1.0} e^{-1.289 \times \frac{10^5}{T}}$	$1.0 \times 10^{18} T^{-1.0}$
5	CO2 + M - > CO + O + M	$3.7 \times 10^{14} e^{-5.25 \times \frac{10^4}{T}}$	$2.4 \times 10^{15} e^{-2.184 \times \frac{10^3}{T}}$
6	N2 + O - > NO + N	$7.4 \times 10^{11} T^{-0.5} e^{-3.794 \times \frac{10^4}{T}}$	$1.6 \times 10^{11} T^{0.5}$
7	NO + O - > O2 + N	$3.0 \times 10^{11} T^{0.5} e^{-1.946 \times \frac{10^4}{T}}$	$9.5 \times 10^{9} T^{1.0}$
8	CO + O - > C + O2	$2.7 \times 10^{12} T^{0.5} e^{-6.945 \times \frac{10^4}{T}}$	$9.4 \times 10^{12} T^{0.25}$
9	CO2 + O - > CO + O2	$1.7 \times 10^{13} e^{-2.65 \times \frac{10^4}{T}}$	$2.5 \times 10^{12} e^{-2.4 \times 10^3}$
10	CO + N - > NO + C	$2.9 \times 10^{11} T^{0.5} e^{-5.363 \times \frac{10^4}{T}}$	$2.6 \times 10^{10} T^{0.5}$

TABLE 3.1: Chemical reactions and specific reaction-rate constants for chemical non-equilibrium calculations

3.3 Finite Volume Method

There are three widely used numerical method-based solvers in Computational Fluid Dynamics (CFD) to solve the governing equations i.e. Navier -Stokes equations. The three techniques are Finite Volume Method (FVM), Finite Element Method (FEM) and Finite Difference Method (FDM). Finite volume method (FVM) is a widely used discretization method for the variety of fluid flow simulations. “Finite volume” refers to the small volume

surrounding each node point on a mesh. In the finite volume method, volume integrals in partial differential equations are converted into surface integral forms of conservation laws (energy, momentum, mass) are used in this method. FVM can be most efficiently used for structured and unstructured grids and it is especially well-suited for complex geometries. In addition, FVM can perform the simple integration of equations through non-overlapping control volume parameters. The implementation of FVM has two related methods, which are classified for control volume frameworks i.e. cell-centered scheme and cell-vertex scheme.

In this present solver a cell centered scheme was implemented due to its ease of application. As shown in fig. 3.1 in the cell center approach the flow quantities are stored in the centroid of grid cells. Hence, the control volumes are identical to grid cells. The solver is developed for unstructured grids.

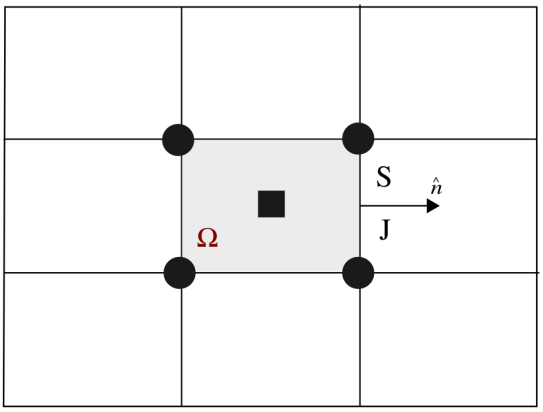


FIGURE 3.1: Representation of cell centered FVM scheme

3.3.1 Mathematical formulation of cell centered FVM

Steps for FVM discretization:

- Writing the required governing equation in its integral form.
- Applying Gauss Divergence Theorem.
- Finally, we will get a discretized form which can be solved using a computer ("Flux terms").

Considering 2D-axisymmetric Navier-Stokes equations discussed above, eq. (3.1) can be written in integral form as,

$$\int_{\Omega} \left(\frac{\partial U}{\partial t} + \frac{\partial E_1}{\partial x} + \frac{\partial F_1}{\partial y} - \frac{\partial E_v}{\partial x} - \frac{\partial F_v}{\partial y} + S + \alpha \left(S_I - S_v \right) \right) \mathrm{d}\Omega = 0 \tag{3.9}$$

Rearranging the equation as:

$$\int_{\Omega} \frac{\partial U}{\partial t} \mathrm{d}\Omega = - \int_{\Omega} \left(\frac{\partial E_1}{\partial x} + \frac{\partial F_1}{\partial y} - \frac{\partial E_v}{\partial x} - \frac{\partial F_v}{\partial y} + S + \alpha \left(S_I - S_v \right) \right) \mathrm{d}\Omega \tag{3.10}$$

Left hand side of eq. (3.10) can be discussed as:

$$\int_{\Omega} \frac{\partial U}{\partial t} \mathrm{d}\Omega = \frac{\partial}{\partial t} \int_{\Omega} U \mathrm{d}\Omega = \frac{\mathrm{d}}{\mathrm{d}t} \left(\bar{U} \Omega \right) = \Omega \frac{\mathrm{d}}{\mathrm{d}t} \bar{U} \tag{3.11}$$

Where, $\bar{U} = \frac{\int_{\Omega} U \mathrm{d}\Omega}{\int_{\Omega} \mathrm{d}\Omega}$

Considering, right hand side of eq. (3.10),

$$\int_{\Omega} \left(\frac{\partial E_1}{\partial x} + \frac{\partial F_1}{\partial y} - \frac{\partial E_v}{\partial x} - \frac{\partial F_v}{\partial y} + S + \alpha \left(S_I - S_v \right) \right) \mathrm{d}\Omega = 0 \tag{3.12}$$

The equation can rearrange as,

$$\int_{\Omega} \Big(\nabla \cdot \left(H_I - H_v \right) + S + \alpha \left(S_I - S_v \right) \Big) \mathrm{d}\Omega = 0 \tag{3.13}$$

Where, $H_I = [E_I \; F_I]$, $H_v = [E_v \; F_v]$ and $\nabla = \left[\frac{\partial}{\partial x} \; \frac{\partial}{\partial y} \right]$

Let, $H = (H_I - H_v)$

Applying Gauss divergence theorem to the first term of eq. (3.12):

$$\int_{\Omega} \nabla \cdot H \mathrm{d}\Omega = \int_{S} H \cdot \hat{n} \mathrm{dS} \tag{3.14}$$

Where, $\hat{n}$ is unit control surface normal pointing outward of the control volume, n_x and n_y are x and y components of the unit vector $\hat{n}$.

$$\int_{S} H \cdot \hat{n} \mathrm{dS} = \sum_{J=1}^{n_f} H_J \cdot \hat{n_J} \Delta S_j = \sum_{J=1}^{n_f} H_{\perp J} \Delta S_J \tag{3.15}$$

Here, ΔS_J is the face length of a control volume. Where, the summarization of inviscid (H_I) and viscous normal (H_v) fluxes goes across the faces of a control volume is represented as $H_\perp$. Therefore,

$$H_\perp = H_{I_\perp} - H_{V_\perp} \tag{3.16}$$

Whereas,

$$H_{I_\perp} = \begin{bmatrix} \rho u_\perp \\ \rho u u_\perp + p n_x \\ \rho v u_\perp + p n_y \\ (\rho e + p)\, u_\perp \\ C_1 u_\perp \\ . \\ . \\ C_{N-1} u_\perp \end{bmatrix}, H_{V_\perp} = \begin{bmatrix} 0 \\ n_x \tau_{xx} + n_y \tau_{xy} \\ n_x \tau_{yx} + n_y \tau_{yy} \\ n_x \Theta_x + n_y \Theta_y \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \tag{3.17}$$

$U_\perp$ is the contravariant velocity can be representing as:

$$u_\perp = u \cdot n_x + v \cdot n_y \tag{3.18}$$

Work done by viscous force and heat transfer by heat conduction in energy equation can be written as,

$$\Theta_x = u\tau_{xx} + v\tau_{xy} - q_x - \sum_{i=1}^{N} h_i C_i \bar{u_i} \tag{3.19}$$

$$\Theta_y = u\tau_{yx} + v\tau_{yy} - q_y - \sum_{i=1}^{N} h_i C_i \bar{v_i} \tag{3.20}$$

Likewise, source terms in eq. (3.11) can be present as;

$$\int_\Omega \Big(S + \alpha \left(S_I - S_v \right) \Big) \mathrm{d}\Omega = \Omega \bar{S_c} + \alpha \Omega \bar{S} \tag{3.21}$$

Here,

$$\bar{S} = \bar{S_I} - \bar{S_v} \tag{3.22}$$

Consequently, the integral form of the governing Equation eq. (3.10) can be written from equations eq. (3.11), eq. (3.15) and eq. (3.21) as,

$$\Omega_i \frac{\mathrm{d}\overline{U_i}}{\mathrm{d}t} + \sum_{J=1}^{n_f} H_{\perp_J} \Delta S_J + \Omega_i \overline{S_{ci}} + \alpha\, \Omega_i \bar{S}_i = 0 \tag{3.23}$$

$$\frac{\mathrm{d}\overline{U_i}}{\mathrm{d}t} = -\frac{1}{\Omega_i} \sum_{J=1}^{n_f} H_{\perp_J} \Delta S_J - \overline{S_{ci}} - \alpha\, \overline{S_i} = R(\overline{U_i}) \tag{3.24}$$

Therefore, this is the semi-discretized governing equation.

3.4 Spatial Discretization

The residual $R(\overline{U_i})$ must be evaluated in order to solve the semi-discretized governing equation described in eq. (3.24). Flux calculation methodologies determine the precision of the solution. Thus, convective and viscous fluxes at the faces of the control volume are used in this solver. The current solver incorporates various flux evaluation schemes, which is described in the subsections below.

3.4.1 Discretization of Convective Fluxes

3.4.1.1 AUSM

The primary aim of developing a numerical analysis algorithm has been to maximize both accuracy and efficiency. This is particularly important when solving complex problems involving the Navier-Stokes equations, which might also involve turbulence models and chemical species equations. The AUSM scheme was first introduced by Liou and Steffen [27]. Many researchers use the Advection Upstream Splitting Method (AUSM) to discretize convective fluxes. In the supersonic and hypersonic flow regimes, this scheme can be used to solve a wide variety of problems. The scheme begins by recognising that the inviscid flux is made up of two theoretically distinct parts namely, convective and pressure fluxes. AUSM scheme is applied to the grid at faces where it calculates the flux on that face using

the corresponding left and right cell values and unit normal of that face.

The flux vector is given by:

$$\vec{F_c} = V \cdot \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho H \end{bmatrix} + \begin{bmatrix} 0 \\ n_x p \\ n_y p \\ n_z p \\ 0 \end{bmatrix} \tag{3.25}$$

The first term in eq. (3.25) represents scalar quantities, which are convected by the contravariant velocity. In contrast, the pressure term is governed by the acoustic wave speed. Where V is the contra variant velocity derived in eq. (3.18) gives the velocity of normal to the surface element. It is known as the dot product of velocity and unit normals n_x, n_y are the unit normal components in their respective directions.

The pressure at the face of the control volume is obtained from splitting

$$p_{I+1/2} = p_L^+ + p_R^- \tag{3.26}$$

Where, $p_{I+1/2}$ is the pressure at the face of the control volume with the split pressure given by;

$$p_L^+ = \begin{cases} p_L & \text{if} \quad M_L \geq 1 \\ \frac{p_L}{4}\left(M_L + 1\right)^2 \left(2 - M_L\right) & \text{if} \quad |M_L| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.27}$$

$$p_R^- = \begin{cases} 0 & \text{if} \quad M_R \geq 1 \\ \frac{p_R}{4}\left(M_L - 1\right)^2 \left(2 + M_R\right) & \text{if} \quad |M_R| < 1 \\ p_R & \text{otherwise} \end{cases} \tag{3.28}$$

It is also possible to use the following lower-order expansion for $\left|M_{L/R}\right| < 1$.

$$p^{\pm}_{L/R} = \frac{p_{L/R}}{2}\left(1 \pm M_{L/R}\right) \tag{3.29}$$

We can also write AUSM in the form below.

$$\vec{F_c} = \frac{1}{2}(M_N)_{I+1/2}\left(\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho H\end{bmatrix}_L + \begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho H\end{bmatrix}_R\right) - \frac{1}{2}|(M_N)_{I+1/2}|)\left(\begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho H\end{bmatrix}_L - \begin{bmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ \rho H\end{bmatrix}_R\right) + \begin{bmatrix}0\\ n_x(p^+_L + p^-_R)\\ n_y(p^+_L + p^-_R)\\ n_z(p^+_L + p^-_R)\\ 0\end{bmatrix} \tag{3.30}$$

eq. (3.30) represents the convective flux.

The first term on the right-hand side of the above eq. (3.30) represents a Mach number-weighted average of left and right cells. The second term has dissipative character. It is scaled by the scalar value $\left|(M_n)_{I+1/2}\right|$.

M_n is the average of mach number used to evaluate the flux vector in eq. (3.30). The Mach number at the face is given as

$$(M_n)_{I+1/2} = M^+_L + M^-_R \tag{3.31}$$

$M_L and M_R$ are the left and right cells mach numbers obtained from the split mach numbers by evaluating from the Vanleers flux vector splitting schemes [46].

$$M_R^- = \begin{cases} 0 & \text{if} \quad M_R \geq 1 \\ \frac{-1}{4}(M_R - 1)^2 & \text{if} \quad |M_R| < 1 \\ M_R & \text{otherwise} \end{cases} \tag{3.32}$$

$$M_L^+ = \begin{cases} M_L & \text{if} \quad M_L \geq 1 \\ \frac{1}{4}(M_L + 1)^2 & \text{if} \quad |M_R| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.33}$$

M_L^+, M_R^- are the left and right cell Mach numbers respectively.

3.4.1.2 AUSM delta

AUSM proved to deliver a crisp resolution of strong shocks and accurate results for boundary layers. However, the original AUSM [27] was found to generate local pressure oscillations at shocks and in cases where the flow is aligned with the grid it was therefore suggested to switch at shocks to Van Leers scheme [37]. When the Mach number at face $(M_n)_{I+1/2}$ tends to zero the dissipation term will approach zero as well. Thus, any disturbances cannot be damped by the scheme. In order to solve the flow alignment problem, it was proposed to modify the scaling of the dissipation term as follows

$$\left|(M_n)_{I+1/2}\right| = \begin{cases} \left|(M_n)_{I+1/2}\right| & \text{if} \quad \left|(M_n)_{I+1/2}\right| > \delta \\ \\ \frac{(M_n)^2_{I+1/2} + \delta^2}{2\,\delta} & \text{if} \quad \left|(M_n)_{I+1/2}\right| \leq \delta \end{cases} \tag{3.34}$$

where, δ is a small value ($0 < \delta \leq 0.5$). In order to retain the accuracy of AUSM for boundary layers, the parameter δ could be reduced in the wall normal direction. The pressure term evaluation will remain the same as the AUSM scheme. It is proved to provide crisp shock better than AUSM scheme. [6].

In the present work, AUSM δ numerical schemes have been implemented in the code.

3.4.2 Discretization of Viscous Fluxes

The flow variables and their derivative must be known at the same place in order to calculate viscous fluxes on the face of a control volume. Consistency and simplicity in spatial discretization was ensured in the current numerical analysis by using the same control volume for viscous flux evaluation as for convective flux evaluation. The flow variables at the faces are calculated by averaging the node values. Using the discrete Gauss divergence theorem, the first derivative of the flow vector is determined at the cell centre [6]. Finally, these values are then distributed to the cell nodes and then calculated on the face with the average of node values. This technique avoids the decoupling of evaluated gradients at the boundaries as it usually happens in the simple cell averaging method at a face. Thus, values of the velocity components (u, v, w), the dynamic viscosity μ , and of the heat conduction coefficient k , which are required for the computation of the viscous terms and of the stresses. The process of finding the node values and cell gradient evaluation are explained in detail.

3.4.2.1 Calculation of Viscous flux

In order to calculate the viscous fluxes we entail node values, cell gradients and node gradients. To determine the node values we have to find the cell values from neighbouring cells and their cell areas. The fig. 3.2 shows a node surrounded by 4 cells, the node value is calculated by:

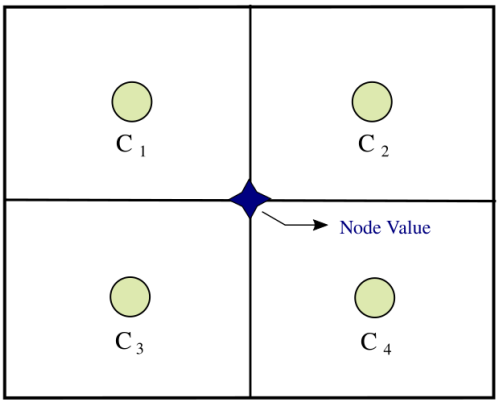


FIGURE 3.2: Node value schematic diagram

$$N_v = \frac{\sum_{i=1}^{4} C_i \times A_i}{\sum A_i} \tag{3.35}$$

where, N_v = Node value; C_i = Cell value; A_i = Cell area.

After calculating the node values we need to compute the cell gradients at the face using node values, unit normal and the face area. The fig. 3.3 shows the calculation of cell gradients for a single cell, the cell gradients are calculated by:

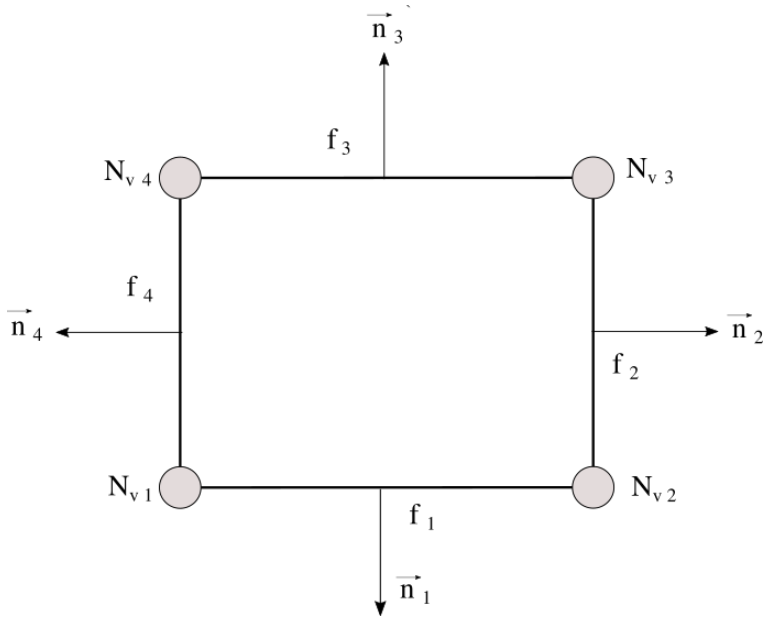


FIGURE 3.3: Cell gradient schematic diagram

$$\varnothing_1 = \frac{N_{v1} + N_{v2}}{2} \tag{3.36}$$

$$F_i = \varnothing_1 \times \vec{n_1} \times ds_1$$

$$\nabla = \frac{\sum_{i=1}^{4} \varnothing_i \times \vec{n_i} \times ds_i}{A_{cell}}$$

where, ∇ = Cell gradient at cell center; A_{cell} = Cell area; $ds_i = i^{th}$ Face area and $\varnothing_i = i^{th}$ face value.

Node gradients are calculated using the same relation as in eq. (3.35) and the average of node gradients is taken as the face gradient and the face values (average of node values). For viscous fluxes we need the gradients and the values (velocity, temperature, etc.) at face center.

3.5 Boundary Conditions

Any numerical simulation should only consider a portion of the physical domain or structure in question. The domain is truncated, resulting in artificial limits where we must prescribe values for such physical quantities. Furthermore, walls that are open to the flow reflect the physical domain's natural borders. The numerical treatment of boundary conditions requires extreme caution. A bad implementation can lead to erroneous simulations of the actual system. Furthermore, the scheme's stability and convergence speed can be affected adversely. Various boundary conditions considered in this work which are described in subsequent sections.

3.5.1 Inflow and Outflow boundary conditions

The number of variables that must be applied at the inflow or outflow boundary for a well-posed problem is determined by the characteristic principle [36] which depends on whether the flow is locally subsonic or supersonic. The focus of this study is on supersonic inflow and outflow conditions.

1. **Supersonic Inflow**

   For Supersonic inflow, all eigenvalues have the same sign. As a result, all flow variables at the supersonic inflow boundary are set to freestream values corresponding to typical hypersonic flight conditions or ground test conditions. Therefore, the values are specified based on Mach number (M_∞) and flow angles(angle of attack etc.)

2. **Supersonic Outflow**

   At the outflow boundary, where the flow field leaves the computational domain, the majority of the flow field is considered to be supersonic. In this case, all of the eigenvalues have the same symbol. As a result, the flow variables at the supersonic outflow boundary are extrapolated from the interior cell, taking into account the zero upstream effect of supersonic flow. A zeroth order extrapolation is used in this case.

3.5.2 Inviscid Wall boundary condition

Viscous forces adjacent to the wall are ignored and fluid thus slides freely across the surface under this condition. It is not zero, whereas the normal velocity portion is assumed to be nil and thus the wall behaves as impermeable. On the other side, no stream of symmetry crosses the wall boundary. Thus, all boundary conditions are mathematically identical, although their spatial significance is distinct. For these boundary conditions, the current solution uses a cell-mirror technique.

3.5.2.1 Mirror or Ghost cell approach

This solution offers the freedom to apply a flux calculation system on the wall as the waves develop. Thermodynamic and tangential velocities are obtained from the inner flow in this method. However, normal velocity is treated differently. The species density is equated to the interior cell density. Here, the graphical representation of the approach is shown in fig. 3.4 and the mathematical formulation is as follows, here the subscript i is interior cell and g is ghost cell:

$$u_{\perp g} = -u_{\perp i}$$

$$u_{\parallel g} = u_{\parallel i}$$

$$p_g = p_i$$

$$\rho_g = \rho_i$$

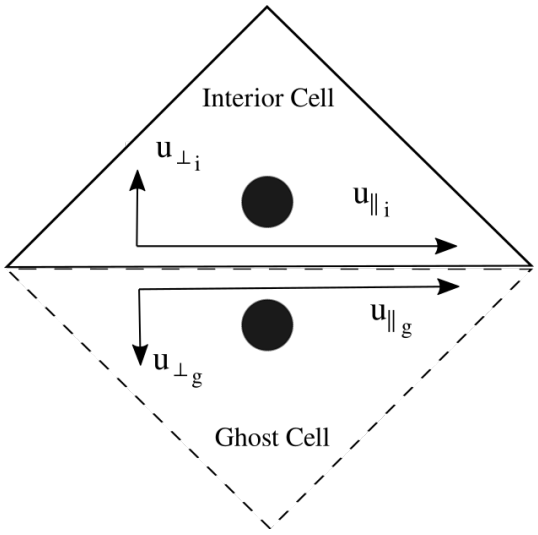


FIGURE 3.4: Mirror or Ghost cell approach [11]

3.5.2.2 Viscous Wall

Through the Ghost cell approach, viscous wall condition has been implemented for wall faces. This method shows that the tangential and regular velocity at the wall are both zero. From the immediate interior cell, the pressure values at the wall faces are extrapolated. Isothermal and adiabatic cases have different temperature boundary conditions for the wall. In the present solver we considered adiabatic wall. For adiabatic condition, the gradients of temperature are set as zero.

Hence by referring to fig. 3.4, for an adiabatic wall, the mirror cell approach gives,

$$u_{\perp g} = -u_{\perp i}$$

$$u_{\parallel g} = -u_{\parallel i}$$

$$p_g = p_i$$

$$\rho_g = \rho_i$$

$$E_g = E_i$$

3.6 Temporal discretization

The ordinary differential equations obtained by the spatial discretization are converted to a system of algebraic equations by the temporal discretization. There are two styles of time-marching methods: explicit and implied time-marching. Explicit methods are simple to use and have a low computational cost per time stage. The Courant-Friedrichs-Lewy (CFL) condition, however, limits the magnitude of the time step for an explicit method to ensure numerical stability. This instability problem can cause longer computation times, particularly when simulating viscous flows. With unconditionally stable implicit schemes, even larger time steps can be used for steady flow problems where higher time precision is not desired. While implicit formulations have a higher computational cost per time step, they can reduce the overall computational time by an order of magnitude as compared to explicit formulations by using a larger time step. We used explicit in the development of the solver and the scheme used is an explicit Euler scheme.

$$\frac{\mathrm{d}U_i}{\mathrm{d}t} = -\frac{1}{\Omega_i}\sum_{J\epsilon i} H_{\perp_J} \triangle S_J - \alpha S_i = R(U_i) \tag{3.37}$$

3.6.1 Euler time stepping

The simple explicit Euler scheme for time integration of eq. (3.37), leads to the fully discrete finite volume formulation for a i^{th} control volume as,

$$\frac{\triangle_t U_i}{\triangle t} = R(U_i^n) \tag{3.38}$$

$$\triangle_t U_i = U_i^{n+1} - U^n$$

$$U_i^{n+1} = U^n + \triangle t R(U_i^n)$$

$$U_i^{n+1} = U^n - \tfrac{\triangle t}{\Omega_i} \textstyle\sum_{J\epsilon i} H_{\perp_J} \triangle S_j - \alpha \triangle t S_i$$

So, with the known values of variables at the n^{th} state, $(n+1)^{th}$ state can be measured directly, and then time marching can be performed. However, since this scheme is only first order accurate in time, it is less suitable for time-dependent simulations that are unsteady.

Chapter 4

Semi Automatic Code Generation

When the number of processors grows, the volume of data transferred between main memory and processors increases as well, resulting in overheads and preventing an application from reaching the optimal level of parallelism, and also the traditional approach for GPU-driven parallelization is laborious, requiring intensive effort for hand-written code and optimize GPU code manually. It is difficult to programme on GPUs, as the compilers will not provide sufficient abstractions to compel programmes to understand many low hardware specifics. The OP2 framework optimization techniques solve this problem and help to increase parallelization efficiency by using a high level API and code translator.

In this chapter, we briefly describe about the OP2 framework, advantages of using it in parallelizing applications and difficulties faced in the code generation for OP2 by explaining some examples of kernel files and parallel loops. Later, the semi automatic code generation technique is described, this semi automatic code generation helps in generating the parallel loops automatically and corresponding kernel files from high level scripts.

4.1 OP2

For the MPI/PVm-based distributed memory performance of unstructured mesh algorithms in FORTRAN, the original O Plus library was created over 20 years ago [7]. Its second version, OP2, is built to exploit common multi-core and many-core hardware (GPU, AVX, etc.) in addition to distributed parallelization of the memory and allows to run one

multi-core/many-core node or a multi-core/many-core node cluster. OP2 currently only supports the development of code in C/C ++ and Fortran.

The OP2 methodology for creating executables for various back-end hardware consists of first pre-processing code written with the OP2 API to create architecture-specific code, and then integrating the created code with the required parallel interface (e.g. OpenMP, CUDA, MPI, etc.). For example, when generating back-end code for NVIDIA GPUs the main programme is parsed through the code generation tools, creating an updated main programme and a CUDA file. Each of the kernel functions has its own file in the CUDA file. Then the oplib.cu library is compiled and attached to it using a C-compiler (e.g. gcc) and the NVIDIA CUDA compiler (nvcc), controlled by a Makefile.

Over a broad variety of computer science and engineering applications, unstructured grids meshes have been and remain used. Unstructured mesh is being used to develop this solver. In computational fluid dynamics (CFD), computational electromagnetic (CEM), structural mechanics and general finite-element techniques, they were introduced to solve the partial differential equations. In general, the millions of elements in three dimensions are always necessary for the optimal solution, which results in large computational costs.

In contrast to structured meshes, unstructured meshes define the mesh topology using connectivity relevant information. The OP2 method for solving unstructured mesh problems includes, splitting the algorithm down into four sections: sets, data on sets, connectivity (or mappings) between the sets and operation over sets. [16]. A set includes nodes, edges, triangular or quadrilateral faces and various elements, depending on the specific application. Mappings between sets that describe how elements from one set connect with elements from another. This results in an API that can be used to describe any mesh or graph fully and abstractly.

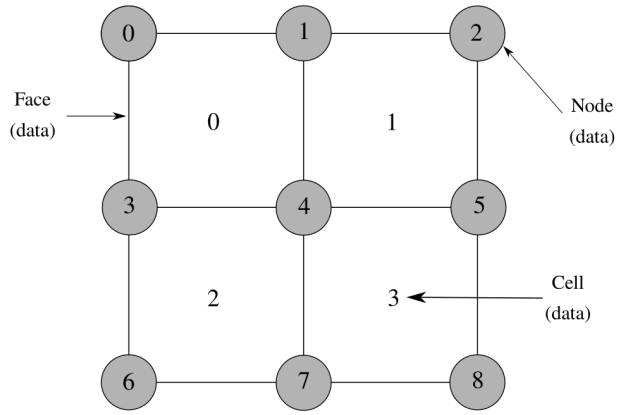


FIGURE 4.1: The mesh represented data layouts provided with OP2

fig. 4.1 shows an example of mesh that includes nodes and face data. The mesh can defined using OP2 API as follows:

```
op_set nodes = op_decl_set(nnodes,"nodes");
op_set faces = op_decl_set(nfaces,"faces");
op_set cells = op_decl_set(ncells,"cells");
op_set bfaces = op_decl_set(nbfaces,"boundary_faces");
```

where, nodes, faces, cells and boundary faces data is incorporated from mesh data. The connectivity is declared through the mappings between cells, nodes, faces and boundary faces.

```
op_map cell_node_map = op_decl_map(cells,nodes,4,cell_node,"pcell");
```

The op map deceleration defines this mapping, where the op cell node map has a dimension of 4. Each element of set cells is mapped with 4 different elements in set nodes. Similarly, each element of set cells mapped with 2 different elements in set faces and in set bfaces. Identically, each element of set faces and set bfaces is mapped with 2 different elements in set nodes.

```
op_map face_node_map = op_decl_map(faces,nodes,2,face_node,"pedge");
op_map face_cell_map = op_decl_map(faces,cells,2,face_cell,"pfacecell");
op_map boundary_node_map = op_decl_map(bfaces,nodes,2,boundface_node,"
    bound_node");
op_map boundary_cell_map = op_decl_map(bfaces,cells,1,boundface_cell,"
    bound_cell");
```

In the unstructured mesh code, all numerically intensive calculations can be represented as sets of operations. This refers to loops over a series of data, accessing data through the mappings (i.e. one level of indirection), doing certain calculations, and then writing back to the data arrays (possibly through the mappings). If the loop contains a mapping indirection, we call it an indirect loop; otherwise the loop is called a direct loop. In these loops, the OP2 API includes a parallel loop declaration syntax that allows to declare the computation over sets [16].

Consider the sequential loop below, which runs through each cell in the mesh. This loop is used to calculate cell quantities which are further used in the simulation. Each of the cells uses data values stored on the four nodes associated with that cell to update the data value.

```
inline void evaluate_cell_quantities(const double *point1,const double *
    point2, const double *point3,const double *point4, double *cell_area,
    double *cell_centroid)
{
    double x0, x2, x3, x1, y0, y2, y3, y1;

    x0 = point1[0]; x1 = point2[0];
    y0 = point1[1]; y1 = point2[1];

    x2 = point3[0]; x3 = point4[0];
    y2 = point3[1]; y3 = point4[1];

    double dx1 = x2-x0; double dy1 = y2-y0;
    double dx2 = x3-x1; double dy2 = y3-y1;

    cell_area[0]=(dx1*dy2-dx2*dy1)/2.0;  // cell area
    cell_centroid[0]=(x1+x2+x3+x0)/4.0;      // cell centroid x
    cell_centroid[1]=(y1+y2+y3+y0)/4.0;      // cell centroid y
}
```

This loop is declared by an application developer using the OP2 API along with the "elemental" kernel feature.

```
op_par_loop(evaluate_cell_quantities, "cell_quantities", cells,
            op_arg_dat(nodes, 0, cell_node_map, 2, "double", OP_READ),
            op_arg_dat(nodes, 1, cell_node_map, 2, "double", OP_READ),
            op_arg_dat(nodes, 2, cell_node_map, 2, "double", OP_READ),
            op_arg_dat(nodes, 3, cell_node_map, 2, "double", OP_READ),
            op_arg_dat(cell_area, -1, OP_ID, 1, "double", OP_WRITE),
            op_arg_dat(cell_centroid, -1, OP_ID, 2, "double", OP_WRITE));
```

OP2 manages the generation and parallelization of specific architectures. In this case, six arguments are used for the elemental kernel function, and a parallel loop declaration requires that the access method (OP WRITE, OP READ, etc.) be specified. OP ID shows that without indirection the cell to cell data should be accessed (i.e. directly). On the other hand, nodes are accessed using the indicated index (0, 1, 2, and 3) via the mapping of the cell node map. The dimension of the data (1, 2 in this example respectively) is also declared.

4.2 Semi Automatic Code Generation Using SymPy

As mentioned above, writing parallel loop and kernel files is error prone because of the complex parallel loop declarations required such as mappings, sets, operation over sets etc, and writing the kernel file is also time consuming. The other disadvantage of writing the OP2 code manually is it becomes static. If a new library arrives we have to port the entire code into that library API.

To overcome this, we are following a semi-automatic code generation approach. Such approaches are previously done in Finite Difference Method (FDM) [24] which is fully automatic code generation and is very beneficial. In the present work, the solver is developed using the symbolic Python package SymPy. SymPy is a python library for symbolic mathematics. It aims to become a full-featured Computer Algebra System (CAS) with a language that is as easy to understand and quickly expanded. SymPy is primarily written in Python [44].

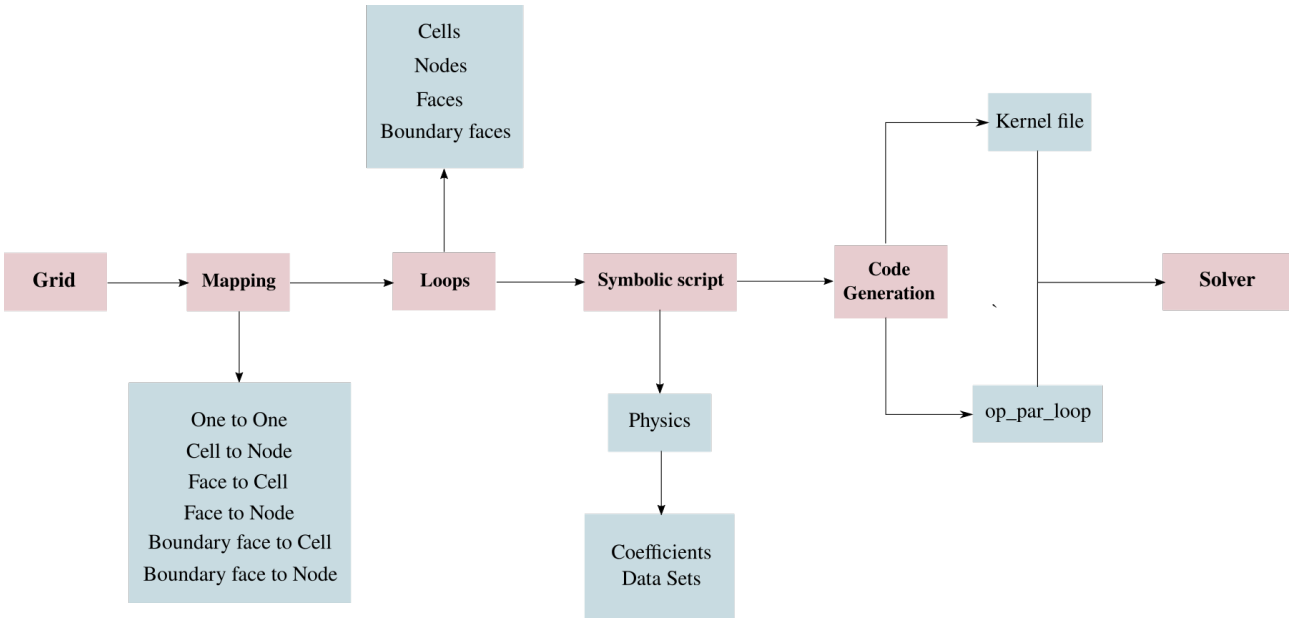


FIGURE 4.2: Flow chart of Code Generation

The fig. 4.2 shows the implementation of mathematical models symbolically. Grid is a mesh file for which mapping is done in multiple ways like cell-to-cell, node-to-node etc. By using data sets and coefficients from physics the required loops are written in Python script using SymPy. Some of the examples for loops are cell loops, face loops, node loops. For a cell loop we use cell-to-cell mapping. Similarly, for a node loop node-to-node mapping is used. Using the code generator we generate an op par loop and kernel file for the loops written in python script. The generated kernel file and op par loop are then copied into the solver code manually, thus it is referred as "Semi-Automatic" code generation. The generated solver code is then translated through an Application Programming Interface, OP2. The OP2 translator generates architecture-specific code, and then integrates the created code with the required parallel interface (e.g. OpenMP, CUDA, MPI, etc.). These kernel files are then compiled and executed respectively as shown in fig. 4.3.

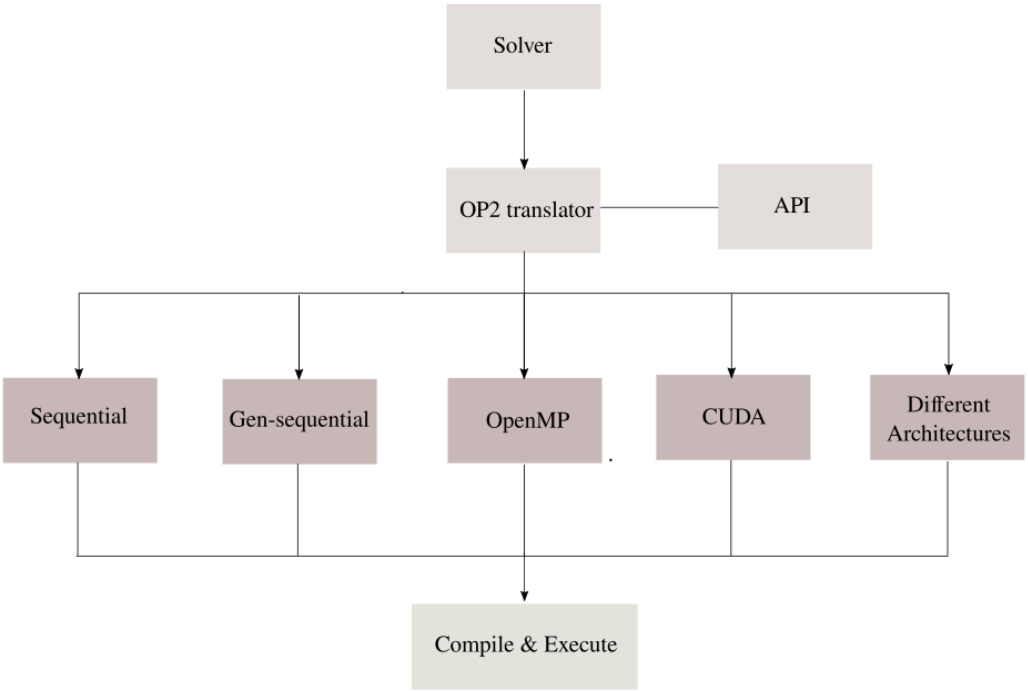


FIGURE 4.3: Flow chart of OP2 Framework

Chapter 5

Validation of in-house solver

The verification and validation process is an essential part of the CFD solver development process. In order to validate the developed solver, a canonical ramp in a channel test case is considered. The results obtained from the first order inviscid solver are validated qualitatively and quantitatively, as well as validated against experimental and analytical data, in this chapter. After validating the in-house solver the computational run times for 1000 iterations using different architectures are compared with the published data.

5.1 Ramp in a channel

5.1.1 Boundary Conditions

The geometry of supersonic aircraft engine inlets, which are wedge-shaped to compress air flow into the combustion chamber, is an example of this test case as shown in fig. 2.2. For the flow inside a ramp in a channel, an oblique shock wave is developed. It is a straightforward case in which the shock can be generated. When supersonic flow is turned on itself, it produces an oblique shock wave with a sharp edge. Across the shock, temperature, pressure, density, and air stream velocity are all minimized. The fig. 5.1 shows the boundary conditions that are used for inviscid and viscous test cases. The mesh with number of grid cells and nodes considered for inviscid and viscous are shown in table 5.1.

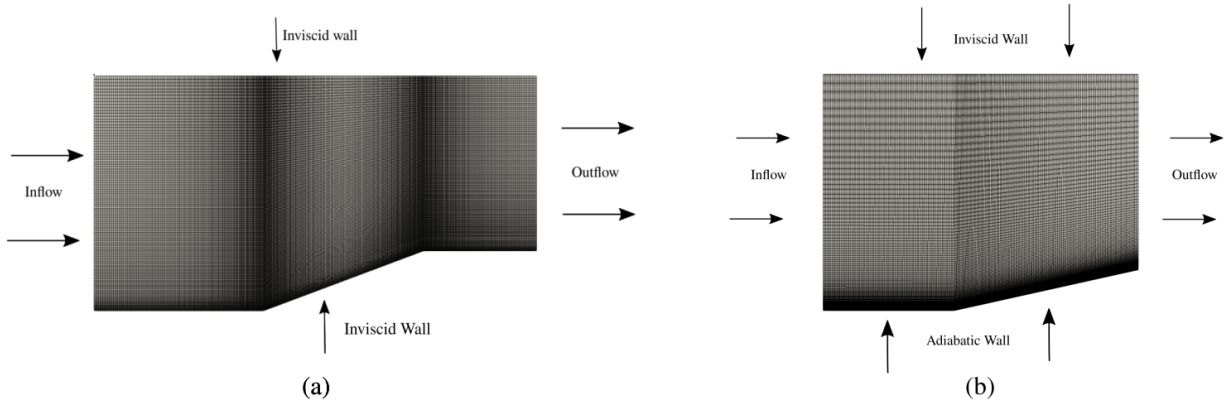


FIGURE 5.1: Boundary conditions for (a) Inviscid test case (Ramp angle, $\theta = 20°$) and (b) Viscous test case (Ramp angle, $\theta = 12.5°$)

	Inviscid		Viscous
Ramp Angle	10°	15°	12.5°
No. of Cells	41600	55900	28800
No. of Nodes	42051	56461	29161

TABLE 5.1: Number of cells and nodes for different test cases

5.1.2 Inviscid test case

Simulations are carried out to analyse the effect of ramp angle. Therefore, different ramp angles of $10°, 20°$ are considered here. Along with the no-slip boundary conditions, free stream conditions are static pressure 199.45Pa and static temperature 131.70K. Here, the simulation is carried out with three different inlet conditions that is with Mach numbers 4, 5 and 7. Then fig. 5.2 shows the pressure contours for different ramp angles by varying the incoming flow values. It is well known that high ramp angles produces a greater level of pressure difference on the incoming flow. Pressure plots over a line along x-axis for different mach numbers are shown in fig. 5.3.

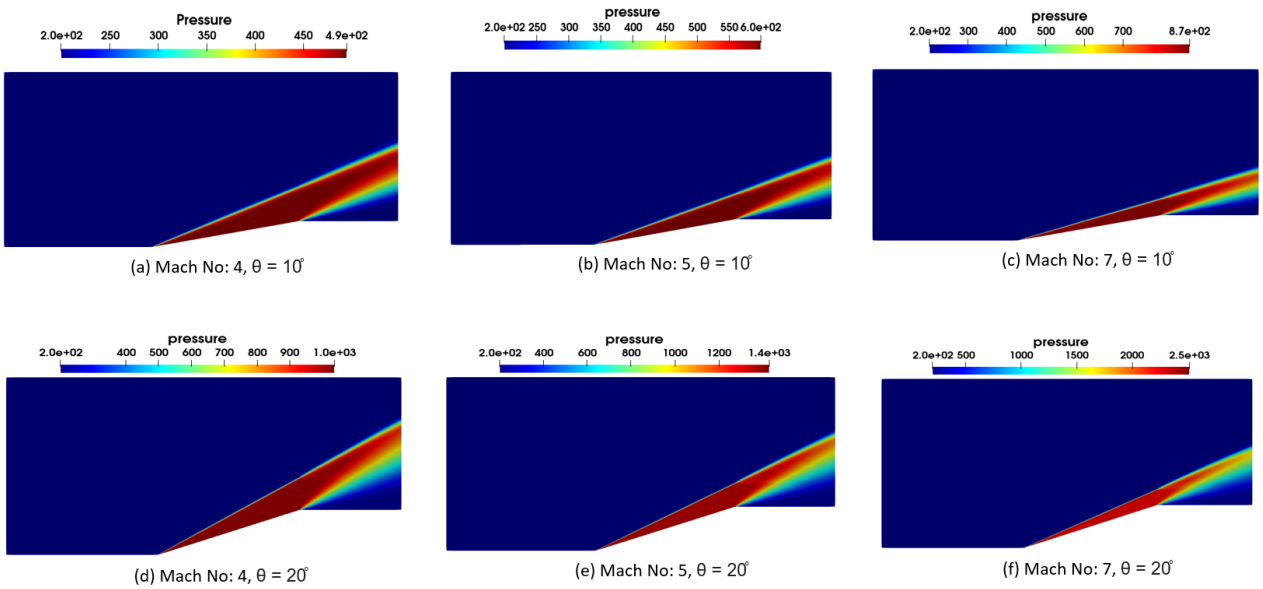
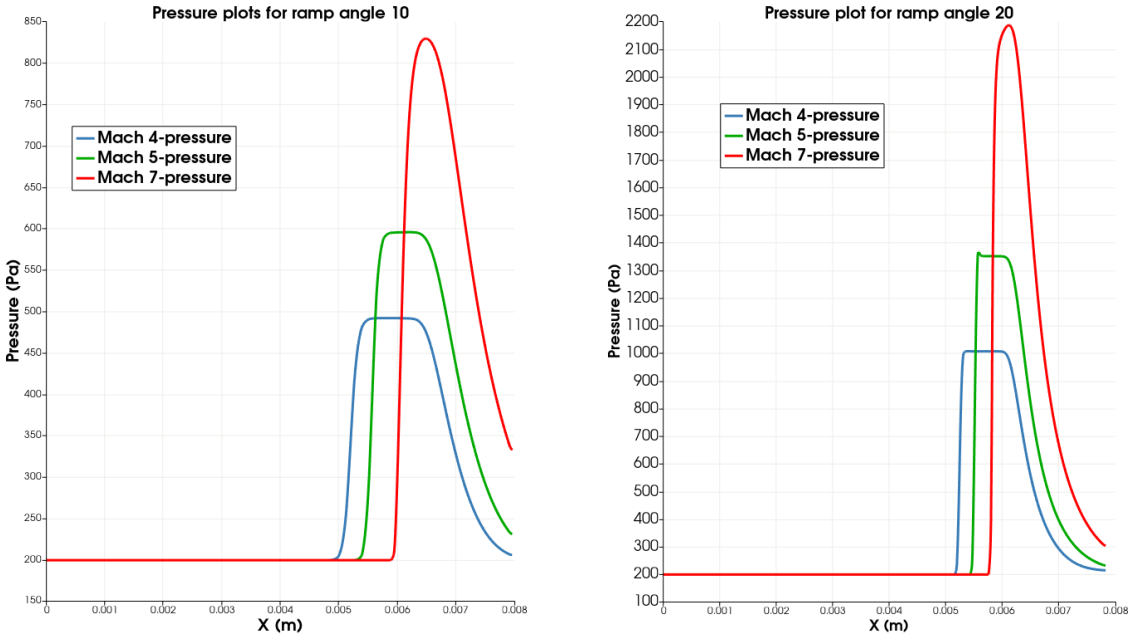


FIGURE 5.2: Pressure contours for Inviscid test case



(A) (B)

FIGURE 5.3: Pressure plots over a line along x-axis for different mach numbers

5.1.3 Viscous test case

Simulation is carried out for canonical ramp in a channel. In this test case, a ramp angle $\theta = 12.5°$ has been considered with input mach number of 5.0. The computational domain used for the present solver marked with boundary conditions is shown in fig. 5.1. This domain has been meshed with 28800 unstructured cells as given in table 5.1. Free stream conditions used in this study are static pressure 199.45Pa and static temperature 131.70K. A real gas model is implemented for this simulation. AUSM delta flux scheme with first order accuracy is used in this simulation. The pressure contours are shown in below fig. 5.4.

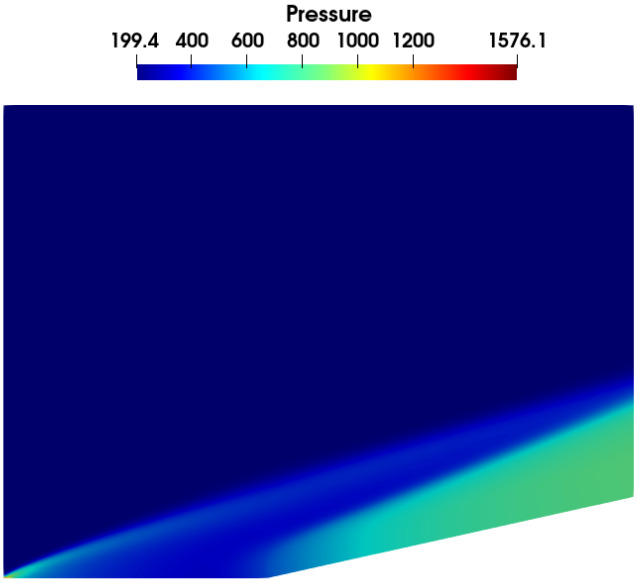


FIGURE 5.4: Pressure contours for viscous test case

5.1.4 Validation of in-house solver

The parameters considered for validation are shock wave angle (β), pressure ratio and temperature ratio. The values obtained from the in-house solver are compared with analytical predictions [41] and reference publication [10]. These comparisons are given in table 5.2. From the table, it is evident that the results of the in-house solver are in good agreement with the analytical predictions and reference data. Therefore, it can be concluded that the in-house solver is validated successfully for low enthalpy conditions.

Ramp angle θ	M	Shock wave angle (β)			Pressure ratio (P_2/P_1)			Temperature ratio (T_2/T_1)		
		Analytical	Reference	In-house solver	Analytical	Reference	In-house solver	Analytical	Reference	In-house solver
10	4	21.76	22.24	21.86	2.35	2.40	2.45	1.22	1.28	1.30
	5	18.90	18.85	18.79	2.82	2.95	2.98	1.29	1.38	1.39
	7	15.88	15.77	15.79	4.00	4.04	4.25	1.45	1.55	1.59
20	4	31.21	31.93	31.50	4.70	4.92	5.04	1.54	1.67	1.71
	5	28.55	29.13	28.79	6.29	6.41	6.77	1.74	1.76	1.95
	7	26.01	26.31	26.18	10.46	11.24	11.25	2.26	2.53	2.53

TABLE 5.2: Shock wave angle, pressure ratio and temperature ratio at different deflection angles and Mach numbers

5.2 Run time Comparison

The solver has achieved estimated results relative to the theoretical results and hence the speed comparison is carried out in different architectures. The machine that is considered here is DGX machine with NVIDIA V100 GPU, 80 CPUs and half terabyte RAM. The following solvers considered are (1) sequential, where OP2 does not do any optimization, (2) Gen-sequential which is an optimized version of sequential code and (3) OpenMP, version of the code generated using OP2 translator. We use the GNU compiler, $g++$ version 9.3.0, with -O3 optimization for the CPUs. In our case we use 34 cores in the DGX machine to run OpenMP. We compared computational run time and speedup of in-house solver (w.r.t Sequential) for different architectures for inviscid test cases which is given in table 5.3.

	Sequential	Gen-Sequential	OpenMP
Inviscid	99.494 sec	93.80 sec	7.319 sec
Speed-up	1	1.06	13.59

TABLE 5.3: Inviscid speedup comparison between different architectures (1000 iterations)

In table 5.4 the computational run times and speedup of in-house solver (w.r.t Sequential) are compared between reference solver [10] and various architectures for viscous test case.

	Reference solver	Sequential	Gen-Sequential	OpenMP
Viscous	1680 sec	87.556 sec	81.009 sec	8.788 sec
Speedup	0.052	1	1.08	9.963

TABLE 5.4: Viscous speedup comparison between different architectures and reference solver (1000 iterations)

Chapter 6

Conclusions and Future work

6.1 Conclusions

The principal objective of this thesis is to accelerate a hypersonic reaction based CO_2 solver for martian atmosphere. Hence, a finite volume method based solver having the potential to simulate hypersonic flows in Mars's atmosphere for 2-Dimensional unstructured test cases which accommodates eight species and their chemical kinetics is developed. The developed solver uses OP2 framework to accelerate on various computing architectures. The necessary kernel files and parallel loops for OP2 API are generated using semi-automatic code generation technique. For validating the solver, ramp in a channel test case for two different angles $(10°, 20°)$ are chosen. The results from the Inviscid first order accuracy run are then compared with analytical and reference data from [10]. The solver has achieved a speedup of 13x and 10x when compared to sequential run using OpenMP for Inviscid and Viscous solver respectively. The sequential run is 19x faster than the reference solver which also runs sequentially using a single core. The reason being, we have reduced the if conditions by the novel implementation of calculating C_p, C_v, energy etc,. Also we have increased computations and reduced memory footprint by re-calculating energy, C_p, C_v etc, as and when needed.

6.2 Future work

Although the present solver is capable of delivering faster and accurate results yet there are a few voids that are to be explored. The following are some of the exploration directions that assist in improving the solver's ability:

1. The present solver is limited to 2-D planar flowfield which can be enhanced by implementing 2-D axisymmetry flowfield.
2. The solver's potential can be further improved by adapting it to the 3-D domain. This would make studying different 3-D components in hypersonic flows extremely easier.
3. The order of accuracy implemented in the present solver is first order which can be improved by executing second order of accuracy using limiters [47].
4. Present solver's simple explicit Euler scheme can be upgraded to explicit Runge-Kutta scheme to achieve higher order temporal accuracy [25].

The following are some of the exploration directions that assist in improving the semi-automatic code generator:

1. Semi-automatic code generation can be improved further to fully automate the OP2 code generation which can generate the code for high level scripts [24].
2. The solver can be parallalized and run on GPUs using the required computational architecture.

Bibliography

[1] Jakob Ackeret, F Feldmann, and N Rott. "Investigations of compression shocks and boundary layers in gases moving at high speed". In: (1947).

[2] Blaise Barney. *Introduction to Parallel Computing Tutorial*. `https://hpc.llnl.gov/training/tutorials/introduction-parallel-computing-tutorial`.

[3] Blaise Barney. *Message Passing Interface (MPI)*. `https://hpc-tutorials.llnl.gov/mpi/`.

[4] Blaise Barney. *OpenMP Tutorial*. `https://hpc.llnl.gov/openmp-tutorial`.

[5] Michael H Bertram and Thomas A Blackstock. *Some simple solutions to the problem of predicting boundary-layer self-induced pressures*. National Aeronautics and Space Administration, 1961.

[6] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[7] David A Burgess, Paul I Crumpton, and Mike B Giles. "A parallel framework for unstructured grid solvers". In: *Programming environments for massively parallel distributed systems*. Springer, 1994, pp. 97–106.

[8] GRAHAM CANDLER. "Computation of thermo-chemical nonequilibrium Martian atmospheric entry flows". In: *5th Joint Thermophysics and Heat Transfer Conference*. 1990, p. 1695.

[9] Dean R Chapman, Donald M Kuehn, and Howard K Larson. "Investigation of separated flows in supersonic and subsonic streams with emphasis on the effect of transition". In: (1958).

[10] Dipankar Das et al. "Performance assessment of energy deposition based drag reduction technique for Earth and Mars flight conditions". In: *Acta Astronautica* 159 (2019), pp. 418–428.

[11] Siddesh Desai. "Numerical study of flow alteration techniques and high temperature effects at supersonic/hypersonic speeds". PhD thesis. 2019.

[12] Manuel J. Diaz. *Re-entry flow regions of the Apollo Command Module*. `https://space.stackexchange.com/questions/32857/what-was-the-nature-of-the-visible-part-of-the-viscous-wake-trailing-the-apollo`. 2018.

[13] Michael G Dunn and Sang-Wook Kang. *Theoretical and experimental studies of reentry plasmas*. Vol. 2232. National Aeronautics and Space Administration Washington, DC, 1973.

[14] Sarah Frauholz et al. "Numerical simulation of hypersonic air intake flow in scramjet propulsion using a mesh-adaptive approach". In: *18th AIAA/3AF International Space Planes and Hypersonic Systems and Technologies Conference*. 2012, p. 5976.

[15] GE Gadd, Douglas William Holder, and JD Regan. "An experimental investigation of the interaction between shock waves and boundary layers". In: *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 226.1165 (1954), pp. 227–253.

[16] Mike B Giles et al. "Designing OP2 for GPU architectures". In: *Journal of Parallel and Distributed Computing* 73.11 (2013), pp. 1451–1460.

[17] F Grasso and M Marini. "Analysis of hypersonic shock-wave laminar boundary-layer interaction phenomena". In: *Computers & fluids* 25.6 (1996), pp. 561–581.

[18] J Don Gray and RW Rhudy. "Effects of Blunting and Cooling on Separation of Laminar Supersoniic Flow". In: *AIAA Journal* 11.9 (1973), pp. 1296–1301.

[19] OpenCL Working Group. *The OpenCL C 2.0 Specification*. `https://www.khronos.org/registry/OpenCL/specs/2.2/pdf/OpenCL_C.pdf`.

[20] Jiaao Hao et al. "Comparison of transport properties models for numerical simulations of Mars entry vehicles". In: *Acta Astronautica* 130 (2017), pp. 24–33.

[21] Michael S Holden. "Boundary-layer displacement and leading-edge bluntness effects on attached and separated laminar boundary layers in a compression corner. II-experimental study". In: *AIAA journal* 9.1 (1971), pp. 84–93.

[22] HG Hornung. "Non-equilibrium dissociating nitrogen flow over spheres and circular cylinders". In: *Journal of Fluid Mechanics* 53.1 (1972), pp. 149–176.

[23] GR Inger and S Zee. "Transonic shockwave/turbulent-boundary-layer interaction with suction or blowing". In: *Journal of Aircraft* 15.11 (1978), pp. 750–754.

[24] Christian T Jacobs, Satya P Jammy, and Neil D Sandham. "OpenSBLI: A framework for the automated derivation and parallel execution of finite difference solvers on a range of computer architectures". In: *Journal of Computational Science* 18 (2017), pp. 12–23.

[25] Bibin John. "Numerical Investigations of Shock Wave Boundary Layer Interaction in Hypersonic Flows". PhD thesis. 2014.

[26] John E Lewis, Toshi Kubota, and Lester Lees. "Experimental investigation of supersonic laminar, two-dimensional boundary-layer separation in a compression corner with and without cooling." In: *AIAA journal* 6.1 (1968), pp. 7–14.

[27] Meng-Sing Liou and Christopher J Steffen Jr. "A new flux splitting scheme". In: *Journal of Computational physics* 107.1 (1993), pp. 23–39.

[28] R KENNETH LOBB. "Experimental measurement of shock detachment distance on spheres fired in air at hypervelocities". In: *AGARDograph*. Vol. 68. Elsevier, 1964, pp. 519–527.

[29] Robert Mitcheltree and Peter Gnoffo. "Wake flow about a MESUR Mars entry vehicle". In: *6th Joint Thermophysics and Heat Transfer Conference*. 1994, p. 1958.

[30] James N Moss and Graeme A Bird. "Direct simulation of transitional flow for hypersonic reentry conditions". In: *Journal of spacecraft and rockets* 40.5 (2003), pp. 830–843.

[31] NVIDIA. *CUDA C++ Programming Guide*. `https://docs.nvidia.com/pdf/CUDA_C_Programming_Guide.pdf`.

[32] Chul Park. "Assessment of a two-temperature kinetic model for dissociating and weakly ionizing nitrogen". In: *Journal of Thermophysics and Heat Transfer* 2.1 (1988), pp. 8–16.

[33] Chul Park. "Review of chemical-kinetic problems of future NASA missions. I-Earth entries". In: *Journal of Thermophysics and Heat transfer* 7.3 (1993), pp. 385–398.

[34] Vito Pasquariello et al. "Large-eddy simulation of passive shock-wave/boundary-layer interaction control". In: *International Journal of Heat and Fluid Flow* 49 (2014), pp. 116–127.

[35] Carlos Alberto Rocha Pimentel and Annibal Hetem Jr. "Computation of air chemical equilibrium composition until 30000K-Part I". In: *Journal of Aerospace Technology and Management* 3.2 (2011), pp. 111–126.

[36] T J&amp Poinsot and SK Lelef. "Boundary conditions for direct simulations of compressible viscous flows". In: *Journal of computational physics* 101.1 (1992), pp. 104–129.

[37] R Radespiel and N Kroll. "Accurate flux vector splitting for shocks and shear layers". In: *Journal of Computational Physics* 121.1 (1995), pp. 66–78.

[38] Aniello Riccio et al. "Optimum design of ablative thermal protection systems for atmospheric entry vehicles". In: *Applied Thermal Engineering* 119 (2017), pp. 541–552.

[39] M Sharma et al. "Experimental and numerical investigation of hypervelocity carbon dioxide flow over blunt bodies". In: *Journal of thermophysics and heat transfer* 24.4 (2010), pp. 673–683.

[40] Lonnie Shekhtman. *With Mars Methane Mystery Unsolved, Curiosity Serves Scientists a New One: Oxygen*. `https://www.nasa.gov/feature/goddard/2019/with-mars-methane-mystery-unsolved-curiosity-serves-scientists-a-new-one-oxygen`. 2019.

[41] L C Squire. "Modern Compressible Flow: with historical perspective—Second edition. JD Anderson. McGraw-Hill Book Co (UK), McGraw Hill House, Shoppenhangers Road, Maidenhead, Berks, SL6 2QL. 1990. 650 pp. Illustrated.£ 18.95." In: *The Aeronautical Journal* 95.947 (1991), pp. 248–248.

[42] Prabhakar Subrahmanyam. "Development of an interactive hypersonic flow solver framework for aerothermodynamic analysis". In: *Engineering Applications of Computational Fluid Mechanics* 2.4 (2008), pp. 436–455.

[43] Ghislain Tchuen and David E Zeitoun. "Effects of chemistry in nonequilibrium hypersonic flow around blunt bodies". In: *Journal of thermophysics and heat transfer* 23.3 (2009), pp. 433–442.

[44] SymPy Development Team. *SymPy*. `https://www.sympy.org/en/index.html`. 2021.

[45] James Courtland Townsend. *Effects of leading-edge bluntness and ramp deflection angle on laminar boundary-layer separation in hypersonic flow*. National Aeronautics and Space Administration, 1966.

[46] Bram Van Leer. "Flux-vector splitting for the Euler equation". In: *Upwind and high-resolution schemes*. Springer, 1997, pp. 80–89.

[47] Venkat Venkatakrishnan. "Convergence to steady state solutions of the Euler equations on unstructured grids with limiters". In: *Journal of computational physics* 118.1 (1995), pp. 120–130.

[48] XY Wang et al. "Assessment of chemical kinetic models on hypersonic flow heat transfer". In: *International Journal of Heat and Mass Transfer* 111 (2017), pp. 356–366.