NUMERICAL SIMULATION OF PULSE DETONATION PHENOMENA IN A PARALLEL ENVIRONMENT

by

PRASHAANTH RAVINDRAN

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2005

To my family and friends for their continued support.

ACKNOWLEDGEMENTS

First and foremost, I thank Dr. Frank Lu for his support and incredible patience with me. His motivation has always kept me on my toes and I wouldn't have come to this point in my degree, the graduating part, if it wasn't for him. I also would like to thank Dr. Donald Wilson, for he was instrumental in reducing my stage fright with all the seminars I had to take before coming to this point. I am deeply indebted to Ms. Rafaela Bellini, Dr. Huiyuan Fan and Mr. Philip Panicker for all the help I got and their valuable time. I enjoyed working with all of you in Aerodynamic Research Center, the only place which felt more like home than anything else.

I cannot leave out Dr. Daniel New from Temasek Laboratories, Singapore. I enjoyed the little amount of time I spent with you and the everlasting impression you left behind. Some day we will meet again.

I am grateful to Mr. Sriram Venkatraman for all the suggestions he had given me and for tolerating my ignorance.

I owe my fascination with the world of Linux, open source and Latex typesetting to Venkat and Nikhil, my partners-in-crime. Without my compadres, without those late-night-early-morning discussions, without the amount of information we shared, I would not have been what I am now. I owe my existence to my parents, without whom I would not have learned about life. I can never repay the love they had given me. I am indebted to my significant other, Harini, the person who completes me. If I have left out anyone, it was purely unintentional and due to lack of sleep added with an overdose of caffeine.

Rationality is the recognition of the fact that existence exists, that nothing can alter the truth and nothing can take precedence over that act of perceiving it, which is thinking... -Ayn Rand,Atlas Shrugged

August 16, 2005

ABSTRACT

NUMERICAL SIMULATION OF PULSE DETONATION PHENOMENA
IN A PARALLEL ENVIRONMENT

Publication No. _____

Prashaanth Ravindran, M.S.

The University of Texas at Arlington, 2005

Supervising Professor: Frank K.Lu

The objective of this work was to develop a parallel algorithm that would be used in the simulation of the detonation process in the chamber of a pulse detonation engine. The emphasis is laid on reducing computation time while maintaining the accuracy of the solution and subsequently developing a numerical solution to be in agreement with real-world physical characteristics of a detonation wave initiation, build-up and progression. The flow is assumed to be unsteady, inviscid and non heat conducting. To adhere to real time effects, the flow equations are coupled with finite rate chemistry and the vibrational energy equation are based on a two-temperature model, to account for possible vibrational non-equilibrium. Finite Volume formulation is employed to ensure conservation and to allow proper handling of discontinuities. Runge-Kutta integration scheme has been utilized to obtain a time-accurate solution, with Roes flux difference splitting scheme applied to cell face fluxes. For higher-order spatial accuracy, MUSCL technique is employed. Equation stiffness has been taken care of by observing point implicit treatment of the source terms and detonation is initiated with the application of a localized

hot-spot. The parallel algorithm has been developed using Message Passing Interface standard developed by the Argonne National Laboratory for the purposes of solving equations in a distributed environment. A proto-cluster of Beowulf type consisting of 8-nodes has been assembled and made operational, and an algorithm which performs space-time calculations simultaneously on the nodes has been successfully developed. A two-step global model for Hydrogen-Air mixture has been selected for validating the parallel algorithm with existing results, to establish veracity and accuracy while reducing computation time to almost a fourth. Excellent agreement has been found on comparison of the results with the same code when solved in a single processor.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

High-speed non-equilibrium flows have received much attention recently due to renewed interest in single-stage and two-stage-to-orbit vehicles, as well as hypersonic airbreathing cruise vehicles. Such vehicles may require novel propulsion devices such as high speed SCRAMjets being studied by the Air Force Hypersonic System technology Program (HySTP) [1]-[5]. The Pulse detonation engine is another potential concept for high-speed propulsion which relies on gaseous detonations for thrust production.

Detonation phenomena have been systematically examined theoretically, experimentally and computationally now for over a century, beginning with a one-dimensional propagating detonation front as a discontinuity according to Chapman-Jouguet (CJ) theory [6]-[8]. The theory of Zel'dovich[9] , Von Neumann[10] and Doering[11], which has come to be known collectively as the ZND detonation model, represents the detonation as a confluence of a shock wave moving at detonation speed, followed by a chemical reaction zone of finite length. This relatively simple model which represents the chemical reaction as a single forward rate process, is known to capture many, if not most, of the essential physical phenomena associated with detonations[12][13].

Detonation, though destructive in nature, is a very efficient combustion process that couples the energy release in combustion to shock waves, generating extremely high pressures directly from the combustion process. A pulse detonation engine (PDE) is a propulsion device using the high pressure generated by repetitive detonation waves in a combustible mixture [14]. The PDE concept holds promise for high thrust density in a

constant volume device requiring little or no rotating machinery. Furthermore, PDE's can be operated on a wide range of flight speeds, from low subsonic to high supersonic, regardless of the engine size [15].

The pulse detonation engine can be classified as an unsteady propulsion engine, which operates in an intermittent manner governed by a cycle frequency, which is defined as the inverse of time required to complete a full detonation cycle. A cycle consists of the following distinct processes:

1. Filling the chamber with fuel/air mixture
2. Detonation initiation
3. Detonation wave propagation
4. Gas expansion in the chamber to cause reduced pressure for refill level.

While many fundamental issues regarding ignition, transition and propagation need to be addressed, numerical modeling for obtaining solutions for the above mentioned has been very successful.

Overviews of past and ongoing numerical simulations of PDE's are described in recent articles by Kailasanath [16]-[19]. Time accurate computational fluid dynamics methods can be used to perform cycle analysis and performance optimization of the PDE from the simulations of the corresponding flow fields with variations in design parameters.

The aim of this research is to present an unsteady numerical simulation of detonation in a hydrogen-air mixture in a parallel environment. The goal was to find an accurate and fast solution using a parallel language called Message Passing Interface (MPI) to simultaneously solve the governing equations in a small cluster having 8 nodes, without any loss in accuracy of real-time physics and efficiency. In this chapter, pertinent detonation physics is reviewed briefly. Past and ongoing computational research is then summarized, concluded by a brief description of my work.

1.2 History and Review of Detonation Physics

The idea of detonation came into existence after disastrous explosions in coal mines more than a hundred years ago. The previous study of combustion concerned a slow process called deflagration. At a first glance it was puzzling, how the slow subsonic combustion could produce such strong mechanical effects. The understanding came after French investigators did laboratory experiments on combustible mixtures in glass tubes [20].

Another source of our knowledge in detonation came from pure theory. Reimann had shown that the movement of compressible gas leads to the formation of breaks with abrupt change of pressure, temperature and density even with smooth initial conditions. The exact thermodynamics of the shock waves was discovered by Hugoniot. The generalization to chemically active media was not difficult after that and so the detonation theory of Chapman and Jouguet was born at the start of the 20th century.

The shock wave amplitude was dependent on the external causes of the shock. The detonation is a self-sustained process. Its properties must depend solely on energy, density and other characteristics of the initial mixture. But formally the equation of Chapman and Jouguet were consistent with a set of final states. The authors had to put forward a new principle (minimum entropy and minimum velocity) in order to choose one point in the set[2][3]. The principle worked well and is in good agreement with experiments. Yet there was no firm theoretical foundation for almost 40 years. It was Zeldovich, Doering and von Neumann who independently found the underlying idea of combustion initiated by the shock wave going ahead of the chemical reaction zone (ZND model). The ZND description neglects transport processes and assumes one-dimensional flow.

The further development of gas detonation has shown a very complicated structure. The idealized ZND model turned out to be unstable in the realistic case of chemical reactions, which are strongly temperature dependent.

A very important topic is the formation of detonation waves out of slow combustion. The landmark here was Shchelkin's experiment the influence of a spiral which turbilzing the gas motion before the flame [21]. This gave rise to what we now call deflagration-to-detonation transition (DDT).

In the continued absence of more appropriate models, the C-J and ZND models are widely used detonation velocities and their variation with the composition of the mixtures and temperature, pressure and density ratios across the leading front are almost invariably computed with a generally satisfied accuracy based on the one-dimensional models [22].

1.3 Importance of Computational Fluid Dynamics in Detonation

The computation of real gas flows has been an active research area for almost 3 decades.The effort resulted in a wealth of information regarding numerical solutions to high temperature non-equilibrium flows [23]-[29].Most of the computer codes developed for chemical and thermal non-equilibrium flows were developed using upwind or flux split algorithms, which yielded accurate solutions of shock wave dominated flows due to good shock capturing capabilities. Extensions of the flux-vector split schemes and flux-difference split scheme of the Roe type to non-equilibrium flows were given by Grossman and Cinella [30]. The same for a more general gas model was given by Liu and Vinokur [31].

Two approaches in the solution of non-equilibrium flows were standard: one is uncoupling the chemical reaction and thermodynamic equations and solving them separately [32]-[34] and the other is to solve them simultaneously in a fully coupled fashion.

The latter usually poses problems with stiffness in the system of equations, and results in a very small time step for a stable time-marching solution.

CFD work on PDEs progressed following the development of numerical methods on high temperature non-equilibrium flow. Cambier and Adelman [35] performed quasi one-dimensional computations with a shock capturing TVD monotone algorithm for multiple species that is second order accurate in space. Eidelmann [36]-[39] performed computational studies using a second order Godunov solver on unstructured grids. He and Karagozian used Weighted Essentially Non-Oscillatory Method, a derivative of the ENO method, which is 5th order accurate, for spatial interpolation of the system of governing equations and TVD Runge-Kutta for time discretization [40].

1.4 Scope of Current Work

The objective of the current work is to develop a time efficient parallel algorithm for simulating 2-D model of the Pulse Detonation Engine. Kim et al performed computational studies using a combination of point-implicit treatment and Local Ignition Averaging applied to a global 2-step reaction model for efficient time accurate solution of a propagating detonation wave. Roe's Flux-Difference split scheme is combined with second order Runge-Kutta integration scheme for an accurate shock capture in space and time[41].

The MPI standard parallel language is used to perform a parallel run of this numerical model on a Network of Workstations. The MPI standard defines the user interface and functionality for a wide range of message passing capabilities. Implementations are available on a wide range of machines. The major goal of MPI, as with my parallel program, is a degree of portability across different machines.

This parallel algorithm is validated by comparing it with a single-process code. Extensive calculations are performed with the integration scheme of different orders in

space-time, and with different mesh sizes to select the most efficient resolution of the physical process.

CHAPTER 2

MATHEMATICAL FORMULATION

2.1 Introduction

A set of coupled partial differential equations that describe gas-phase reactive flow fields is derived here, and which will be used to calculate initiation and propagation of detonation waves through fuel-air mixture in a pulse detonation engine. The flow equations used will be non-heat-conducting and inviscid since the major physical processes associated are inviscid phenomena like shock compression of the gas mixture, chemical reactions in the shock compressed region, wave interactions, detonation wave phenomena and expansion of burned gases [42].

The flow fields are assumed to be accurately described by a continuum formulation, though the processes, which are being simulated, are ultimately molecular and atomic interactions in a reactive medium. The continuum formulation requires that the Knudsen number must be much less than one to ensure a large number of particles within a computational volume. This implies that there is little statistical variation at any point in the domain. The continuum approximation incorporates statistical averaging processes performed over small-scale particle phenomena. This natural process is one which small-scale phenomena are averaged to produce a global quantity called "chunking" by Hofstadter [43] and which has been used as the basis of many scientific studies. We use a macroscopically chunked fluid or continuum picture of the particles, which give us distinct chemical species.

At high temperatures, portions of the internal energy may be out of equilibrium. The internal energy of each species will generally depend upon the various internal en-

ergy modes such as translational and rotational, vibrational, electronic excitation, and free electron energy mode. An adequate description of the energy distribution is rather subjective, and depends upon the problem to be solved. The primary application in this work is to model a hydrogen-air reactive flow, and since the temperature range for this application is expected to be around 3000 K at most, the translational and rotational energy modes are fully excited, while the electronic excitation and the effect of ionization can be ignored.

The energy in the translational modes of all species is assumed to be characterized by a single temperature T, and the rotational state of the diatomic molecules is assumed to be in equilibrium with this translational temperature. Thus, a single temperature T is used to characterize the translational and rotational state of the gas mixture, which is assumed to be fully excited in the temperature range of interest.

A low-temperature model is selected to describe the remaining energy modes. The energy in the vibrational modes of all the molecules is assumed to be described by a single vibrational temperature T_v. This model has been assessed by Chul Park [44] in high-temperature flow fields and observed to yield accurate results while requiring much less computing time. More detailed thermal models can be constructed in which the vibrational energy of each molecular species is modeled by its own vibrational temperature. This requires additional conservation equations of vibrational energy for each species, and thus more computing time. The application of a multi-vibrational temperature model is also limited by the availability and accuracy of the relaxation time data required to describe the energy exchanges due to collisions among species.

The feasibility of the two-temperature model to be more effective than a single temperature model is debatable. The accuracy of the solution when comparing a single temperature model and a two-temperature model for this particular case was mostly similar, with drastic changes occurring only in very high temperature regimes. Since

the temperatures involved did not reach such high levels, the single temperature models in fact reduce computation time significantly. But, the research work maintained a two-temperature model to maintain uniformity and to furnish precise comparison data between the normal algorithm and parallel algorithm.

The other main ingredients of the formulation include a chemical kinetic model to ensure accurate prediction of the chemical composition in the mixture, proper description of species and mixture thermodynamic properties including possible excitation of internal energy modes at high temperature, and vibrational energy relaxation process.

2.2 Governing Equations

The time-dependent conservation equations governing inviscid, non-heat-conducting, reacting gas flow in which thermal nonequilibrium is modeled with a two-temperature approximation are derived in this section. The conservation equations for the individual species are derived first, and then these are combined to yield the complete set of equations. This more general approach will make clear the underlying assumptions embodied in the governing equations.

The mass conservation equation for species s of the gas mixture is given by [45]

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho_s u_{sj}\right) = w_s \tag{2.1}$$

The species momentum, total energy, and vibrational energy conservation equations are written as

$$\frac{\partial}{\partial t}\left(\rho_s u_{si}\right) + \frac{\partial}{\partial x_j}\left(\rho_s u_{si} u_{sj}\right) = -\frac{\partial p_s}{\partial x_i} + P_{si} \tag{2.2}$$

$$\frac{\partial}{\partial t}\left(\rho_s E_s\right) + \frac{\partial}{\partial x_j}\left[\rho_s\left(E_s + \rho_s\right) u_{sj}\right] = Q_s \tag{2.3}$$

$$\frac{\partial}{\partial t}\left(\rho_s e_{v,s}\right) + \frac{\partial}{\partial x_j}\left(\rho_s e_{v,s} u_{sj}\right) = Q_{v,s} + w_s e_{d,s} \tag{2.4}$$

where u_{si} is the velocity of the species s in the i direction and w_s, P_{si}, Q_s represent the mass, momentum and energy transfer rates, respectively, of species s from the interaction with other species in the mixture. The first term in the right hand side of the vibrational conservation equation, $Q_{v,s}$ represents the vibrational energy exchange rate of species s due to the relaxation process with translational energy, and the second term, $w_s e_{d,s}$ represents the amount of vibrational energy gained or lost due to production or depletion of species s from chemical reactions.

For a gas mixture of N_s species, this approach would entail solving $5N_s$ equations simultaneously for a two-dimensional problem, and the evaluation of the momentum transfer term P_{si}. This may not be feasible for a multi-dimensional flow filed with some species. The problem can be simplified by using the mass-averaged velocity u_i, and the diffusion velocity u_{Dsi} defined by

$$u_i = \sum_s \frac{\rho_s}{\rho} u_{si}, \quad u_{Dsi} = u_{si} - u_i \tag{2.5}$$

Using these new variables, the species mass conservation equation 2.1 becomes

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho_s u_j\right) = -\frac{\partial}{\partial x_j}\left(\rho_s u_{Dsj}\right) + w_s \tag{2.6}$$

The species momentum, total energy, and vibrational energy equations can be summed over all the species to yield the following mass-averaged equations:

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = -\frac{\partial p}{\partial x_i} \tag{2.7}$$

$$\frac{\partial}{\partial t}\left(\rho E\right) + \frac{\partial}{\partial x_j}\left[\rho\left(E+p\right)u_j\right] = -\frac{\partial}{\partial x_j}\sum_s h_s u_{Dsi} \tag{2.8}$$

$$\frac{\partial}{\partial t}\left(\rho e_v\right) + \frac{\partial}{\partial x_j}\left(\rho e_v u_j\right) = \sum_s Q_{v,s} + \sum_s w_s e_{d,s} - \frac{\partial}{\partial x_j}\sum_s h_{v,s} u_{Dsi} \tag{2.9}$$

where the total pressure and density are defined as the sum of the species quantities. In the momentum equation, the momentum transfer rates between species,P_{si}, as well as the diffusion flux terms identically sum to zero.

A simplification comes from the assumption that molecular diffusion is not significant and can be safely neglected when the chemical reactions and the corresponding energy release occur rapidly as in a detonation process [46]. When we assume zero diffusion velocity, the following simplified set of equations is obtained.

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho_s u_j\right) = w_s \tag{2.10}$$

$$\frac{\partial}{\partial t}\left(\rho u_i\right) + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = -\frac{\partial p}{\partial x_i} \tag{2.11}$$

$$\frac{\partial}{\partial t}\left(\rho E\right) + \frac{\partial}{\partial x_j}\left[\rho\left(E+p\right)u_j\right] = 0 \tag{2.12}$$

$$\frac{\partial}{\partial t}\left(\rho e_v\right) + \frac{\partial}{\partial x_j}\left(\rho e_v u_j\right) = \sum_s Q_{v,s} + \sum_s w_s e_{d,s} \tag{2.13}$$

The governing equations are written in the conservation law form which has the property that the coefficients of the derivative terms are either constant or, if variable, their derivatives appear nowhere in the equation. Normally this means that the divergence of a physical quantity can be identified in the equation. This form is advantageous in numerical simulations to correctly capture shock waves [47]. In a two-dimensional Cartesian coordinate system, the above take the following form:

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = S \tag{2.14}$$

where U is the vector of conserved variables, F and G are the convective flux vectors, and S is the vector of source terms. Each vector is written as

$$U = \begin{bmatrix} \rho_s \\ \rho u \\ \rho v \\ \rho e_v \\ \rho E \end{bmatrix},\ F = \begin{bmatrix} \rho_s u \\ \rho u^2 + p \\ \rho uv \\ \rho u e_v \\ \rho u E + pu \end{bmatrix},\ G = \begin{bmatrix} \rho_s v \\ \rho uv \\ \rho v^2 + p \\ \rho v e_v \\ \rho v E + pv \end{bmatrix},\ S = \begin{bmatrix} w_s \\ 0 \\ 0 \\ w_v \\ 0 \end{bmatrix} \tag{2.15}$$

In this equation, the subscript s ranges from 1 to N_s, where N_s is the number of species, followed by two momentum conservation equations for the mixture. The next row describes the rate of change in the vibrational energy, and the final row is the total energy conservation equation. In the above equation, u and v are the velocities in the x and y direction respectively, ρ is the mixture density, p is the pressure,e_v is the vibrational energy, and E is the total energy per unit mass of mixture. ρ_s is the s^{th} species density, w_s is the mass production rate of species s per unit volume, and w_v is the vibrational energy source term is defined as

$$w_v = \sum_s Q_{vs} + \sum_s w_s e_{d,s} \tag{2.16}$$

For an axisymmetric flow, the conservation equation takes the form as

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial x} + \frac{1}{y}\frac{\partial (yG)}{\partial y} = S \tag{2.17}$$

where U, F and G are the same as in equation 2.15, but S is given by

$$S = \begin{bmatrix} w_s \\ 0 \\ p/y \\ w_v \\ 0 \end{bmatrix} \tag{2.18}$$

2.3 Thermodynamic Properties

A general representation of species internal energy includes a portion of the internal energy in thermodynamic equilibrium and the remaining portion in a nonequilibrium state. The equilibrium portion of the internal energy is the contribution due to translational and internal modes that can be assumed to be in equilibrium at the translational temperature T. The remaining nonequilibrium portion is the contribution due to inter-

nal modes that are not in equilibrium at the translational temperature T, but may be assumed to satisfy a Boltzmann distribution at a different temperature.

For the temperature range of interest in the present study, the rotational mode is assumed to be fully excited and in equilibrium with translational temperature T, while the electronic excitation and free electrons modes can be safely ignored. Thus, the only remaining energy mode that could be in nonequlibrium with translational temperature T is the vibrational energy mode. Therefore, the species internal energy based on the two-temperature model can be written as follows:

$$e_s = e_{q,s}\left(T\right) + e_{v,s}\left(T_v\right) \tag{2.19}$$

where $e_{q,s}$ is the equilibrium portion of the internal energy and $e_{v,s}$ is the vibrational energy which is not in thermodynamic equilibrium. The equilibrium portion of the energy can be further defined as

$$e_{eq,s} = \int_{T_{ref}}^{T} \left(C_{v,t}^{s} + C_{v,r}^{s}\right) d\tau + e_{s,o} \tag{2.20}$$

where T_{ref} is the reference temperature, $e_{s,o}$ is the energy of formation, and $C_{v,t}^{s}$ and $C_{v,r}^{s}$ are translational and rotational portion of specific heat at constant volume, respectively. Since the translational and rotational modes are assumed to be fully excited, $C_{v,t}^{s}$ and $C_{v,r}^{s}$ can be written as

$$C_{v,t}^{s} = 1.5\bar{R}/M_s \tag{2.21}$$

$$C_{v,r}^{s} = \begin{cases} \bar{R}/M_s, \text{ diatomic molecule} \\ 1.5\bar{R}/M_s, \text{ polyatomic molecule} \end{cases} \tag{2.22}$$

where R is the universal gas constant and M_s is the molecular weight of the species s. The energy of formation $e_{s,o}$ can be obtained from readily available heat of formation data as

$$e_{s,o} = h_{s,o} - \frac{\bar{R}}{M_s} T_{ref} \tag{2.23}$$

Therefore, the equilibrium portion of energy can be written as follows:

$$e_{eq,s}(T) = K_s \frac{\bar{R}}{M_s}(T - T_{ref}) - \frac{\bar{R}}{M_s} T_{ref} + h_{s,o} \tag{2.24}$$

where Ks is 1.5, 2.5 and 3.0 for monatomic, diatomic or linear polyatomic, and nonlinear polyatomic species respectively.

The heat capacity of the vibrational energy mode can now be obtained from the fact that the translational and rotational heat capacities are independent of the temperature. This can be evaluated by utilizing the readily available curve fit for total heat capacity evaluated at temperature T_v and subtracting out the constants obtained from the translational and rotational heat capacities as follows:

$$C^s_{v,v}(T_v) = C^s_v(T_v) - C^s_{v,t} - C^s_{v,r} \tag{2.25}$$

where

$$C^s_v(T_v) = C^s_p(T_v) - \frac{\bar{R}}{M_s} \tag{2.26}$$

C^s_v and C^s_p are specific heat at constants volume and constant pressure respectively, and the curve fit data for $C^s_v(T_v)$ can be found in the following form [48][49]

$$C^s_p(T) = \frac{\bar{R}}{M_s} \sum_{k=1}^{5} A^s_k T^{k-1} \tag{2.27}$$

Therefore $C^s_{v,v}$ is obtained as follows:

$$C^s_{v,v}(T_v) = \begin{cases} \frac{\bar{R}}{M_s}\left(\sum_{k=1}^{5} A^s_k T^{k-1} - \frac{7}{2}\right), & \text{diatomic molecule} \\ \frac{\bar{R}}{M_s}\left(\sum_{k=1}^{5} A^s_k T^{k-1} - 4\right), & \text{polyatomic molecule} \end{cases} \tag{2.28}$$

The species vibrational energy $e_{v,s}$ can be obtained by integrating $C^s_{v,v}$ such that

$$e_{v,s}(T_v) = \int_{T_{ref}}^{T} C^s_{v,v}(\tau)\, d\tau \tag{2.29}$$

An alternative formulation using enthalpy relation may be used as follows[49]:

$$\begin{aligned} h_{v,s}\left(T_v\right) &= h_s\left(T_v\right) - \left(C^s_{p,t} + C^s_{p,r}\right)\left(T_v - T_{ref}\right) - h_{s,o} \\ h_s\left(T, T_v\right) &= h_{v,s}\left(T_v\right) + \left(C^s_{p,t} + C^s_{p,r}\right)\left(T - T_{ref}\right) + h_{s,o} \end{aligned} \tag{2.30}$$

where

$$C^s_p\left(T_v\right) = C^s_v\left(T_v\right) + \frac{\bar{R}}{M_s}, \;\; C^s_{p,r} = C^s_{v,r}$$

Here, $h_{v,s}\left(T_v\right)$ and $h_s\left(T_v\right)$ are the vibrational enthalpy and enthalpy per unit mass of species s and $h_s\left(T_v\right)$ which includes the heat of formation is readily available in the following form:

$$h_s\left(T_v\right) = \frac{\bar{R}}{M_s}\left(\sum_{k=1}^{5} \frac{A^s_k T^k_v}{k} + A^s_6\right) \tag{2.31}$$

Therefore, $e_{v,s}$ and e_s can be directly obtained from $h_{v,s}$ and h_s as follows:

$$\begin{aligned} e_{v,s}\left(T_v\right) &= h_{v,s}\left(T_v\right) \\ e_s\left(T, T_v\right) &= h_s\left(T, T_v\right) - \tfrac{\bar{R}}{M_s} T \end{aligned} \tag{2.32}$$

Thus, vibrational energy is obtained basically from the difference between total internal energy in equilibrium and the fully excited translational/rotational mode of internal energy. At the reference temperature where $T_v = T_{ref} = 298.15$ K, $h_s\left(T, T_v\right)$ in equation 2.31 is made equal to the heat of formation $h_{s,o}$. Therefore, $e_{v,s} = h_{v,s} = 0$ at and below this reference temperature, and the vibrational energy will increase gradually as temperature rises. However, in the temperature range slightly higher than the reference temperature, equation 2.31 yields small negative values of vibrational energy [5]. The negative vibrational energy is caused by the fully excited translational/rotational mode of heat capacity from the assumption that the translational/rotational energy mode is fully excited, which is untrue for this temperature range. Negative vibrational energy can cause problems in determining the vibrational temperature using a Newton-Raphson iteration method near the reference temperature. So, it becomes necessary to introduce

a limiter on the vibrational energy, which makes the vibrational energy zero whenever it reaches a negative value in the lower temperature ranges. The limiter introduced here will not have any effects when calculating the thermodynamic properties.

The mixture relations are expressed in the following way:

$$\begin{aligned} C_{p,q} &= \tfrac{1}{\rho}\sum_s \rho_s C^s_{p,q} \\ C_{v,q} &= \tfrac{1}{\rho}\sum_s \rho_s C^s_{v,q} \\ e_v &= \tfrac{1}{\rho}\sum_s \rho_s e_{v,s} \\ e &= \tfrac{1}{\rho}\sum_s \rho_s e_s \end{aligned} \tag{2.33}$$

where

$$\rho = \sum_{s=1}^{N_s} \rho_s$$

Total energy and total enthalpy are obtained from

$$\begin{aligned} E &= \tfrac{u^2+v^2}{2} + e \\ H &= E + \tfrac{p}{\rho} \end{aligned} \tag{2.34}$$

It is assumed here that each individual species behaves as a thermally perfect gas. With this assumption, the pressure can be given as

$$p = \sum_s p_s = \sum_s \frac{\rho_s \bar{R} T}{M_s} = \rho \tilde{R} T \tag{2.35}$$

where

$$\tilde{R} = \frac{1}{\rho}\sum_s \rho_s \frac{\bar{R}}{M_s}$$

2.4 Chemical Kinetics Model

High temperature flows typically involve some chemical reactions, and the time scale in which chemical reactions take place is important in the estimation of flow field properties. When a characteristic flow time is compared to a typical chemical reaction

time, three cases can occur. The first case is when a reaction time is much greater than the flow time, in which case a reaction has no enough time to occur. In this case, it is assumed that the flow is frozen with respect to that specific reaction. The second case is for a reaction time much shorter than the fluid dynamic time, wherein the reaction has virtually infinite time to evolve, and consequently an equilibrium state is attained during a fluid dynamic time scale. The third case is the general case of finite rate chemistry, when both times are of the same order. In this case, the actual kinetics of the reaction must be considered together with the fluid dynamic equations.

For accurate modeling of a detonation wave, especially in the detonation front where rapid chemical reactions take place in the shock compressed region, species continuity equations based on the chemical kinetics should be solved together with fluid dynamic equations to account for possible chemical nonequilibrium. The mass production rate of species s from the chemical reactions can be written as [49]

$$w_s = M_s \sum_{r=1}^{N_r} \left(\beta_{s,r} - \alpha_{s,r}\right) \left(R_{f,r} - R_{b,r}\right) \tag{2.36}$$

where M_s is the molecular weight of the species s, N_r is the number of reactions, $\alpha_{s,r}$ and $\beta_{s,r}$ are the stoichiometric coefficients for reactants and products respectively, in the r reaction. $R_{f,r}$ and $R_{b,r}$ are the forward and backward reaction rates of r reaction respectively, which are defined by

$$\begin{aligned} R_{f,r} &= 1000 \left[K_{f,r} \prod_{s=1}^{N_s} \left(0.001 \rho_s / M_s\right)^{\alpha_{s,r}} \right] \\ R_{b,r} &= 1000 \left[K_{b,r} \prod_{s=1}^{N_s} \left(0.001 \rho_s / M_s\right)^{\beta_{s,r}} \right] \end{aligned} \tag{2.37}$$

The factors 1000 and 0.001 are required for unit conversion from CGS to Metric units, since most of the reaction rate chemistry literature use CGS units.

The forward reaction rate chemistry can be expressed by

$$K_{f,r} = A_{f,r} T^{Nf,r} \exp\left(-E_{f,r} / \bar{R} T\right) \tag{2.38}$$

where $E_{f,r}$ is the activation energy of the r^{th} forward reaction. The values of parameters $A_{f,r}, N_{f,r}$ and $E_{f,r}$ are usually found in the table format according to the reactions involved. The backward reaction rate coefficient is evaluated using the equilibrium constant for the reaction such that

$$K_{b,r} = \frac{K_{f,r}}{K_{c,r}} \tag{2.39}$$

2.5 Vibrational Energy Relaxation

The energy exchange between vibrational and translational modes due to intermolecular collisions has been well described by the Landau-Teller formulation [50][51]. According to the formulation, it is reasonable to assume that the vibrational level of a molecule can change only by one quantum level at a time. The resulting energy exchange rate is given by,

$$Q_{v,s} = \rho_s \frac{e^*_{v,s}(T) - e_{v,s}}{\langle \tau_s \rangle} \tag{2.40}$$

where $e^*_{v,s}(T)$ is the vibrational energy per unit mass of species s evaluated at the local translational-rotational temperature, and $\langle \tau_s \rangle$ is the averaged Landau-Teller relaxation time of species given by

$$\langle \tau_s \rangle = \frac{\sum_{j=1}^{N_s} n_j \tau_{sj}}{\sum_{j=1}^{N_s} n_j} \tag{2.41}$$

where τ_{sj} is the vibrational-translational relaxation time of species s caused by intermolecular collision with species j, and n_j is number density of species$j$.

The Landau-Teller inter-species relaxation time τ_{sj} can be obtained in seconds using semi-empirical expression developed by Millikan and White [52] such that

$$\tau_{sj} = \frac{1}{p} \exp\left[A_{sj}\left(T^{-1/3} - 0.015\mu_{sj}^{1/4}\right) - 18.42\right] \tag{2.42}$$

where

$$\left[\begin{array}{rl} p & - \text{ Pressure in atm} \\ A_{sj} & = 1.16 \times 10^{-3} \mu_{sj} \theta_{sj} \\ \mu_{sj} & = M_s M_j / \left(M_s + M_j\right) ; \text{ reduced mass} \\ \theta_{s,j} & - \text{ Characteristic Vib. Temperature} \\ & \text{of Harmonic Oscillator} \end{array}\right.$$

The vibrational energy relaxation rate can be simplified using the following approximation

$$\sum_{\mathrm{s}} \rho_{\mathrm{s}} \frac{e^*_{v,s} - e_{v,s}}{\langle \tau_s \rangle} \approx \sum_s \rho_s C^s_{v,v} \frac{T - T_v}{\langle \tau_s \rangle} \approx \frac{\rho C_{v,v}}{\tau v} \left(T - T_v\right) \tag{2.43}$$

where

$$\frac{1}{\tau_v} = \frac{\sum\limits_s \rho_s / \left(M_s \langle \tau_s \rangle\right)}{\sum\limits_s \rho_s / M_s}$$

This approximation not only reduces the number of species dependent parameters but also simplifies the evaluation of the vibrational relaxation term multiplied by the difference in the translational and vibrational temperature. So, when a point implicit formulation is used on the source terms in the numerical algorithm, the above approximation greatly simplifies an implicit treatment of the temperature difference which drives the relaxation process.

CHAPTER 3

NUMERICAL FORMULATION

3.1 Introduction

The numerical methods employed to solve the governing differential equations are derived and discussed in this chapter. The goal is to develop parallel numerical algorithms to obtain a time-accurate solution of the thermo-chemical non-equilibrium flow fields.

The first step to a numerical simulation is discretization, which reduces the given partial differential equations into a set of algebraic equations, thus making the equations easier to solve. It is vital to maintain the conservative properties exactly for any mesh size over an arbitrary finite region containing any number of grid points [47]. Finite volume methods having conservative property are employed in this work.

The next step is to decide on an algorithm for time marching. In this regard, a major difficulty often encountered in numerical simulations of thermo-chemical non-equilibrium flow is stiffness, which is caused by vast orders of magnitudes of time scales of the different physical and chemical processes. Usually, an implicit integration in time is the method of choice in bypassing this problem. However, implicit schemes do not give good solutions for unsteady flow fields where a time-accurate solution is needed. This is because an implicit scheme loses overall efficiency it has from its unconditional stability and consequent increase in time step, when we actually need a smaller time step. This also means that a huge linear problem needs to be solved at every time step small enough to ensure a time-accurate solution. Implementation of an implicit scheme to solve non-equilibrium flows creates another problem in the derivation of the Jacobian. When

Roe flux-difference splitting schemes are used for cell face fluxes, the Jacobian becomes complicated.

Explicit time integration schemes, on the other hand, causes extreme inefficiencies in obtaining a time-accurate solution. For reasons of stability and accuracy, the integration time-step should be much smaller than the characteristic times associated with chemical reactions and thermal relaxation, which would become impractical in many cases. Hence, a point implicit scheme wherein the source terms are treated implicitly and the fluxes remain explicit is chosen along with a 2-step Runge-Kutta method as a time integration procedure. This would be beneficial as it retains both the advantages of implicit and explicit schemes, such as rescaling the characteristic times, simple and efficient nature of explicit scheme and avoiding derivation of complicated flux Jacobian for the flux difference splitting scheme.

Roe's flux-difference splitting scheme extended to nonequilibrium flow is implemented for cell interface fluxes. The implemented scheme is first order accurate, and a higher approximation is obtained by MUSCL (Monotone Upstream-centered Schemes for Conservation Laws) approach for added spatial accuracy. A MINMOD limiter would be applied to limit the slope of the variables used in the extrapolation [42].

3.2 Finite Volume Formulation

A discretized set of equations is derived in this section from the governing partial differential equations by Finite Volume Method. The advantage of this method over finite difference methods is the conservation property, as it uses the integral form of the equations and allows correct treatment of discontinuities [47]. In the following derivation, the cell-centered approach will be described.

For an arbitrary volume ω enclosed by a boundary σ, the governing equations in integral form can be written as

$$\frac{\partial}{\partial t}\int\int\int U d\Omega + \oint_{\sigma} \vec{H}.\vec{n} d\sigma = \int\int\int S d\Omega \tag{3.1}$$

where

$$\vec{H} = (F, G) \tag{3.2}$$

where U is the vector of conserved variables, S is the vector of source terms and H is the flux vector respectively. The unit vector n is normal to the infinitesimal area $d\sigma$ and points outwards. The first step is to discretize the above equation is to introduce volume averaged values of the conserved variables and the source term as follows:

$$\langle U \rangle = \frac{1}{\Omega}\int\int\int U d\Omega, \ \langle S \rangle = \frac{1}{\Omega}\int\int\int S d\Omega \tag{3.3}$$

These volume averaged variables are substituted into the integral form of the governing equations to yield

$$\frac{\partial}{\partial t}\left(\langle U \rangle \Omega\right) + \oint_{\sigma} \left(F, G\right).\vec{n} d\sigma = \langle S \rangle \Omega \tag{3.4}$$

For a two dimensional Cartesian coordinate system where the computational cell is defined by two constant lines in both the x and y directions, the surface integral can be split into four contributions, one from each bounding surface. When the index of the cell centered variables is (i, j), the above surface integral can be written as

$$\oint_{\sigma} \left(F, G\right).\vec{n} d\sigma = \int\int_{\sigma_{i+1/2}} F d\sigma + \int\int_{\sigma_{i-1/2}} F d\sigma + \int\int_{\sigma_{j+1/2}} G d\sigma + \int\int_{\sigma_{j-1/2}} G d\sigma \tag{3.5}$$

Then, area-averaged values of fluxes can be defined such that

$$\langle F \rangle_{i+1/2} = \frac{1}{\sigma_{i+1/2}}\int\int_{\sigma_{i+1/2}} F d\sigma, \ \langle G \rangle_{j+1/2} = \frac{1}{\sigma_{j+1/2}}\int\int_{\sigma_{j+1/2}} G d\sigma \tag{3.6}$$

where the bounding surface area $\sigma_{i+1/2}, \sigma_{j+1/2}$ actually represents cell face lengths in the two-dimensional Cartesian coordinate system. After substituting these definitions of

averaging into equation 3.4, the following discrete forms of the conservation equations written in two-dimensional Cartesian coordinate system can be obtained:

$$\begin{aligned}\frac{\partial}{\partial t}\left(\langle U\rangle_{i,j}\,\Omega\right) + \langle F\rangle_{i+1/2}\,\sigma_{i+1/2} - \langle F\rangle_{i-1/2}\,\sigma_{i-1/2} \\ + \langle G\rangle_{j+1/2}\,\sigma_{j+1/2} - \langle G\rangle_{j-1/2}\,\sigma_{j-1/2} = \langle S\rangle_{i,j}\,\Omega\end{aligned} \tag{3.7}$$

After dropping the brackets for simplicity, the equation is rearranged to give the final form of equations as follows:

$$\begin{aligned}\frac{\partial U_{i,j}}{\partial t} = &-\left(F_{i+1/2}\frac{\sigma_{i+1/2}}{\Omega_{i,j}} - F_{i-1/2}\frac{\sigma_{i-1/2}}{\Omega_{i,j}}\right) \\ &-\left(G_{j+1/2}\frac{\sigma_{j+1/2}}{\Omega_{i,j}} - G_{j-1/2}\frac{\sigma_{j-1/2}}{\Omega_{i,j}}\right) + S_{i,j}\end{aligned} \tag{3.8}$$

A discrete form of the conservation equations for axisymmetric geometry can be obtained similarly:

$$\begin{aligned}\frac{\partial U_{i,j}}{\partial t} = &-\left(F_{i+1/2}\frac{\sigma_{i+1/2}}{\Omega_{i,j}} - F_{i-1/2}\frac{\sigma_{i-1/2}}{\Omega_{i,j}}\right) \\ &-\frac{1}{y_j}\left(y_{j+1/2}G_{j+1/2}\frac{\sigma_{j+1/2}}{\Omega_{i,j}} - y_{j-1/2}G_{j-1/2}\frac{\sigma_{j-1/2}}{\Omega_{i,j}}\right) + S_{i,j}\end{aligned} \tag{3.9}$$

3.3 Point Implicit Time Integration

As stated above, nonequilibrium flows involving finite-rate chemistry and thermal energy relaxation often can be very difficult to solve numerically because of stiffness of the problem due to varying orders of characteristic timescales. The stiffness in terms of timescale can be defined as the ratio of the largest to the ratio of the smallest such that

$$Stiffness = \tau_{largest}/\tau_{smallest}$$

where τ can be any characteristic time in the flow field. For reactive flow problems, there can be several chemical time scales and relaxation time scales in addition to the fluid dynamic time scale associated with convection. The stiffness parameter can be high as order 10^6. The point implicit formulation evaluating the source terms at time level $n+1$ has been an effective method used to numerically integrate stiff systems [53]. The point

implicit treatment will reduce stiffness by effectively rescaling all characteristic times in the flow fields into the same order of the magnitude.

Equation 3.8 is written again with source terms evaluated at time level n+1 as follows:

$$\frac{\partial U_{i,j}}{\partial t} = -\left(F^n_{i+1/2}\frac{\sigma_{i+1/2}}{\Omega_{i,j}} - F^n_{i-1/2}\frac{\sigma_{i-1/2}}{\Omega_{i,j}}\right) - \left(G^n_{j+1/2}\frac{\sigma_{j+1/2}}{\Omega_{i,j}} - G^n_{j-1/2}\frac{\sigma_{j-1/2}}{\Omega_{i,j}}\right) + S^{n+1}_{i,j} \tag{3.10}$$

The source vector S is then linearized about the present time level such that

$$S^{n+1} = S^n + \left(\frac{\partial S}{\partial U}\right)^n \Delta U \tag{3.11}$$

When simple Euler time integration is used, substituting the above linearization into Eqn. 3.10 and rearranging it results to give

$$\left[\frac{I}{\Delta t} - \left(\frac{\partial S}{\partial U}\right)^n\right]\Delta U = -\left(F^n_{i+1/2}\frac{\sigma_{i+1/2}}{\Omega_{i,j}} - F^n_{i-1/2}\frac{\sigma_{i-1/2}}{\Omega_{i,j}}\right) - \left(G^n_{j+1/2}\frac{\sigma_{j+1/2}}{\Omega_{i,j}} - G^n_{j-1/2}\frac{\sigma_{j-1/2}}{\Omega_{i,j}}\right) + S^n_{i,j} \tag{3.12}$$

Similarly the axisymmetric version of the equation can be written as follows

$$\left[\frac{I}{\Delta t} - \left(\frac{\partial S}{\partial U}\right)^n\right]\Delta U = -\left(F^n_{i+1/2}\frac{\sigma_{i+1/2}}{\Omega_{i,j}} - F^n_{i-1/2}\frac{\sigma_{i-1/2}}{\Omega_{i,j}}\right) - \frac{1}{y_j}\left(y_{j+1/2}G^n_{j+1/2}\frac{\sigma_{j+1/2}}{\Omega_{i,j}} - y_{j-1/2}G^n_{j-1/2}\frac{\sigma_{j-1/2}}{\Omega_{i,j}}\right) + \tag{3.13}$$

These equations can be evaluated to get ΔU entirely at the current time level at the expense of matrix inversion containing the source term Jacobian.

Temporal accuracy can be added by using Runge-Kutta integration schemes instead of first order accurate Euler integration. The two-step explicit Runge-Kutta tie integration schemes can be written as follows

$$\begin{aligned} U^{n+1/2} &= U^n + \gamma_1 \Delta U^n \\ U^{n+1} &= U^{n+1/2} + \gamma_2 \Delta U^{n+1/2} + \zeta_2 \Delta U^n \end{aligned} \tag{3.14}$$

where $\gamma_1 = 1, \gamma_2 = 0.5$ and $\zeta_2 = -0.5$, and the three step scheme is given by

$$\begin{aligned} U^{n+1/3} &= U^n + \gamma_1 \Delta U^n \\ U^{n+2/3} &= U^{n+1/3} + \gamma_2 \Delta U^{n+1/3} + \zeta_2 \Delta U^n \\ U^{n+1} &= U^{n+2/3} + \gamma_3 \Delta U^{n+2/3} + \zeta_3 \Delta U^{n+1/3} \end{aligned} \tag{3.15}$$

where $\gamma_1 = 8/15, \gamma_2 = 5/12, \gamma_3 = 3/4\zeta_2 = -17/60, \zeta_3 = -5/12$

3.4 Flux Difference Split Algorithm

The basic aspect of the flux-difference split algorithm is to solve a local Reimann problem at the cell interface in order to determine the cell face flux. Roe's scheme was originally developed for a perfect gas [54]. An approximate Reimann problem is used with Roes scheme and this approach has been very successful. An extension of this method to a thermo-chemical non-equilibrium gas was made by Grossman and Cinnella [55], and the scheme used in this work has been adapted from their method.

The approximate Reimann solver is implemented by computing the cell face flux as a summation of the contributions from each wave component.

$$\begin{aligned} F_{i+1/2} &= \tfrac{1}{2}\left(F_R + F_L\right) - \tfrac{1}{2}\sum_{i=1}^{N_s+4} \hat{\alpha}_i \left|\hat{\lambda}_i\right| \hat{E}_i \\ &= \tfrac{1}{2}\left(F_R + F_L\right) - \tfrac{1}{2}\left(\llbracket F \rrbracket_A + \llbracket F \rrbracket_B + \llbracket F \rrbracket_C\right) \end{aligned} \tag{3.16}$$

where subscript R and L represent right and left state respectively, λ_i are eigenvalues, E_i are eigenvectors,α_i are corresponding wave strengths and $\wedge$ indicates Roe averaged

quantity. The term $[\![F]\!]_A$ corresponding to the repeated eigenvalues $\lambda_i = u$ can be written as

$$[\![F]\!]_A = \left([\![\rho]\!] - \frac{[\![p]\!]}{\hat{a}^2} \right) |\hat{u}| \begin{bmatrix} \hat{\rho}_i \\ \hat{u} \\ \hat{v} \\ \hat{e}_v \\ \hat{H} - \hat{a}^2/(\hat{\gamma}-1) \end{bmatrix} + \hat{\rho}\,|\hat{u}| \begin{bmatrix} [\![\rho_i/\rho]\!] \\ 0 \\ [\![v]\!] \\ [\![e_v]\!] \\ \theta \end{bmatrix} \tag{3.17}$$

where

$$\begin{aligned} &[\![(.)]\!] = (.)_R - (.)_L \\ &\theta = [\![e_v]\!] - \sum_{i=1}^{N_s} \hat{\Psi}_i\,[\![\rho_i/\rho]\!] + \hat{v}\,[\![v]\!] \\ &\hat{\Psi}_i = \tfrac{1}{\tilde{\gamma}-1}\tfrac{\partial p}{\partial \rho_i} = \tfrac{R_i T}{\tilde{\gamma}-1} - e_{eq,i} + \tfrac{u^2+v^2}{2} \\ &\tilde{\gamma} = \tfrac{\tilde{C}_p}{\tilde{C}_v} = \tfrac{C_{p,t}+C_{p,r}}{C_{v,t}+C_{v,r}} \\ &R = \bar{R}/M_s \end{aligned}$$

The $[\![F]\!]_B$ and $[\![F]\!]_C$ terms which are contributions from the eigenvalues $\lambda_i = u \pm a$, are found to be

$$[\![F]\!]_{B,C} = \frac{1}{2\hat{a}^2}\left([\![p]\!] \pm \hat{\rho}\hat{a}\,[\![u]\!]\right)(\hat{u} \pm \hat{a}) \begin{bmatrix} \hat{\rho}_i \\ \hat{u} \pm \hat{a} \\ \hat{v} \\ \hat{e}_v \\ \hat{H} \pm \hat{u}\hat{a} \end{bmatrix} \tag{3.18}$$

When equations 3.17 and 3.18 are inserted into equation 3.16 to get the cell face flux, the absolute values of the wave speeds should be substituted in equations 3.17 and 3.18

Similarly, the cell-face flux in y-direction can be written as

$$G_{j+1/2} = \frac{1}{2}(G_U + G_L) - \frac{1}{2}([\![G]\!]_A + [\![G]\!]_B + [\![G]\!]_C) \tag{3.19}$$

and $[\![G]\!]_A$, $[\![G]\!]_B$ and $[\![G]\!]_C$ terms can be found similarly as in the x-direction.

Following the derivation of Grossman and Cinnella, the Roe-averaged quantities are determined to be

$$
\begin{aligned}
& r \equiv \sqrt{\rho_R/\rho_L} \\
& \hat{\rho} = \sqrt{\rho_R/\rho_L} \\
& \hat{u} = \tfrac{u_L + r u_R}{1+r} \\
& \hat{v} = \tfrac{v_L + r v_R}{1+r} \\
& \hat{\rho}_i = \tfrac{(\rho_i)_R + (\rho_i)_L}{(1+r)}, \quad i = 1, 2, ..., N_s \\
& \hat{e}_v = \tfrac{(e_v)_L + (e_v)_R}{1+r} \\
& \hat{H} = \tfrac{H_L + r H_R}{1+r} \\
& \hat{\Psi} = \tfrac{R_i \hat{T}}{\hat{\gamma}-1} - \hat{e}_i + \tfrac{u^2+v^2}{2}, \quad i = 1, 2, ..., N_s \\
& \hat{a}^2 = (\hat{\gamma}-1)\left(\hat{H} - \tfrac{u^2+v^2}{2} + \hat{C}_v^* \hat{T} - \sum_{i=1}^{N_s} \hat{\rho}_i \hat{e}_i - \hat{e}_v \right) \\
& \hat{T} = \tfrac{T_L + r T_R}{1+r} \\
& \hat{C}_v^* = \sum_{i=1}^{N_s} \hat{\rho}_i \hat{C}_{v,i}^* \\
& \hat{C}_{v,i}^* = \tfrac{1}{[\![T]\!]} \int_{T_L}^{T_R} \tilde{C}_v^i dT = C_{v,t}^i + C_{v,r}^i \\
& \hat{e}_i = \tfrac{(e_{eq,i})_L + (e_{eq,i})_R}{1+r} \\
& \hat{\gamma} = 1 + \tfrac{\hat{R}}{\hat{C}_v^*} \\
& \hat{R} = \tfrac{(\tilde{R})_L + (\tilde{R})_L}{1+r}, \text{where} \tilde{R} = \sum_{i=1}^{N_s} \tfrac{\rho_i}{\rho} R_i
\end{aligned}
\tag{3.20}
$$

For added spatial accuracy, higher order approximation using MUSCL approach can be applied. When the MUSCL approach is employed, the primitive variables of right and left states at the cell interface are evaluated using the following extrapolation formulae[54]

$$
\begin{aligned}
& q_{i+1/2}^L = q_i + \tfrac{1-\kappa}{4} \bar{\delta}^+ q_{j-1/2} + \tfrac{1+\kappa}{4} \bar{\delta}^- q_{j+1/2} \\
& q_{i+1/2}^R = q_{i+1} - \tfrac{1+\kappa}{4} \bar{\delta}^+ q_{j+1/2} - \tfrac{1-\kappa}{4} \bar{\delta}^- q_{j+3/2}
\end{aligned}
\tag{3.21}
$$

where q is any primitive variable, superscript L and R represents left and right extrapolation respectively, and the value of κ determines the type of extrapolation method such that

$$\kappa = \begin{cases} -1 & \text{2nd order upwind scheme} \\ 1/3 & \text{3rd order upwind scheme} \\ 1 & \text{2nd order classic centered scheme} \end{cases} \tag{3.22}$$

In the above equations, the slopes of the variables are limited to prevent nonphysical oscillations and to preserve TVD (Total Variation Diminishing) property. The limited slopes can be written using minmod limiter as follows:

$$\begin{aligned} \bar{\delta}^{-} q_{i+1/2} &= \text{min mod}\left(\delta q_{i+1/2}, \omega \delta q_{i-1/2}\right) \\ \bar{\delta}^{+} q_{i+1/2} &= \text{min mod}\left(\delta q_{i+1/2}, \omega \delta q_{i+3/2}\right) \end{aligned} \tag{3.23}$$

The minmod limiter is a function that selects the smallest number from a set when all have the same sign but is zero when they have different signs such that

$$\text{min mod}\,(x, y) = \begin{cases} x & \text{if } |x| < |y| \text{ and} xy > 0 \\ y & \text{if } |x| > |y| \text{ and} xy > 0 \\ 0 & xy > 0 \end{cases} \tag{3.24}$$

with the limits on ω given as

$$1 \leq \omega \leq \frac{3-\kappa}{1-\kappa} \tag{3.25}$$

3.5 Source Terms and Jacobian

The Jacobian of the source terms needs to be developed. This arises from the point implicit treatment of source terms. The vector of conserved variables U and the vector of source terms S for axisymmetric flow are given below

$$U = \begin{bmatrix} \rho_s \\ \rho u \\ \rho v \\ \rho e_v \\ \rho E \end{bmatrix}, \quad S = \begin{bmatrix} w_s \\ 0 \\ p/y \\ w_v \\ 0 \end{bmatrix} \tag{3.26}$$

where

$$w_s = M_s \sum_{r=1}^{N_s} (\beta_{s,r} - \alpha_{s,r}) (R_{f,r} - R_{b,r})$$
$$w_v = \frac{\rho C_{v,v}}{\tau_v} (T - T_v) + \sum_s w_s e_{d,s}$$

The second term in w_v is the average vibrational energy created or destroyed according to the mass production rate. If we suppose a molecule in a higher vibrational state is more likely to react and be depleted, and molecules produced are more likely to be in a higher vibrational state, then $e_{d,s}$ should be larger than the average vibrational energy $e_{v,s}$ of the system. At the same time, it must be smaller than the dissociation energy of the molecule [56]. One of Park's models [57] for preferential reaction is used here such that

$$e_{d,s} = 0.3\tilde{D}_s, \quad \tilde{D}_s = \text{Dissociationenergyofmolecules} \tag{3.27}$$

Since the term w_s depends explicitly upon the species density and temperature, the Jacobian of w_s with respect to U is evaluated using the chain rule such that

$$\frac{\partial w_s}{\partial U_j} = \left.\frac{\partial w_s}{\partial U_j}\right|_T + \left.\frac{\partial w_s}{\partial T}\right|_U \frac{\partial T}{\partial U_j} \tag{3.28}$$

Partial derivatives of w_s can be obtained directly from the partial derivatives of the reaction rates

$$\begin{aligned}
\left.\frac{\partial R_{f,r}}{\partial \rho_s}\right|_T &= \frac{\alpha_{s,r}}{\rho_s} R_{f,r} \\
\left.\frac{\partial R_{b,r}}{\partial \rho_s}\right|_T &= \frac{\beta_{s,r}}{\rho_s} R_{b,r} \\
\left.\frac{\partial R_{f,r}}{\partial T}\right|_U &= \left(N_{f,r} + \frac{E_{f,r}}{RT}\right) \frac{R_{f,r}}{T} \\
\left.\frac{\partial R_{b,r}}{\partial T}\right|_U &= \left((N_{f,r} - N_{c,r}) + \frac{\left(E_{f,r} - E_{c,r}\right)}{RT}\right) \frac{R_{b,r}}{T}
\end{aligned} \tag{3.29}$$

Equation 3.28 is made complete by defining the partial derivatives of temperature which are found to be

$$\rho C_{v,tr} \frac{\partial T}{\partial U_j} = \begin{cases} \frac{u^2+v^2}{2} - e_{eq,j} & , \ j = 1, ..., N_s \\ -u & , \ j = N_s + 1 \\ -v & , \ j = N_s + 2 \\ -1 & , \ j = N_s + 3 \\ 1 & , \ j = N_s + 4 \end{cases} \tag{3.30}$$

where

$$C_{v,tr} = C_{c,t} + C_{v,r}$$

The partial derivatives of the vibrational energy production rate can be written as

$$\frac{\partial w_v}{\partial U_j} = \frac{\rho C_{v,v}}{\tau_v} \left(\frac{\partial T}{\partial U_j} - \frac{\partial T_v}{\partial U_j}\right) + \sum_s \frac{\partial w_s}{\partial U_j} e_{d,s} \tag{3.31}$$

All of the terms in this equation are described if the partial derivatives of the vibrational temperature are additionally defined to be[60]

$$\rho C_{v,tr} \frac{\partial T}{\partial U_j} = \begin{cases} -e_{v,j} & , \ j = 1, ..., N_s \\ 0 & , \ j = N_s + 1 \\ 0 & , \ j = N_s + 2 \\ 1 & , \ j = N_s + 3 \\ 0 & , \ j = N_s + 4 \end{cases} \tag{3.32}$$

The last term to be considered is due to the axisymmetric formulation of the equation set, and is written as

$$\frac{\partial\left(p/y\right)}{\partial U_j} = \frac{1}{y}\frac{\partial p}{\partial U_j} \tag{3.33}$$

where the pressure derivatives are found to be

$$\frac{1}{\tilde{\gamma}-1}\frac{\partial p}{\partial U_j} = \begin{cases} \xi_j & , \ j = 1, ..., N_s \\ u & , \ j = N_s + 1 \\ v & , \ j = N_s + 2 \\ -1 & , \ j = N_s + 3 \\ 1 & , \ j = N_s + 4 \end{cases} \tag{3.34}$$

and

$$\xi_j = \frac{R_j T}{\tilde{\gamma}-1} - e_{eq,j} + \frac{u^2+v^2}{2}$$

3.6 Calculation of Temperature

The conserved variables at each cell center are updated by a matrix inversion scheme. From these conserved variables, new values of the primitive variables, ρ_s, u, v, e_v, E are easily obtained. Finally, the temperatures have to be determined at each iteration cycle. In order to obtain the temperatures, a Newton-Raphson method is used in the following manner[58][59]

$$\begin{aligned} T^{k+1} &= T^k + \frac{\rho e - \sum\limits_s \rho_s e_s\left(T^k, T_v^k\right)}{\rho C_{v,tr}} \\ T_v^{k+1} &= T_v^k + \frac{\rho e_v - \sum\limits_s \rho_s e_{v,s}\left(T^k\right)}{\rho C_{v,v}} \end{aligned} \tag{3.35}$$

From the updated conservative variables, e and e_v are obtained directly and species internal energies e_s and vibrational energies $e_{v,s}$ are calculated from the gas model using the current values of both temperatures. The iteration proceeds till the converged values of both temperatures are obtained.

CHAPTER 4

PARALLEL ALGORTIHM FORMULATION

4.1 Introduction

Parallel computing is a process by which a task is simultaneously executed on multiple processors in order to obtain faster results. The term parallel processor is sometimes used for a computer with more than one processor, available for parallel processing. The systems consisting of a large number of such processors are known as massively parallel [60].

Parallel computers can be classified according to the following criteria:

- The type of interconnection and communication
  - Between the processors
  - Between processors and memories
- The nature of instruction execution in processors
  - SIMD (Single Instruction/Multiple Data)-execution of same instructions at the same instant
  - MIMD (Multiple Instruction/Multiple Data)-execution of different instructions at the same instant
- The runtime of Operating System code
  - Symmetric - all the processors are capable of same runtime environments and I/O device accessibility
  - Asymmetric - Privileges are assigned to unique processors for runtime and/or I/O device accessibility

While a system of n parallel processors is less efficient than a single n-times-faster processor, the parallel system is often cheaper to build. Parallel computation is an excellent solution for tasks which require very large amounts of computation, have time constraints on completion and particularly for those which can be divided into$n$ number of execution threads. Processors in a parallel computer may communicate with each other in a number of ways, including shared memory (either multiported or multiplexed), a crossbar, a shared bus or an interconnect network of a myriad of topologies including star, ring, tree, hypercube, fat hypercube (an hypercube with more than one processor at a node), a n-dimensional mesh and so on. Parallel computers based on interconnect network need to employ a kind of routing that will enable passing of messages between nodes that are not directly connected. The communication medium used for communication between the processors is likely to be hierarchical in large multiprocessor machines. Similarly, memory may be available either privately to the processor, shared between a number of processors, or globally shared.

Approaches to the design of parallel computers include:

- Multiprocessing
- Computer cluster
- Parallel supercomputers
- Distributed computing
- Grid computing

4.2 Scalable Clusters

A computer cluster is a group of loosely coupled computers that work together closely so that in many respects it can be viewed as though it were a single computer. Clusters are commonly connected through fast local area networks. Clusters are usually deployed to improve speed and/or reliability over that provided by a single computer,

while typically being much more cost-effective than single computers of comparable speed or reliability.

4.2.1 Cluster Types

There are many ways to categorize clusters, but the following is a common categorization [61]:

- High Availability Cluster
- Load Balancing Cluster
- High Performance Cluster

High-availability (HA) Clusters are implemented primarily for the purpose of improving the availability of services which the cluster provides. They operate by having redundant nodes which are then used to provide service when system components fail. The most common size for an HA cluster is two nodes, since that's the minimum required to provide redundancy. HA cluster implementations attempt to manage the redundancy inherent in a cluster to eliminate single points of failure. There are many commercial implementations of High-Availability clusters for many operating systems. The Linux-HA project is one commonly used free software HA package for the Linux OS.

Load balancing clusters operate by having all workload come through one or more load-balancing front ends, which then distribute it to a collection of back end servers. Although they are implemented primarily for improved performance, they commonly include high-availability features as well. Such a cluster of computers is sometimes referred to as a server farm. There are many commercial load balancers available. The Linux Virtual Server project provides one commonly used free software package for the Linux OS. High-performance (HPC) clusters are implemented primarily to provide increased performance by splitting a computational task across many different nodes in the cluster, and are most commonly used in scientific computing. One of the more popular HPC

implementations is a cluster with nodes running Linux as the OS and free software to implement the parallelism. This configuration is often referred to as a "Beowulf" cluster. Such clusters commonly run custom programs which have been designed to exploit the parallelism available on HPC clusters. Many such programs use libraries such as MPI which are specially designed for writing scientific applications for HPC computers. operate by having all workload come through one or more load-balancing front ends, which then distribute it to a collection of back end servers. Although they are implemented primarily for improved performance, they commonly include high-availability features as well. Such a cluster of computers is sometimes referred to as a server farm. There are many commercial load balancers available. The Linux Virtual Server project provides one commonly used free software package for the Linux OS.

High-performance (HPC) clusters are implemented primarily to provide increased performance by splitting a computational task across many different nodes in the cluster, and are most commonly used in scientific computing. One of the more popular HPC implementations is a cluster with nodes running Linux as the OS and free software to implement the parallelism. This configuration is often referred to as a "Beowulf" cluster. Such clusters commonly run custom programs which have been designed to exploit the parallelism available on HPC clusters. Many such programs use libraries such as MPI which are specially designed for writing scientific applications for HPC computers.

The proto-cluster employed in performing the computations consists of 8 nodes connected via an ethernet switch to form a typical Beowulf cluster. The architecture is Pentium based, performing at an optimum 500 MHz, with distributed memory.The installed OS is RedHat Enterprise Linux 3, and the message passing paradigm is MPICH [62].

4.3 Parallel Programming Models

A parallel programming model is a set of software constructs to express parallel algorithms and match applications with the underlying parallel systems. It encloses the areas of applications, languages, compilers, libraries, communication systems, and parallel I/O. A proper parallel programming model has to be chosen to develop the parallel applications on a particular platform. Parallel models are implemented in several ways such as libraries invoked from traditional sequential languages, language extensions, or complete new execution models.

A few of the current main-stream parallel programming models are:

- PVM (Parallel Virtual Machine)
- MPI (Message Passing Interface)
- OpenMP
- Global Arrays
- Co-Array Fortran
- UPC (Unified Parallel C)
- HPF (High Performance Fortran)
- SHMEM (SHared MEMory)

4.3.1 Message Passing Interface (MPI)

Message passing is a programming paradigm used widely on parallel computers, especially Scaleable Parallel Computers with distributed memory, and on Networks of Workstations (Clusters) [63]. MPI has been designed to make use of most of the advantages of a number of existing message-passing systems, rather than selecting one of them and adopting it as the standard. Critical shortcomings of existing message passing systems in areas like complex data layouts or support for modularity and safe communi-

cation have been overcome, and user interface and functionality has been improved for a wide range of message passing capabilities.

The major goals of MPI are a degree of portability across different machines, smooth operation in heterogeneous systems, and to allow efficient implementations across machines of differing characteristics. The goals can be briefly summarized as:

- Design an Application Program Interface
- Allow efficient communication, with no memory-to-memory copying
- Allow for implementations in heterogeneous environments, along with convenient bindings for C and FORTRAN 77 with the semantics being language independent
- Provision of a reliable communication interface

Drawbacks of the current implementation include:

- Explicit shared memory operations
- Program construction tools
- Debugging facilities
- Explicit support for threads
- I/O functions
- Task management support

4.3.2 Semantic Terms: Types of MPI Calls

When discussing MPI procedures, the following terms are used

- Local: The completion of the procedure depends only on the local executing process, such an operation not requiring an explicit communication with another user process
- Non-local: The completion of the procedure may require the execution of some MPI procedure on another process. Many MPI communication calls are non-local

- Blocking: The return from the procedure indicates that the user is allowed to reuse resources specified in the call. Any visible change in the state of the calling process, affected by a blocking call, occurs before the call returns
- Non-blocking: The procedure may return before the operation initiated by the call completes, and before the user is allowed to reuse resources (such as buffers) specified in the call. A non-blocking call may initiate changes in the state of the calling process that actually take place after the call in returned, which means that this call can initiate a receive operation, but the message is actually received after the call is returned.
- Collective: All processes in a process group need to invoke the procedure

4.4 Point-to-Point Communication

The basic communication mechanism of MPI is the transmittal of data between a pair of processes, one side sending and the other side receiving. This operation is known as 'Point-to-Point Communication'. It is noted that almost all MPI operations are point-to-point based.

MPI provides a set of send and receive functions that allow the communication of typed data with an associated tag. Typing of the message contents is necessary for heterogeneous support - the type information is needed so that the correct data representation conversions can be performed so that data is sent from one architecture to another.

4.4.1 Blocking Send and Receive Operations

This section is devoted for the standard-mode, blocking sends and receives.

Blocking Send

$$MPI\_SEND(buf,\ count,\ datatype,\ dest,\ tag,\ comm) \tag{4.1}$$

where

- *buf* - This contains the initial address of the receive buffer
- *count* - This specifies the number of entries that need to be sent
- *datatype* - This is used to specify the datatype of each entry
- *dest* - This specifies the rank of the destination
- *tag* - This allocates memory for messages to be sent to the receiver
- *comm* - Communicator

Send Buffer - This consists of "count" successive entries of the type indicated by "datatype", starting with the entry at address "buf".

Message Envelope - In addition to data, messages carry information that is used to distinguish and selectively receive them. This information consists of a fixed number of fields, collectively known as the message envelope, which consists of

- Source
- Destination
- Tag
- Communicator

Focus is on the last member of the message envelope, the communicator, which is a local object that represents a communication domain. A communication domain is a global, distributed structure that allows process in a group to communicate with each other, or to communicate with process in another group. A communication domain of the first type (communication within a group) is known as intracommunicator whereas a communication domain of the second type (communication between groups) is represented by an intercommunicator. Processes in a group are ordered, and are identified by their

integer rank. Processes may participate in several communication domains, which are distinct and may have partially or even completely overlapping communications. Thus a process can communicate between other processes via two distinct communication domains, using two distinct communicators.

Communicators provide an important encapsulation mechanism for libraries and modules. They allow modules to have their own communication space and their own process numbering scheme. The message envelope is often encoded by a fixed length message header, which carries its own communication domain id, sometimes referred to as context id. In addition, the header will usually carry message source and tag; source can be represented as rank within group or as an absolute task id.

Blocking Receive

$$MPI\_RECV(buf,count,datatype,source,tag,comm,status) \tag{4.2}$$

where

- *buf* - This contains the address of the send buffer
- *count* - This contains the number of entries to be sent to the receiver
- *datatype* - This specifies the datatype of each entry
- *dest* - This contains the rank of the destination
- *tag* - This is the memory allocated for messages
- *comm* - Communicator
- *status* - This is to return error code, if any

MPI_RECV performs a standard-mode, blocking receive. The semantics are as follows:

- Receive Buffer - The receive buffer consists of storage sufficient to contain count consecutive entries of the type specified by datatype, starting at address buf. The length of the received message must be less than or equal to the length of the receive

buffer, otherwise an overflow error occurs if the information does not fit into the receive buffer.

- Message Selection - The selection of the message by a receive operation is governed by the value of it message envelope. A message can be received if it envelope matches the source, tag and comm values specified in the receive operation .
- Return Status - The receive call does not specify the size of an incoming message, but only an upper bound. The source or tag of the received message may not be known if wildcard values were used in the receive operation. Also, if multiple requests are completed by a single MPI function, a distinct error code may be returned for each request. This information is returned by the status argument of MPI_RECV. The type of status is defined by MPI. Status variables are not system objects, which mean that they need to be allocated by the user.

It is noted that a receive operation may accept messages from an arbitrary sender, but a send operation must specify a unique receiver. This matches a *push* communication mechanism, wherein data transfer is effected the sender, rather than a *pull* mechanism, where data transfer is effected by the receiver.

4.4.2 Fairness

It should be noted at this point that MPI makes no guarantee of fairness in the handling of communication. For example, if a send is posted, it is possible that the destination process repeatedly posts a receive that matches this send, yet the message is never received, because it is repeatedly overtaken by other messages, sent from other sources. This requires for a receive to use the wildcard MPI_ANY_SOURCE as its source argument.

Similarly, if there is a scenario wherein a receive is posted by a multi threaded process, then it is possible that messages that match this receive are repeatedly consumed,

Figure 4.1 Block Portioning of a Matrix

yet the receive is never satisfied, because it is overtaken by other receives posted at this node by other threads. It becomes the programmer's responsibility in such situations.

4.4.3 Data Blocks

A parallel algorithm is derived from the choice of data distribution. The distribution should be balanced, allocating the same number of entries to each processor, also minimizing communication. Figure 4.1 illustrates two possible distributions: a 1D (block) distribution, where the matrix is partitioned in one dimension, and a 2D (block, block) distribution, where the matrix is partitioned in two dimensions. Since the communication occurs at block boundaries, communication volume is minimized by the 2D partition which has a better area to perimeter ratio. However, in this partition, each processor communicates with four neighbors, rather than two neighbors in the 1D partition. When the ratio of n/P (P - number of processors) is small communication time will be dominated by the fixed per message, and the first partition will lead to better performance. For purposes of simplicity, this work uses 1D partition; a realistic code would actually use a *polyalgorithm* that selects on of the two partitions, according to the problem size, number of processors and communication performance parameters. For a 1D block problem, communications are needed at block boundaries in order to receive

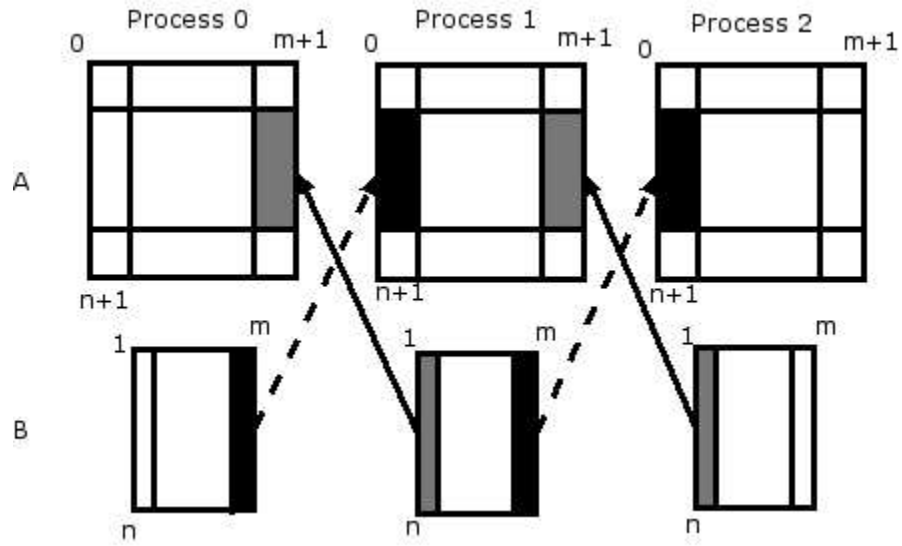

Figure 4.2 1D Block Partitioning with Overlap and Communication Pattern

values of neighbor points which are owned by another processor. Communications can be simplified if an overlap area is allocated at each processor for storing the values to be received by the neighboring processor. Essentially, storage is allocated for each entry both at the producer and at the consumer of that entry. If an entry is produced by one processor and consumed by another, then storage is allocated for entry at both processors. With such a scheme there is no need for dynamic allocation of communication buffers, and the location of each variable in the data is fixed. Such a scheme works if the data dependencies in the computation are fixed and simple. In this work, the spatial derivatives are represented by a four point stencil. Therefore, a one column overlap is needed, for a 1D partition.

Figure 4.2 shows an instance of data transfer in each iteration. The algorithm used has an advantage wherein all the values needed from a neighbor are brought in one message. Coalescing of communications in this manner reduces the number of messages and generally improves performance.

4.5 Conclusion

The discussion about MPI ends by addressing some important topics which are concerned more with integration of the MPI specification, and the rationale behind some aspects of its design rather than semantics and syntax. The discussion is started with the main drawback of MPI, buffering and proceeds on to ambiguous communication and ends with heterogeneous computing with MPI.

4.5.1 Buffering

MPI does not guarantee to buffer arbitrary messages because memory is a finite resource on all computers. This means that all computers will fail under sufficiently adverse communication loads. Different computers at different levels are capable of providing differing amounts of buffering, so if a program relies on buffering it may fail under certain conditions, but work perfectly under other conditions, which is clearly undesirable.

One of the major problems in cluster computing is that it is not obvious how to specify the amount of buffer space that is available, nor is it easy to estimate how much buffer space is consumed by the program. Now different buffering policies make sense in different environments. Messages can be buffered at the sending node or the receiving node, or in both. In the former case, buffers can be

1. Dedicated to one destination in one communication domain
2. Dedicated to one destination for all communication domains
3. Shared by all outgoing communications
4. Shared by all processes running at a processor node
5. Partly dedicated and partly shared

Similar choices occur if messages a buffered at the destination. Communication buffers may be fixed in size or may be dynamically allocated out of the heap in competition with the application. The buffer allocation policy depends on the size of the

messages (preferably buffering short messages), and on communication history (preferably buffering on busy channels).

The choice of the right policy is highly dependent on the hardware and software environment. For instance, in a dedicated environment, a processor with process blocked on a send is idle and so computing resources are not wasted if this processor copies the outgoing message to a buffer. In a time shared environment, the computing resources may be used by another process. In a system where buffer space can be in a page file memory, such space can be allocated in a heap. If a buffer space cannot be paged or has to be in a kernel space, then a separate buffer is required. Flow control may require that some of the buffer space be dedicated to each pair of communicating processes.

The optimum settings strongly depends on the following cluster parameters:

1. Bandwidth
2. Communication startup time
3. Scheduling and context switching overheads
4. Amount of potential overlap between communication and computation

The choice of a buffering and scheduling policy may not be entirely under the control of MPI implementer, as it is partially determined by the properties of the underlying communications layer. Also, experience in this arena is quite limited with the underlying technology changing rapidly: fast, user-space inter-processor communication mechanisms are an active research area.

Attempts by the MPI Forum to design mechanisms for querying or setting the amount of buffer space available to standard communication led to the conclusion that such mechanisms will either restrict allowed implementations unacceptably, or provide bounds that will be extremely pessimistic on most implementations in most cases. Another problem is that parameters such as buffer sizes work against portability. Rather than restricting the implementation strategies for standard communication, the choice

was taken to provide additional communication modes for those users that do not want the implementation to make the right choice for them.

4.5.2 Ambiguous Communications and Portability

MPI employs communicator abstraction to promote software modularity by allowing the construction of independent communication streams between processes, thereby ensuring that messages sent in one phase of an application are not incorrectly intercepted by another phase. Communicators are particularly important in allowing libraries that make message passing calls to be used safely within an application. Communicators, in effect, provide an additional criterion for message selection and hence permit construction of independent tag spaces.

4.5.3 Heterogeneous Computing with MPI

Heterogeneous computing uses different computer connected by a network to solve a problem in parallel. With heterogeneous computing a number of issues arise that are not applicable when using a homogeneous parallel computer. For example, as in the proto-cluster designed for this application, computers may differ in computing power, so care must be taken to distribute the work between them to avoid a load imbalance. Other problems may arise because of the different behavior of floating point arithmetic on different machines. However, the two most fundamental issues that must be faced in heterogeneous are

1. Incompatible data representation
2. Interoperability of differing implementations of the message passing layer

Incompatible data representations arise when computers use binary representations for the same number. In MPI, all communication routines have a datatype argument so

implementations can use this information to perform appropriate representation conversion when communicating data between computers.

Interoperability refers to the ability of different implementations of a given piece of software to work together as if they were a single homogeneous implementation. A prerequisite for interoperability for MPI would be the standardization of MPI's internal data structures, communication protocols, initialization-termination-error handling procedures and implementation of collective operations. Since this has been accomplished in MPI-2 and not MPI, interoperability is not supported in MPI. In general, hardware specific implementations of MPI will not be interoperable. However, it is still possible for different architectures to work together if they both use the same portable MPI implementation.

4.5.4 MPI implementations

There are several portable implementations of MPI currently available, some of which are as follows

- MPICH implementation from Argonne National Laboratory and Mississippi State University, which was implemented in the course of this work
- CHIMP implementation from Edinburgh Parallel Computing Center
- Lam implementation from Ohio Supercomputing Center
- UNIFY system provides a subset of MPI within the PVM environment

In addition, hardware specific MPI implementations exist for the Cray T3D, the IBM SP-2, NEC Cinju, and the Fujitsu AP1000. Information on MPI implementations and other useful information on MPI can be found on the MPI web pages of Argonne National Laboratory [64].

CHAPTER 5

RESULTS AND VERIFICATION

5.1 Hydrogen-Air Model

The major application of this work was to reduce the computation time while maintaining accuracy in the simulation of the detonation waves caused by the detonation of a mixture of Hydrogen and Air. The scope of this work includes the construction of an efficient parallel algorithm, so that introduction of detailed chemical reaction mechanisms which can cause excessive computation and storage requirement would be performed to ensure a detailed simulation efficient enough to be used in parametric studies, as well as accurate enough to be used in analysis and design. A simple two-step reaction model proposed by Rogers and Chinitz [65] is used for the purpose of validating the parallel algorithm by comparison with the detailed study hydrogen-air simulation performed earlier.

The two-step reaction model was developed to represent Hydrogen-Air chemical kinetics with as few reaction steps as possible. This model consists of the two steps as follows:

$$\begin{aligned} H_2 + O_2 &\leftrightarrow 2OH \\ 2OH + H_2 &\leftrightarrow 2H_2O \end{aligned} \tag{5.1}$$

This model is valid for temperatures of 1000-2000 K, and requires a provision of a hot spot as the chemistry model is not valid for temperatures below 1000 K. It is noted that Nitrogen is counted as a collisional partner in the thermodynamic model, but not included in the chemical model as the dissociation temperature of Nitrogen is not reached.

An interesting aspect of this simulation was that all the major changes in species concentration take place in the first 10^{-7} seconds, typical of a fluid dynamic time step. The integration time step should be in the order of 10^{-12} seconds to properly follow chemical kinetics. From this observation, it becomes evident that the using 10^{-12} seconds as an integration step for the flow solver would result in the requirement of 10^{9} to generate a timescale of interest of the order of 10^{-3} seconds. This means a computation time requirement of 10^{4} days of CPU time assuming 1 second of CPU time per computation cycle. This problem can be overcome by rescaling the chemical reaction time scale. But the very first time step where all the drastic changes take place in chemical kinetics cannot be obtained by rescaling the time step alone. This was overcome by a method known as Local Ignition Averaging Model proposed by Dr. Kim et al. The basic idea of this model comes from the fact that species mass fractions change rapidly on ignition and comes back to equilibrium soon afterwards. Hence the cell where the condition of hot spot is applied, chemical reactions are solved with a time step of 10^{12} seconds. Though computation time was significantly reduced, it still took 117 hours of computation to attain convergence. Higher step reactions and complex reaction mechanisms would inevitably become highly computation intensive. Hence, the effort of this work was to execute the simulation in a parallel environment to reduce computation time whereas retain the levels of accuracy.

5.2 Verification of Results

The verification of the results obtained is compared with the results achieved in the earlier work. A typical calculated result showing the formation and propagation of a detonation wave is presented in this section, along with results from the work done previously to assure that the results obtained are the same and correct behavior of detonation wave is predicted. Figures 5.1-5.8 show a detonation wave traveling into the fuel air mixture initially at 1 atm pressure and 298.15 K temperature. Ignition has been initiated

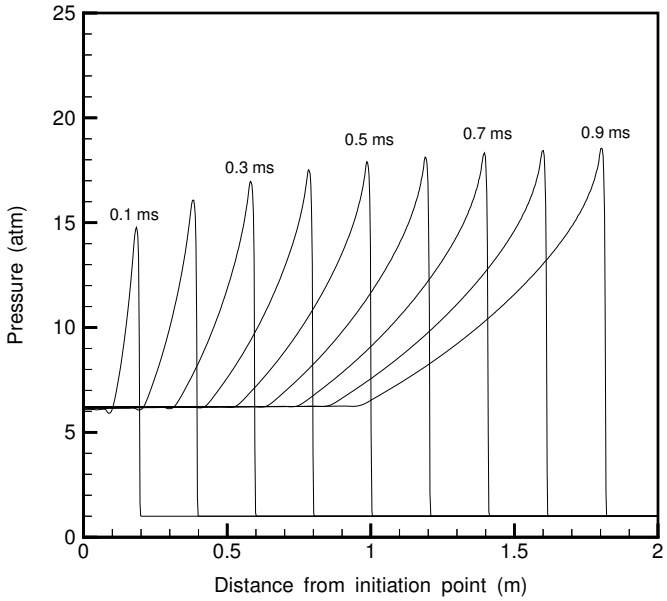

Figure 5.1 Time Evolution of Detonation Pressure Profiles

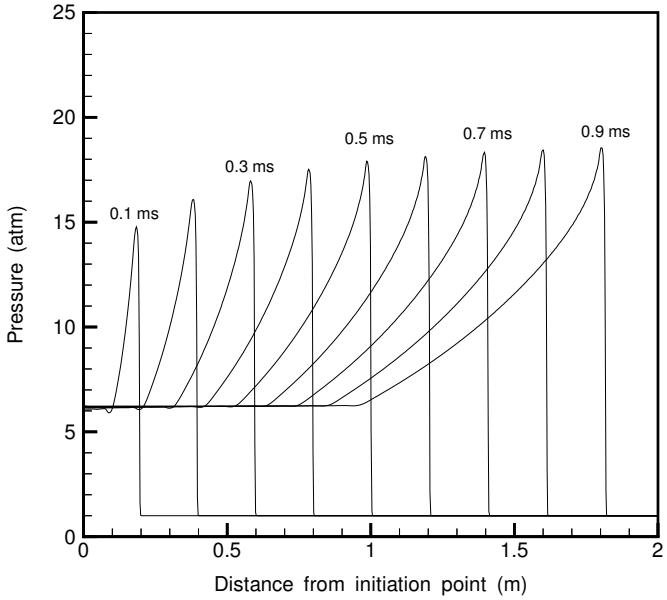

Figure 5.2 Time Evolution of Detonation Pressure Profiles - Dr. Kim et al

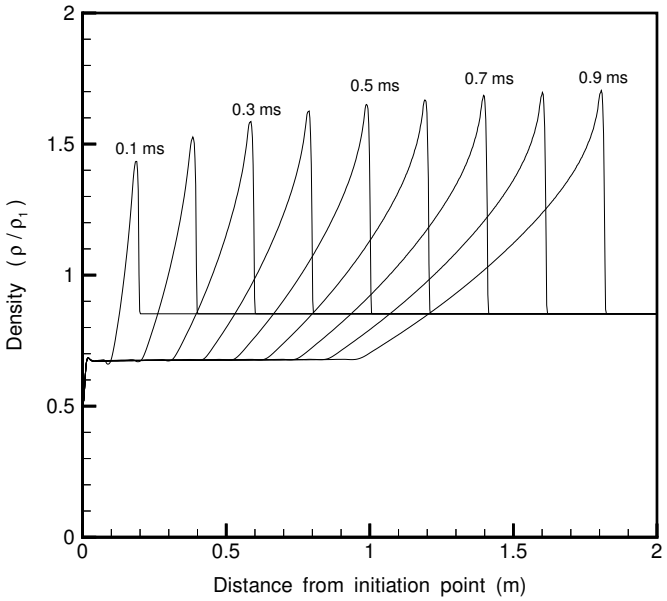

Figure 5.3 Time Evolution of Detonation Density Profiles

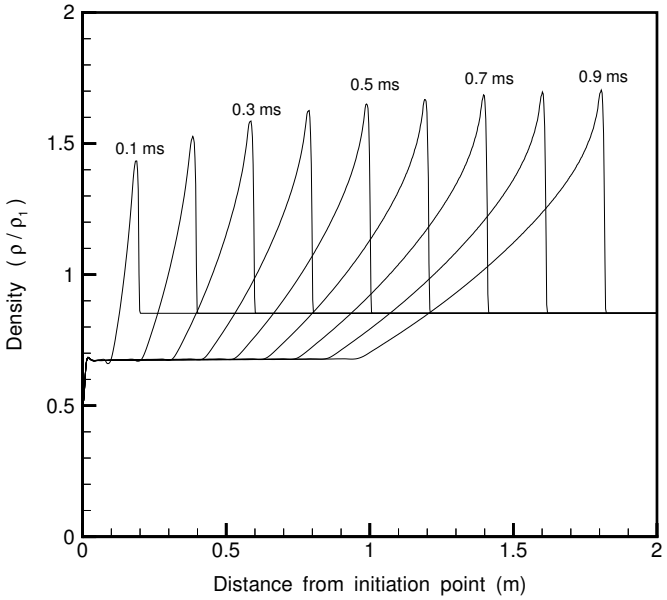

Figure 5.4 Time Evolution of Detonation Density Profiles - Dr. Kim et al

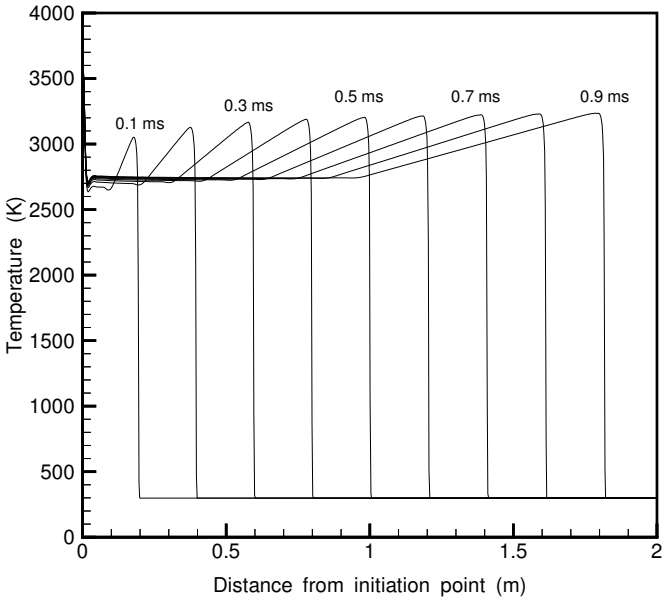

Figure 5.5 Time Evolution of Detonation Temperature Profiles

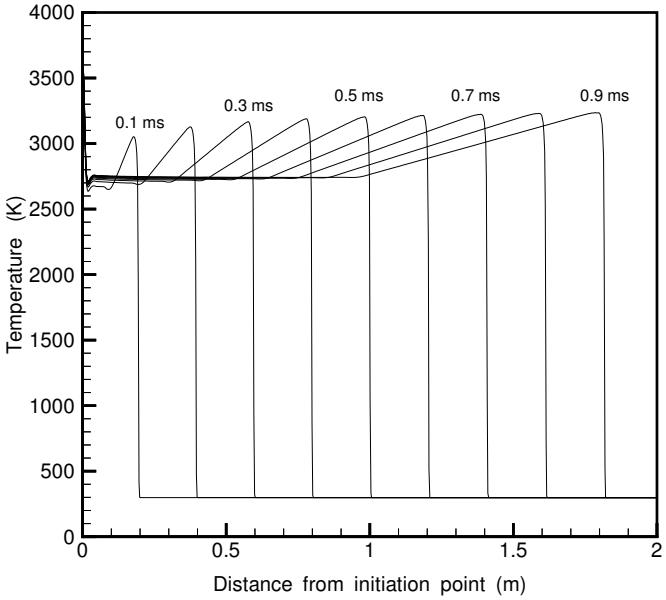

Figure 5.6 Time Evolution of Detonation Temperature Profiles - Dr. Kim et al

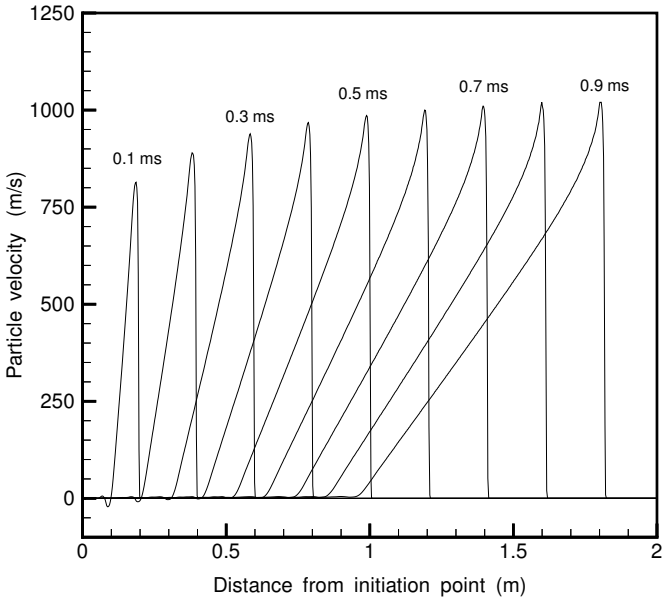

Figure 5.7 Time Evolution of Detonation Particle Velocity Profiles

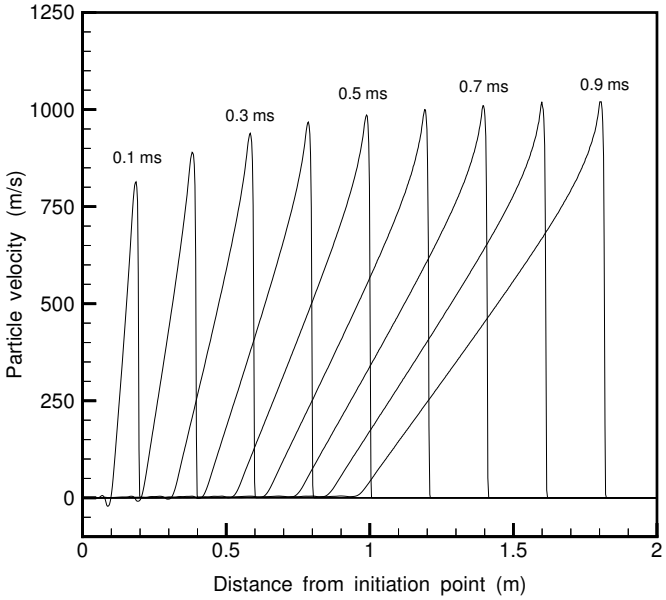

Figure 5.8 Time Evolution of Detonation Particle Velocity Profiles - Dr. Kim et al

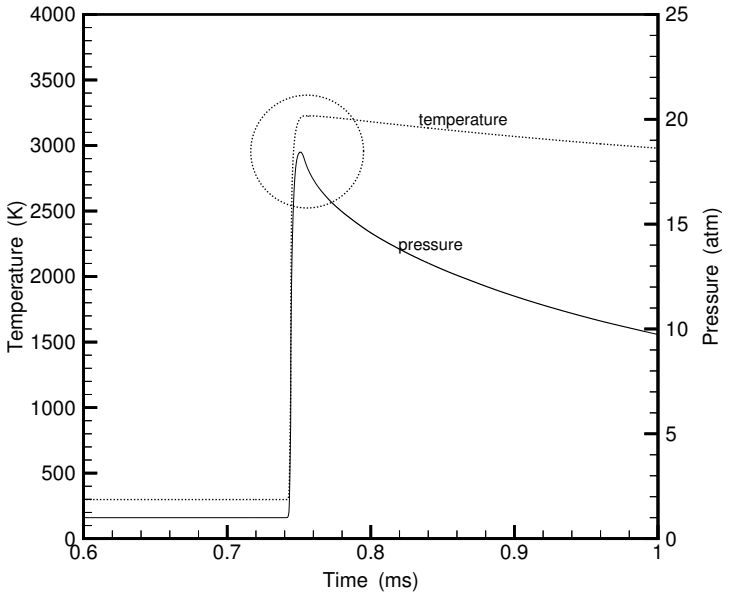

Figure 5.9 Profile of a Typical Detonation Wave

near the left-end wall. These figures show the evolution of pressure, density, temperature and particle velocity profiles, as a function of distance from initiation point.Figure 5.9 shows the time histories of pressure and temperature for a typical detonation wave at a fixed location

The first thing that can be observed is the self sustaining nature of the propagating detonation wave and the shape of the wave can be seen to adequately follow the usually description of a detonation wave. A strong shock that is the leading part of the detonation front compresses the gas mixture, and chemical reactions are triggered by a temperature rise from this strong shock compression. The leading shock is supported in turn by the energy released from the chemical reactions. The expansion of the burned gas immediately follows to match the boundary condition, and the values in the constant region are determined to meet the left wall boundary condition of zero velocity.

In general, a higher order scheme may yield more accurate results but requires more computing time. The parallel algorithm is designed specifically to take care of both spatial and temporal accuracy by performing second-order computations for Runge-Kutta for time integration, and MUSCL approach for higher-order approximations. It is noted that third-order Runge-Kutta integration results in very slight increase in accuracy whereas a drastic increase in computation time.

The properties of a fully developed detonation wave should be the same regardless of the geometry involved whenever initial condition and composition of the fuel-air mixture are the same. Hence, there are four cases with two kinds of geometry and two initiation methods, which are two-dimensional calculations with planar initiation and with point initiation, repeated for axisymmetric flow with the same initiation methods. The calculation domain is 68.5 cm x 3.75 cm for both and is filled with stoichiometric hydrogen-air mixture. Initiation is on the left wall.

Figures 5.10-5.13 show pressure contour plots at specified times for the above four combinations of geometry. The resulting shapes of the propagating detonation waves is well observed in these figures. In figure , a planar wave can be observed to evolve from a spherical wave originated from a point initiation through interaction of reflecting waves. The interaction of the waves and the formation of the resulting detonation wave can be seen more easily in figures 5.14 and 5.15 that depict the time evolution of the pressure profiles along the center line. The large jump in the pressure wave behind the wave front after 50 microseconds is due to reflection of waves from the circumferential wall. It is also noted that this reflected wave reabsorbs into the wavefront to form a fully developed detonation wave.

For point initiation cases, detonation wave velocities and pressures on the axis are observed to be lower than those of planar initiation cases and keep increasing, whereas detonation velocities and pressures on the wall are much higher than those of planar

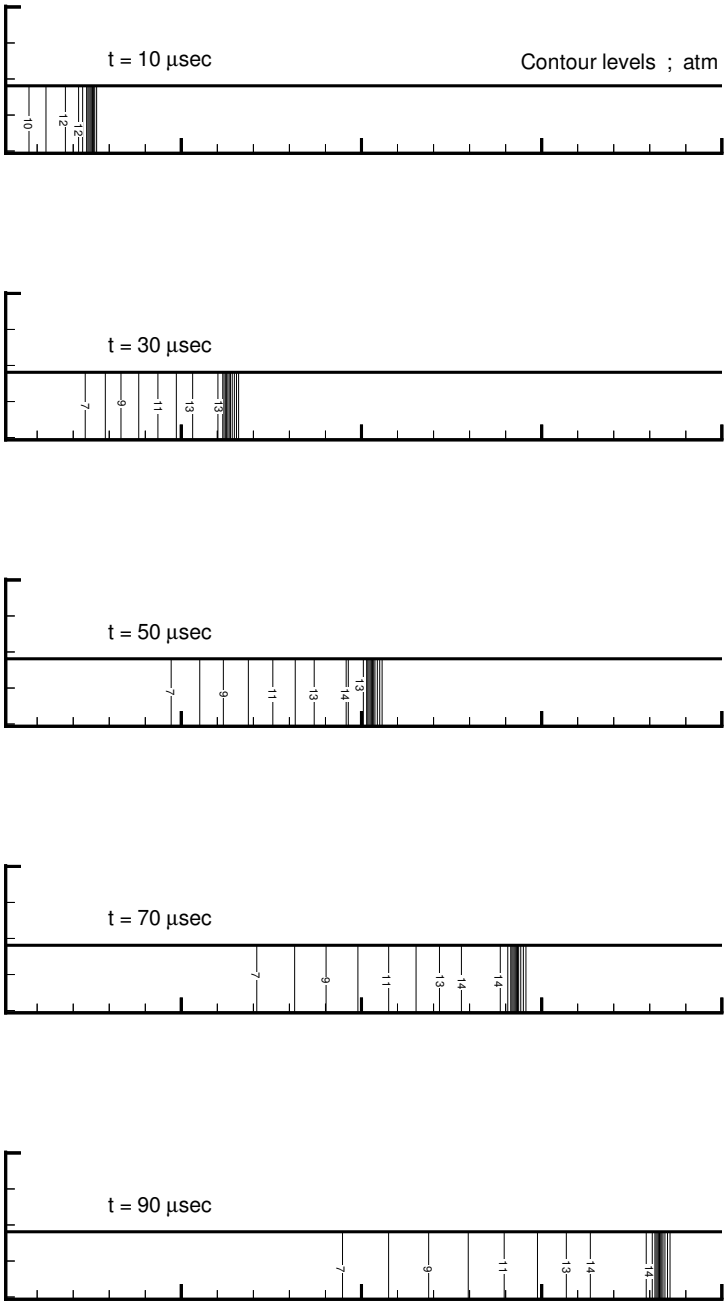

Figure 5.10 Planar Initiation in 2-D geometry

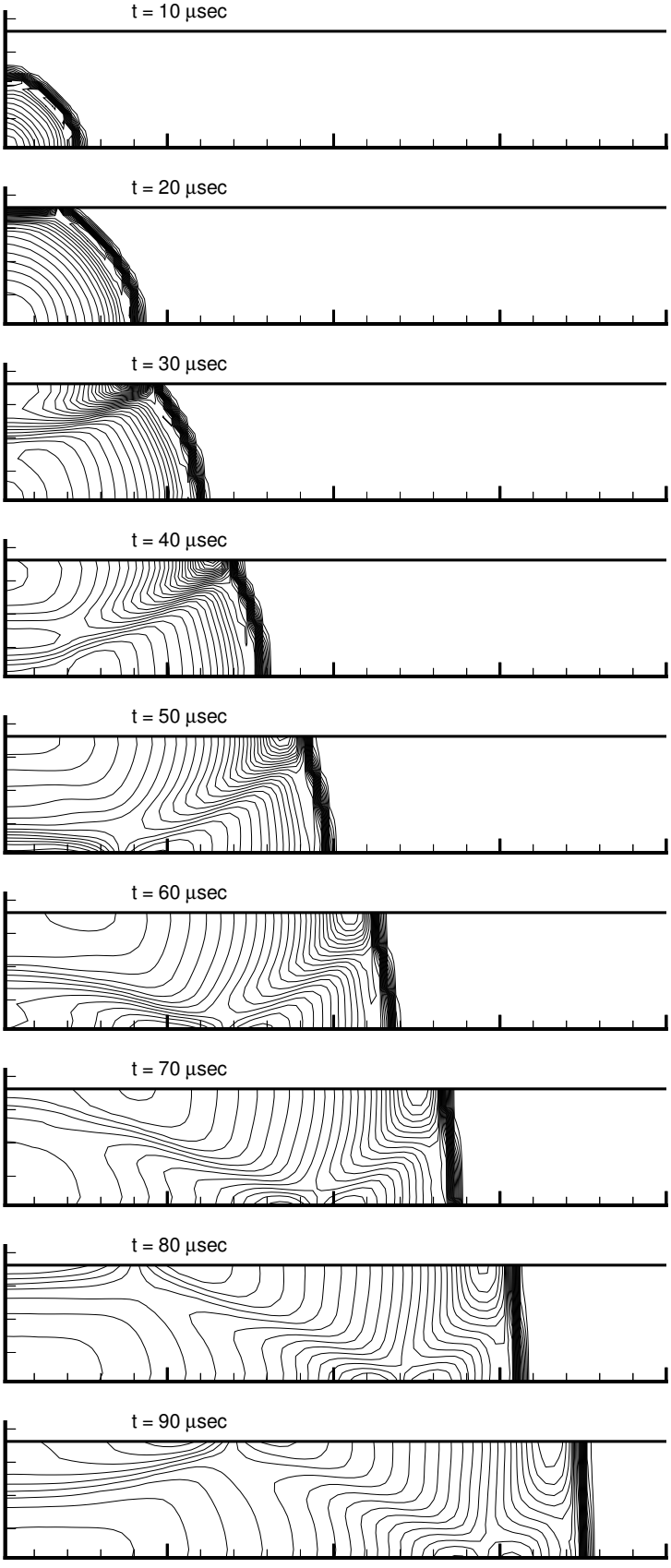

Figure 5.11 Point Initiation in 2-D geometry

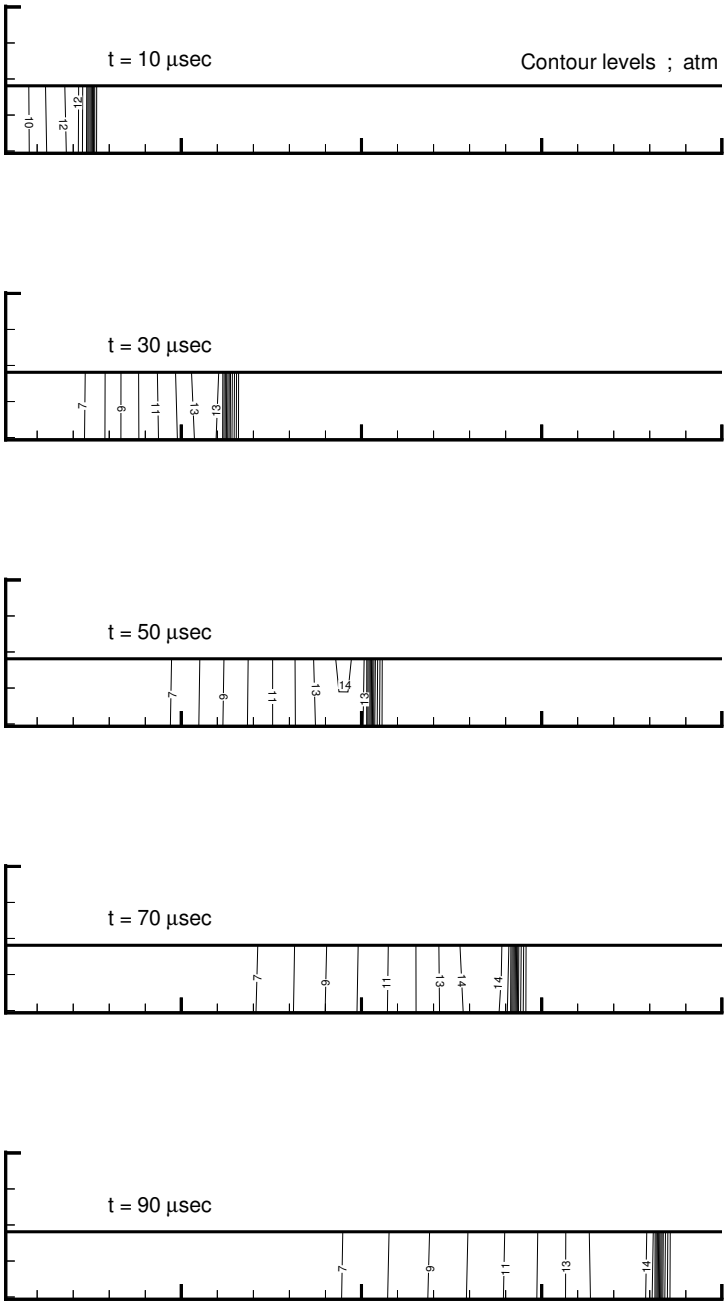

Figure 5.12 Planar Initiation in Axisymmetric Geometry

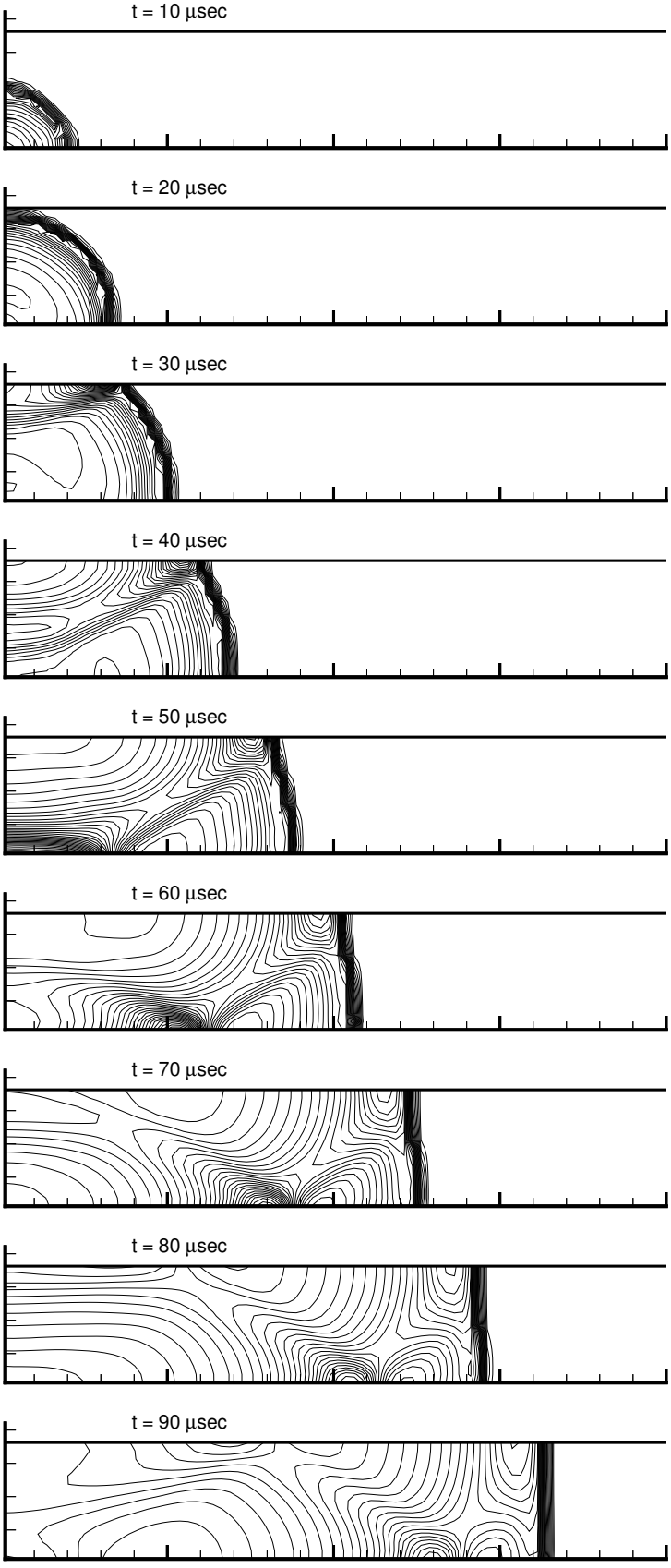

Figure 5.13 Point Initiation in Axisymmetric geometry

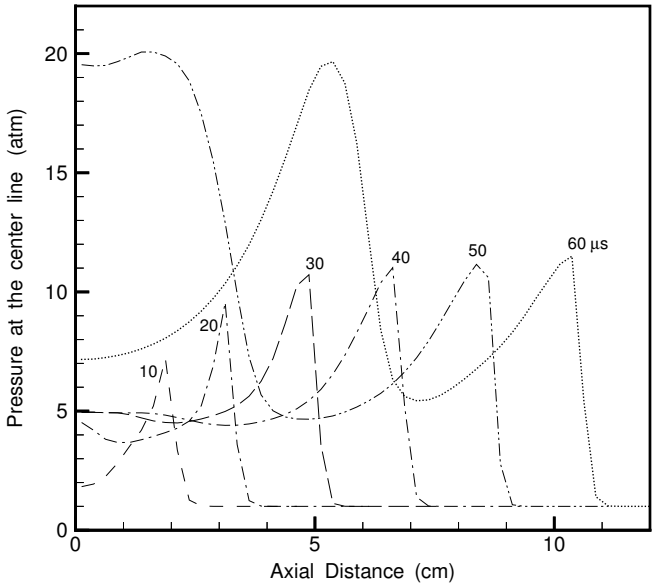

Figure 5.14 Early Profiles of Pressure Propagation

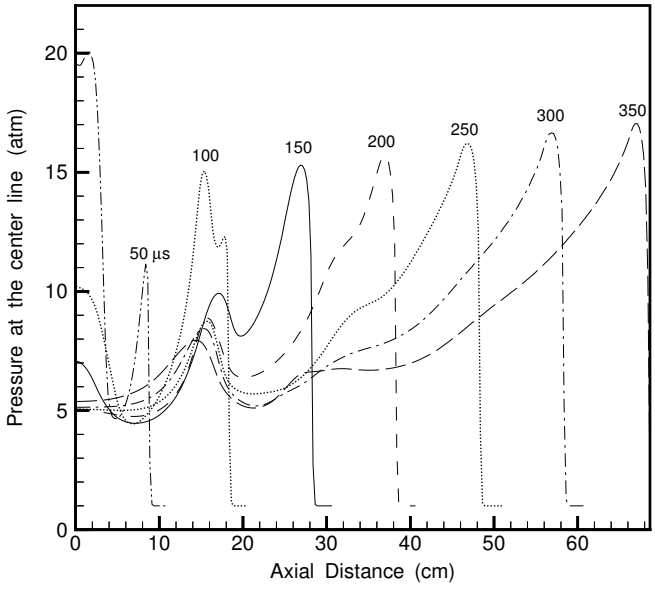

Figure 5.15 Developing Profiles of Pressure Propagation

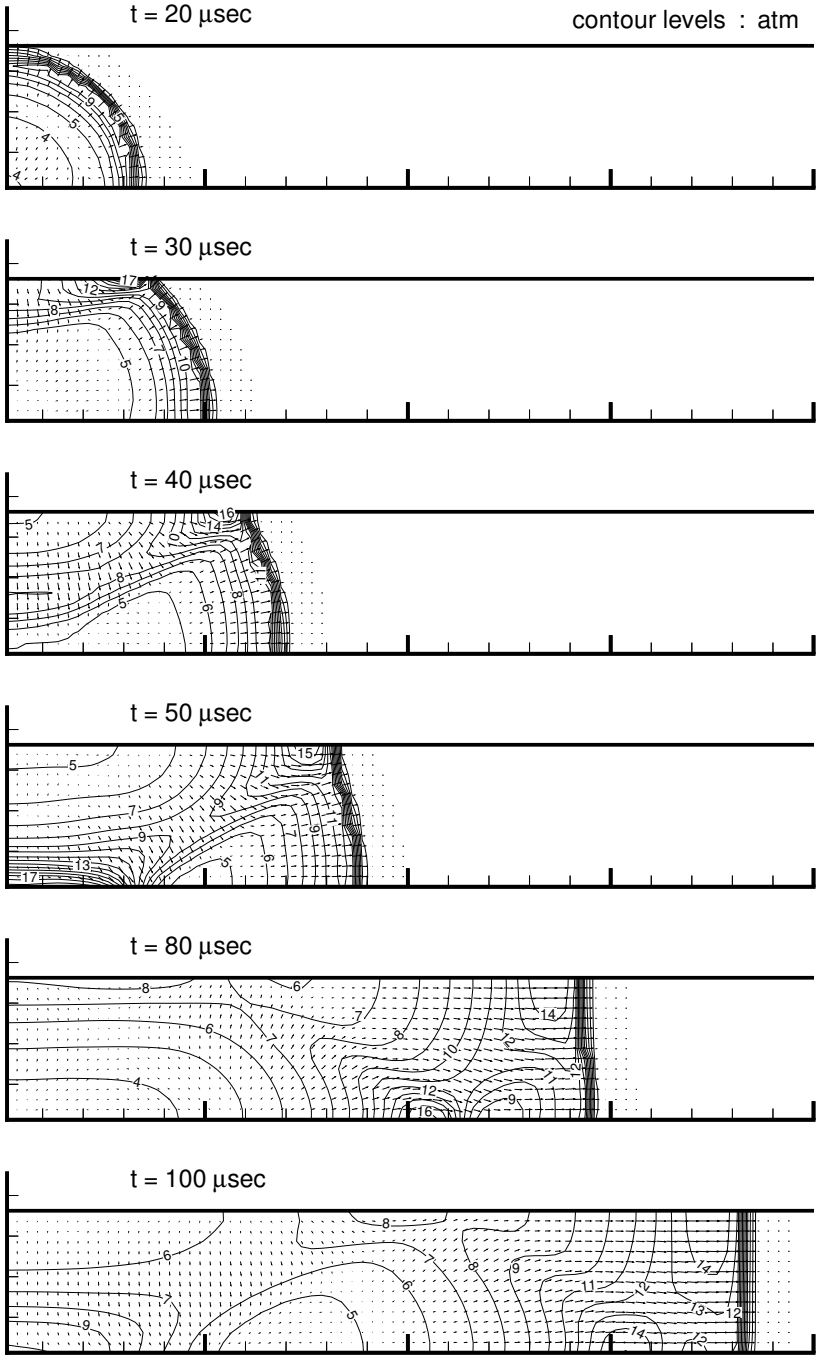

Figure 5.16 Successful Initiation for a hot spot at T=1500K, P=10 atm

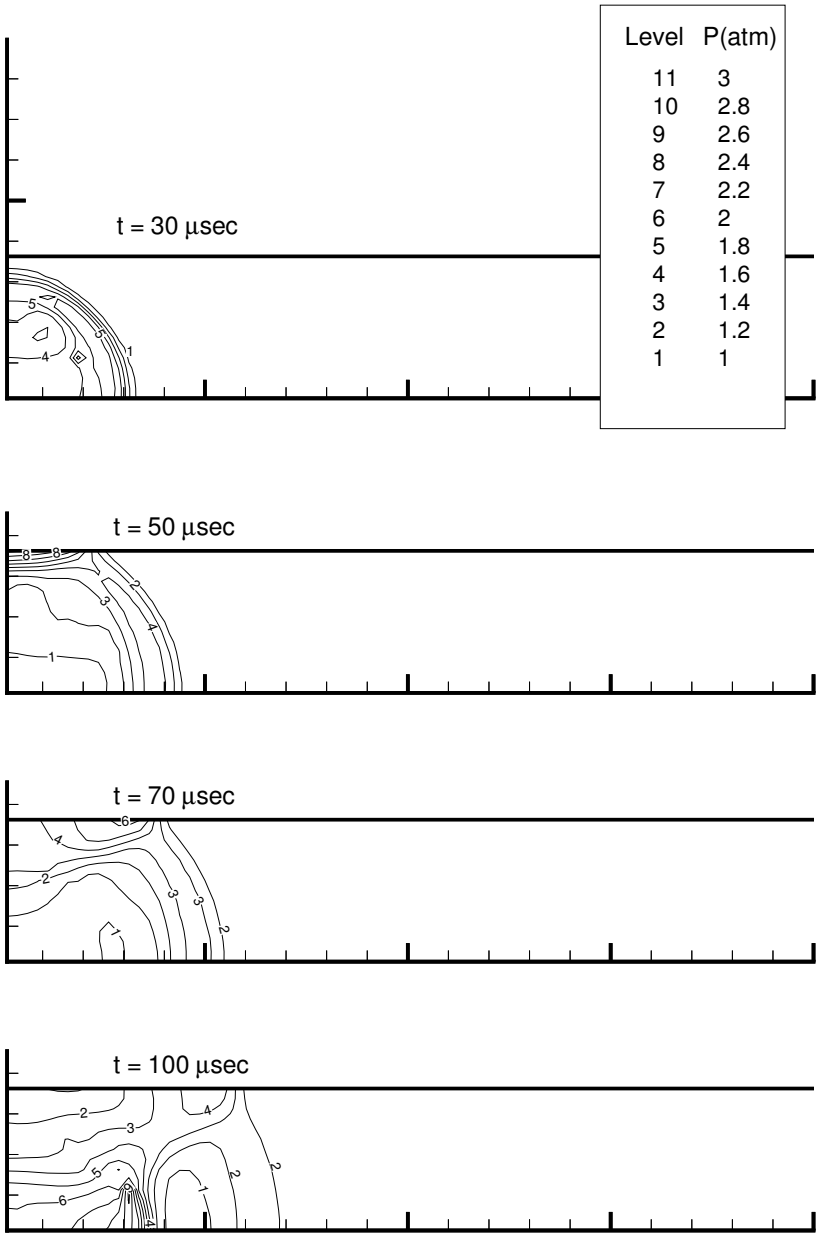

Figure 5.17 Unsuccessful Initiation of T=1500K, P=5 atm

initiation cases and keep decreasing until the formation of the planar wave. The higher pressures and velocities on the wall is due to the reflected detonation wave. It is also observed that detonation velocities and the pressures for all for cases converge to the same value at a certain distance from the initiation point where fully developed planar waves are formed. The figures 5.16 and 5.17 show successful and failed initiation of detonation waves under given conditions of pressure and temperature.

5.3 Benchmark Results of the Proto-Cluster

This chapter concludes with the results of benchmark performed on the proto-cluster. The benchmark was performed to produce an estimate of the computing efficiency as well as the cumulative processor speed obtained from the cluster of 8 nodes.

The tool used in benchmarking this cluster is High Performance Linpack (HPL) [66]. HPL is a software package that solves a dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. It is portable as well as freely available implementation of the High Performance Computing Linpack Benchmark.

The algorithm used by HPL can be summarized by the following keywords: Two-dimensional block-cyclic data distribution - Right-looking variant of the LU factorization with row partial pivoting featuring multiple look-ahead depths - Recursive panel factorization with pivot search and column broadcast combined - Various virtual panel broadcast topologies - bandwidth reducing swap-broadcast algorithm - backward substitution with look-ahead of depth 1.

The HPL package provides a testing and timing program to quantify the accuracy of the obtained solution as well as the time it took to compute it. It is dependent on a variety of factors, from communication bandwidth to the shared memory available, nevertheless, with some restrictive assumptions on the interconnection network, the algorithm described here and its attached implementation are scalable in the sense that

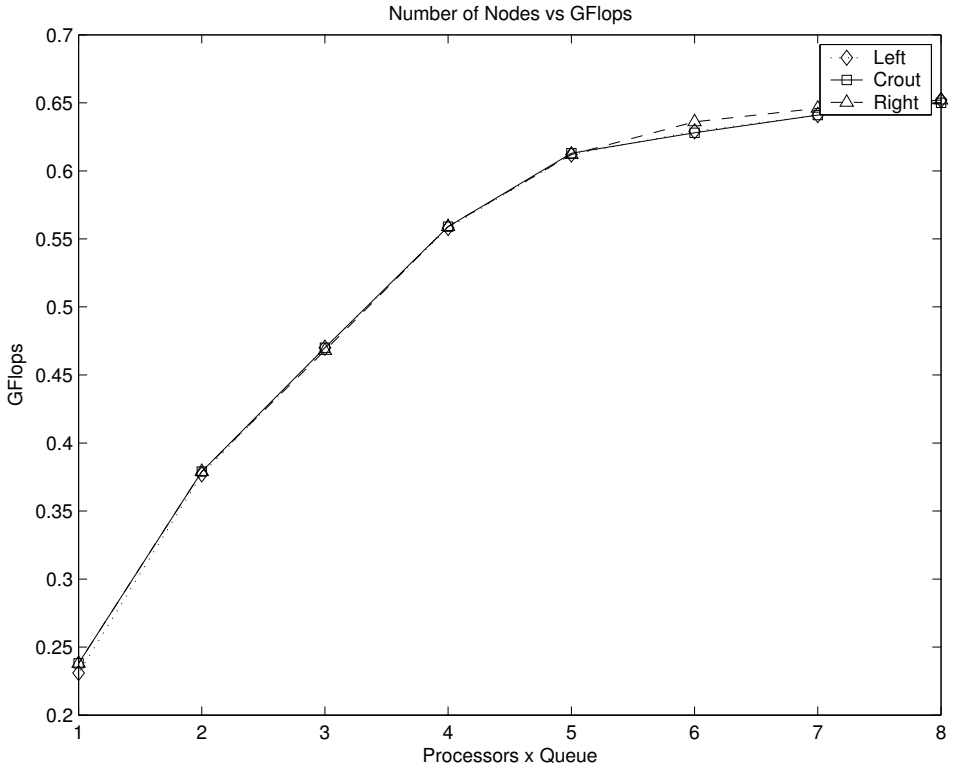

Figure 5.18 Number of Nodes versus Processing Power

their parallel efficiency is maintained constant with respect to the per processor memory usage.

The figures 5.18 and 5.19 give a representation of cumulative processor speed and time versus the number of process queues, that is nodes. The cluster comprises of 6 Pentium 550 MHZ machines and 2 Pentium 450 MHz machines. The optimum processing power is 500 MHz obtained by overclocking the slower machines and underclocking the faster machines. It is noticed that there is not a significant change in the amount of processor speed nor time taken, due to limitations of the system. The scalability of the system can be improved by increasing the bandwidth of the switch and the processing speed can be improved on executing the code on faster processors. It is noticed that communication plays an effective role, and increasing the bandwidth of the fast ethernet switch and the ethernet adapters would significantly improve scalability.

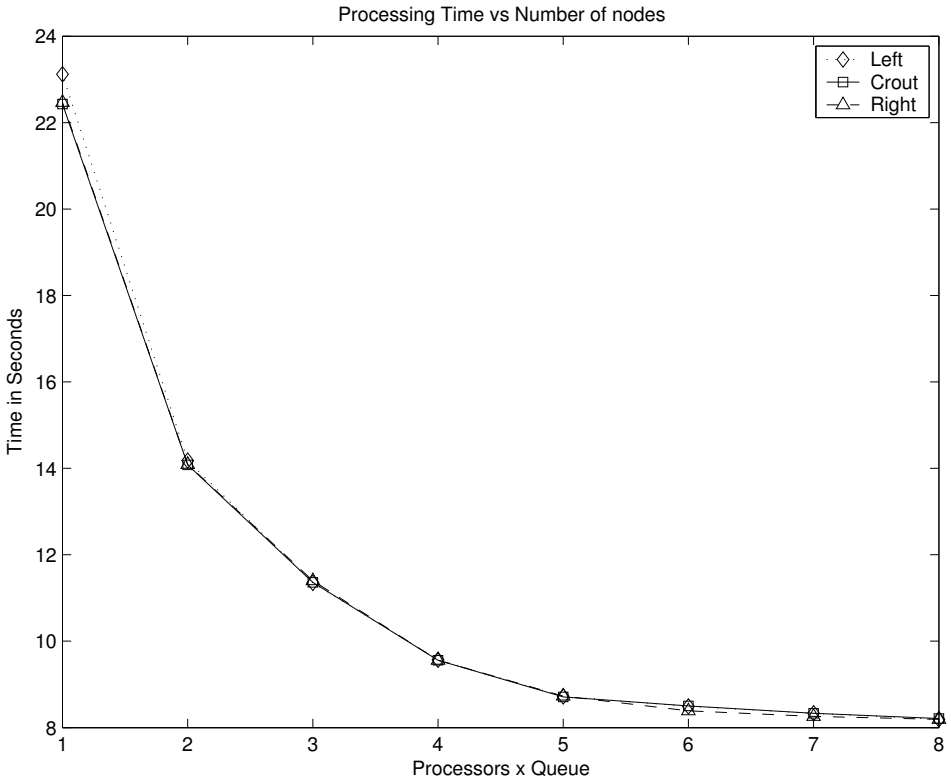

Figure 5.19 Number of Nodes versus Computation Time

CHAPTER 6

Conclusions

A parallel numerical model to simulate the detonation phenomena in a pulse detonation engine has been developed in the course of this work. The flow equations are assumed to be inviscid and non heat conducting and are coupled with the chemical kinetics of the reactions for accountability of chemical non-equilibrium. Vibrational energy conservation based on the two temperature model is used to account for possible thermal excitation. Finite Volume formulation is used to discretize the governing equations due to its conservative property. Second order Runge-Kutta integration scheme has been adopted for obtaining a time accurate solution, with a point implicit treatment of the source terms and Roes flux difference split scheme for non-equilibrium flow is implemented for cell-face fluxes, with MUSCL approach used for higher order spatial accuracy. The parallel algorithm was implemented on space-time marching and has been validated by comparison with the work done by Dr. Kim.

Numerical schemes to the third-order have been tested in both temporal and spatial regimes. Though orders higher than second result in longer periods of computation, the parallel algorithm is sufficiently stable to reduce the computation time by half for higher order schemes and close to a third for second order schemes. Mesh size has played an important factor in computation times with very slight difference in accuracy. Smaller mesh sizes have resulted in fluctuations in the variables, with no change in convergence. Hence an optimum mesh size of 2.5 mm was tested and proven to be successful. Communication build-up was another major detrimental factor in obtaining faster results. This is usually taken care of by clearing the communication channels.

The proto-cluster used for performing parallel computations has performed well and can be used as a technology demonstrator for future work in the field of computational fluid dynamics. It is a viable form of cheap supercomputing and has many advantages when compared to normal parallel computers. The scale-up charts indicate the scalability of the cluster. The proper fine-tuning of the cluster depends on the type of algorithm being executed and is usually a straightforward task.

6.1 Recommendations on Future Work

There are two directions specifically for the work to progress. One would be to employ the parallel algorithm to smaller meshes, higher-order numerical schemes, higher-step and more complex chemical reactions involving propane and air. Also, since the current numerical simulation is based on inviscid flow equations, viscous equations can be simulated with boundary layer effects. Accurate boundary layer simulation involves long computation times, and can be numerically solved by distributed computing techniques.

The second direction of progress would be to develop an intelligent general algorithm capable of handling any kind of numerical schemes. The current algorithm uses less-intensive 1D block communication pattern due to constraints in communication. A polyalgorithm, capable of intelligent fine-tuning depending on computational requirements could be developed so that heat-conduction can be taken into consideration and yet accuracy and efficiency remains unaffected. This would lead to the development of a tool to perform a complete cycle anaylsis of a pulse detonation engine.

REFERENCES

[1] M. J. Grismer and J. M. Powers, "Numerical predictions of oblique detonation stability boundaries," in *15th Intl. Colloquim. Dynamics of Explosions and Reactive Systems*, July 1995.

[2] J. A. Reiman and M. D. Gustafson, "Air force aeronautical hypersonic systems: Linking needs to concepts to technology requirements," in *Proc. of the 13 International Symposium on Airbreathing Engines*, 1997.

[3] S. Douglass, "B-3 and beyond," p. 46, 2000.

[4] R. Chase and M. Tang, "The quest for single stage earth-to-orbit: Tav, nasp, dc-x and x-33 accomplishments, deficiencies and why they did not fly," in *11th AIAA/AAAF International Conference Space Planes and Hypersonic Systems and Technologies*, 2002.

[5] W. J. D. Escher, "Rocket-based combined-cycle (rbcc) powered spaceliner class vehicles can advantageously employ vertical takeoff and landing (vtol)," in *6th International Aerospace Planes and Hypersonics Technologies Conference*, vol. AIAA-1995-6145, April 1995.

[6] D. L. Chapman, "On the rate of explosions in gases," p. 90, 1899.

[7] E. Jouget, "On the propagation of chemical reactions in gases," *J. Math. Pure Appliations*, vol. 1, p. 347, 1905.

[8] I. B. Zel'dovich, "On the theory of the propagation of detonation in gases," *Zh. Eksp. Teor. Fiz.*, vol. 10, p. 542.

[9] J. von Neumann, *Theory of Detonation waves*, ser. John von Neuman, Collected Works, vol. 61942.

[10] W. Doering, "On the detonation processes in gases," *Ann. Phys.*, vol. 43, p. 421, 1943.

[11] A. Bourlioux and A. J. Majda, "Theoretical and numerical structure for unstable 2-d detonations," *Combust.Flame.*, vol. 90, p. 211, 1992.

[12] C. Campbell and D. W. Woodhead, "The ignition of gases by an explosive wave part: 1 carbon monoxide and hydrogen mixtures," *J. Chem. Soc.*, p. 1572, 1927.

[13] S. Eidelmann, W. Grossman, and I. Lottati, "Review of propulsion applications and numerical simulations of the pulsed detonation engine concept," *Journal of Propulsion and Power*, vol. 7(6), p. 857.

[14] G. Roy, "Practical pulse detonation engines:how far are they?" in *ISABE*, 2001, p. 1170.

[15] K.Kailasanath, "Recent developments in the research on pulse detonation engines," in *AIAA*, 2002, p. 470.

[16] E. Jouget, "Mecanique des explosifs," 1917.

[17] K. Kailasanath, G. Patnaik, and C. Li, "The flowfield and performance of pulse detonation engines," in *Proc. Of 29th Symposium (Intl.) on Combustion.*

[18] K. Kailasanath, "A review of pde research: Performance estimates," in *AIAA 39th Aerospaces Meeting*, vol. AIAA paper 2001-0474.

[19] K. Kailasanath, "Review of propulsion applications of detonation waves," *AIAA Journal*, vol. 38(9), p. 1698.

[20] M. A. Nettleton, *Gaseous Detonations: their nature, effects and control.* New York: Chapman and Hall.

[21] K. J. Shchelkin and Y. K. Troshin, *Gas Dynamics of Combustion.* Baltimore: Mono Book Corporation.

[22] I. B. Zel'dovich and A. S. Kompaneets, *Theory of Detonation.* New York: Academic Press.

[23] P. M. Chung and D. Anderson, "Heat transfer around blunt bodies with non-equilibrium boundary layers," in *Proc. Of 1960 Heat Transfer and Fluid Mechanics Instt.* 1960: Stanford University Press.

[24] W. H. Dorrance, *Viscous Hypersonic Flow.* McGraw-Hill Book Company Inc., 1962.

[25] F. G. Blottner, "Chemical non-equilibrium boundary layer," *AIAA Journal*, vol. Vol. 8, no. No. 3, 1970.

[26] J. L. Stollery and C. Park, "Computer solutions to the problem of vibrational relaxation in hypersonic nozzle flows, Tech. Rep. 115, January 1963.

[27] G. Moretti, "A new technique for the numerical analysis of non-equilibrium flows, Tech. Rep. 3, 1965.

[28] J. G. Hall and C. E. Treanor, "Non-equilibrium effects in supersonic nozzle flows, Tech. Rep. CAL 163, March 1968.

[29] J. D. Anderson, "A time-dependent analysis for vibrational and chemical non-equilibrium nozzle flows," *AIAA Journal*, vol. 8, no. 3, Mar 1970.

[30] B. Grossman and P. Cinella, "Flux-split algorithm for flows with non-equilibrium chemistry and vibrational relaxation," *J. of Comp. Phys.*, vol. No. 88, p. 131, 1990.

[31] Y. Liu and M. Vinokur, "Upwind algorithms for general thermochemical non-equilibrium flows," in *27th Aerospace Sciences Meeting*, 1989.

[32] P. A. Gnoffo and R. S. McCandless, "Three dimensional aotv flowfields in chemical non-equilibrium," *AIAA Journal*, vol. 86-0230, 1986.

[33] H. G. M., "High resolution calculations of unsteady 2-dimensional non-equilibrium gas dynamics with experimental comparisons," in *19th AIAA Fluid Dynamics, Plasma Dynamics and Lasers Conference*, June1987.

[34] J. H. Miller, "Development of an upwind pns code for thermo-chemical non-equilibrium flows," *AIAA Journal*, vol. 95-2009, June 1995.

[35] J. L. Cambier and H. G. Edelmann, "Preliminary numerical simulations of a pdwe," *AIAA Journal*, vol. 88-2960, July 1988.

[36] S. Eidelmann, W. Grossman, and I. Lottati, "Computational analysis of pulsed detonation engines and applications," *AIAA Journal*, vol. 90-0460.

[37] S. Eidelmann, W. Grossman, and I. Lottati, "Air breathing pulse detonation engine concept: A numerical study," *AIAA Journal*, vol. 92-0392, 1992.

[38] S. Eidelmann, W. Grossman, and I. Lottati, "Parametric study of air breathing pde," *AIAA Journal*, vol. 92-0392, 1992.

[39] S. Eidelmann, X. Yang, and J. K. Tegner, "Pulsed detonation engine: Key issues," *AIAA Journal*, vol. 95-2754, 1995.

[40] X. He and A. R. Karagozian, "Numerical simulation of pulse detonation engine phenomena," *SIAM Journal of Computing*, 2003.

[41] H. Kim, D. A. Anderson, F. K. Lu, and D. R. Wilson, "Numerical simulation of transient combustion process in pulse detonation engines," in *AIAA 38th Aerospace Sciences Meeting*, vol. AIAA Paper 2000-0887, January 2000.

[42] H. Kim, "Numerical simulation of transient combustion process in a pulse detonation wave engine."

[43] D. R. Hofstadter, *Godel, Escher, Bach: An Eternal Golden Braid.* New York: Basic Books, 1979.

[44] C. Park, "Assessment of two-temperature kinetic model for ionizing air," *AIAA Journal*, vol. AIAA 87-1574, June 1987.

[45] J. H. Lee, *Basic Governing Equations for the Flight Regimes of Aeroassisted Orbital Transfer Vehicles*, 1985, vol. No. 96.

[46] E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow.* New York: Elsevier, 1987.

[47] J. C. Tannehill, D. A. Anderson, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*. Taylor and Francis, 1997.

[48] B. J. McBride, S. Heimel, J. G. Ehlers, and S. Gordon, "Thermodynamic properties to 6000 k for 210 substances involving the first 18 elements," *NASA*, vol. NASA SP-3001, 1963.

[49] S. Gordon and B. J. McBride, "Computer program for calculation of complex chemical equilibrium compositions, rocket performance, incident and reflected shocks, and chapman-jouguet detonations," *NASA*, vol. NASA SP-273, 1971.

[50] G. V. Candler, *The Computation of Weakly Ionized Hypersonic Flows in Thermo-Chemical Nonoequilibrium*. University of Texas at Arlington, 1988.

[51] W. G. Vincenti and J. Kruger, C. H., *Introduction to Physical Gas Dynamics*. Krieger Publishing Company, 1965.

[52] R. C. Millikan and D. R. White, "Systematics of vibrational relaxation," *Journal of Chem. Phys.*, vol. No.39, p. 3209, 1963.

[53] T. Bussing and E. Murman, "Finite volume method for the calculation of combustible chemically reacting flows," *AIAA Journal*, vol. 26(9), 1988.

[54] P. Roe, "Approximate reimann solvers, parameter vectors and difference schemes," *Journal of Computational Physics*, vol. 43(2), 1981.

[55] B. Grossman and P. Cinella, "Flux split algorithms for flows with non-equilibrium chemistry and vibrational relaxation," *Journal of Computational Physics*, vol. 88(1), 1991.

[56] P. A. Gnoffo, R. N. Gupta, and J. L. Shinn, "Conservation equations and physical models for hypersonic air flows in thermal and chemical nonequlibrium," *NASA*, vol. NASA TP-2867, 1989.

[57] S. P. Sharma, W. M. Huo, and C. Park, "The rate parameters for coupled vibration - dissociation in a generalized schwartz, slawsky and herzfeld approximation," *AIAA Journal*, vol. 88-2714, June 1988.

[58] P. A. Gnoffo, R. N. Gupta, and J. L. Shinn, "Conservation equations and physical models for hypersonic air flows in thermal and chemical non-equilibrium," *NASA*, vol. TP-2867, 1989.

[59] R. Munipalli, H. Kim, D. A. Anderson, and D. R. Wilson, "Computation of unsteady non-equilibrium propulsive flowfields," *AIAA*, vol. 97-2704, July 1994.

[60] G. Pfister, *In Search of Clusters*. New York: Prentice Hall.

[61] E. Marcus and H. Stern, *Blueprints for High Availability: Designing Resilient Distributed Systems*. John Wiley and Sons.

[62] K. Kopper, *The Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software*. No Starch Press.

[63] M. Snir, *MPI: The Complete Reference*. Boston: MIT Press.

[64] A. N. Laboratory, "The message passing interface standard," July 2005.

[65] R. Rogers and W. Chinitz, "Using a global hydrogen-air combustion model in turbulent reacting flow calculations," *AIAA Journal*, vol. 21(4), 1983.

[66] A. Petitet, R. C. Whaley, J. Dongarra, and A. Cleary, "Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," January 2004.

BIOGRAPHICAL STATEMENT

Prashaanth Ravindran was born in Madras (now Chennai) in India, in 1981. He received his elementary education in Montfort Preparatory School and high school diploma from Santhome Higher Secondary school. He started his Bachelor's degree in Aerospace Engineering in University of Madras, in 1998 and graduated in 2002. Between 2002 and 2003 he worked on various contracts in CAD with Central Institute of Plastics Engineering and Technology. He joined University of Texas at Arlington in 2003 to pursue his graduate studies in Aerospace Engineering. His areas on interest includes Computational Fluid Dynamics, Computer Aided Design and as a hobby, system administration.