Modeling and Simulation of Axisymmetric Stagnation Flames

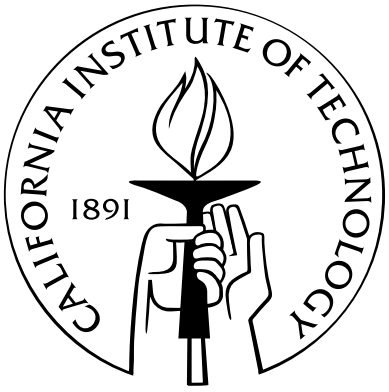
Thesis by

Kazuo Sone

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2007

(Defended May 18, 2007)

To Juri

Acknowledgements

I would like to thank my advisors, Professor Paul Dimotakis and Professor Daniel Meiron, for guiding me throughout my doctoral research. It was truly a privilege to be able to work with them. This research project was very challenging, yet I was able to complete it because of their guidance. I was constantly inspired by their knowledge and intelligence. I would also like to thank Professors David Goodwin, Joseph Shepherd, and Dale Pullin for serving on my thesis committee and providing many suggestions that improved the quality of this thesis. Outside of the thesis committee, I am indebted to Professors Anthony Leonard and Tim Colonius for the discussion of the polar axis treatment that was useful in developing the new basis functions used in this study.

This research project is running in parallel with experiments. In addition to providing experimental data, I had many discussions on the subject with Jeff Bergthorson, and that significantly influenced this thesis. Laurent Benezech also helped me with additional data, some post-processing of data and by providing some of one-dimensional solutions used in this study. Many members of the Compressible Turbulence group gave me input including Carlos Pantano, David Hill, George Matheou, Trent Mattner, and Patric Hung. I am grateful for the discussions with them.

The members of the Center for Advanced Computing Research at Caltech and Livermore computing were very supportive by maintaining and updating computational resources, as well as giving me a hand in computing. In particular, I am very thankful to Michael Aivazis, Sharon Brunett, Mark Bartelt, Julian Cummings, and Jan Lindheim. Computer support by Dan Lang and administrative support by Christina Mojahedi at Professor Dimotakis' group are also appreciated. This project was funded by the Caltech Advanced Simulation and Computing (ASC) program of the Center for Simulation of Dynamic Response of Materials, and the generous computing resources are also provided by the program, with additional computing resources provided by the Air Force Office of Scientific Research.

Finally, I would like to thank my wife, Juri for supporting and encouraging me throughout my doctoral studentship.

Abstract

Laminar flame modeling is an important element in turbulent combustion research. The accuracy of a turbulent combustion model is highly dependent upon our understanding of the laminar flames and their behavior in many situations. How much we understand various phenomena can only be measured by a model that describes the phenomena and by how well the model describes and predicts them. One of the most commonly used methane combustion models is GRI-Mech 3.0. However, how well the model describes the reacting flow phenomena is still uncertain, even after many attempts to validate the model or quantify uncertainties. This is because, in flames, chemisty is coupled with fluid mechanics, thermodynamics, and transport process, and the separation of one from another is not easy, if at all possible.

In the present study, the behavior of laminar flames under different aerodynamic and thermodynamic conditions is studied numerically in a stagnation-flow configuration. The present study follows an experimental study by J. Bergthorson conducted earlier in our group. In numerical study of reacting flows, a one-dimensional model is commonly used to assess the performances of chemical kinetics models. The model describes stagnation flames along the symmetric axis through several key assumptions. One such assumption is a uniform pressure-eigenvalue assumption, *i.e.*, that the curvature of the pressure field is uniform throughout. Although it is shown that this assumption does not hold through more sophisticated numerical studies capable of a two-dimensional description, it is shown that the model works reasonably well in the case of non-reacting (cold) flow and diluted hydrogen flames. However, how well the assumption holds and whether or not the model approximates hydrocarbon flames well are not known. The present study employs the full chemical kinetics model of methane combustion, and a realistic transport model that accommodates differential-diffusion effects within an axisymmetric two-dimensional flow modeling. This allows direct comparisons of two-dimensional and one-dimensional models,

as well as numerical and experimental data, to quantify modeling errors that arise from the use of one-dimensional hydrodynamics model and the chemical-kinetics model.

In order to make such a numerical study possible, the spectral element method is reformulated to accommodate the large density variations in methane reacting flows. In addition, a new axisymmetric basis function set for the spectral element method that satisfies the correct behavior near the axis is developed that avoids the well-known singular behavior there. This basis function satisfies all the parity requirements, by construction, that axisymmetric fields must meet. To accomodate computationally expensive detailed methane combustion and transport models, efficient integration techniques are developed to accurately model axisymmetric reacting flow within a reasonable amount of computational time. The numerical method is implemented using an object-oriented programming technique, and the resulting computer program is verified with several different methods.

First, cold-flow simulation is conducted to understand the nature of the underlying flow field without chemical reactions. It is shown that detailed modeling of the experimental apparatus is important for a direct comparison of numerical simulation and experiments to be meaningful.

Reacting flow simulations are conducted in three phases: one-dimensional simulations by Cantera, two-dimensional simulations with an idealized representation of the experimental configuration, and finally, simulations with full details of experimental setup. It is shown that, although the plug-flow boundary condition cannot be used, as is, to predict flame locations, the model can reliably be used to predict flame speed under strain. Through a direct simulation of laboratory flames that allows direct comparison to experimental data, the present study then shows variances with the commonly used GRI-Mech 3.0 chemical kinetics model. It is shown that the methane combustion model based on GRI-Mech 3.0 works well for methane-air mixtures near stoichiometry. However, GRI-Mech 3.0 leads to an overprediction of laminar flame speed for lean mixtures and an underprediction for rich mixtures. This result is slightly different from conclusions drawn in previous work, in which experimental data are compared with a one-dimensional numerical solution. Detailed analysis reveals that flame speed is sensitive to even slight flame front curvature as well as to its finite extension in the radial direction. Neither of these can be incorporated in one-dimensional flow modeling.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Often, it is people rather than the technology itself who pose the greater challenge to solving an otherwise technical problem.—John Lakos

1.1 Motivation

The behavior of laminar flames and insights into such phenomena have many implications to our understanding of turbulent as well as laminar flames. Even when the underlying flow is turbulent, as is the case in internal combustion or jet engines, the flame in the case of fast kinetics (high Damköhler number) is considered as an ensemble of stretched laminar flames called flamelets (Williams, 1975). Recent advances in flamelet modeling are due, in part, to the understanding of the laminar flame structure and behaviors both experimentally and numerically (Law, 1988; Law & Sung, 2000; Williams, 2000). In particular, the laminar flame speed under the influence of aerodynamic effects such as stretch and strain, flame curvature, or heat losses has been studied experimentally. Numerical simulations have also previously been employed to model such flame behavior, but with limited success. This is mostly because simulation of combustion phenomena and its interaction with fluid mechanics is computationally expensive and oversimplified models required to obtain numerical results within a practical amount of time were insufficient to reveal important aspects of the full phenomenology.

Simulation is used everywhere. Automobile and aerospace industries are among the major users of Computer-Aided Engineering (CAE), including Computational Fluid Dynamics (CFD), but the application of CAE goes beyond such traditional users and now extends to semiconductors, pharmaceuticals, food processing (Lange, 2007), high-performance sport-

ing goods (McKee, 2004), and so on. The use of computer simulations in science and engineering is now widespread. For example, simulations used in product design allows engineers to develop products quickly for market. The short development cycle can lead to low R&D costs and therefore cost competitiveness in the market, in addition to the direct savings from replacing some expensive laboratory experiments. Sometimes, even when cost does not come at the top of the priority list, simulations are the only possibility to conduct science, due to safety concerns when highly toxic or explosive materials are involved, or due to conditions which are either impossible or difficult to attain in the laboratory.

However, there is a down side to this rapid growth of the application of computational engineering. Toyota Motor Corp., which is known for its quality automobiles, recently suffered multiple waves of recalls, and after internal investigation, the president of the company had to say (Shirouzu, 2006):

> "We relied on computer-aided engineering and other computer analysis and didn't conduct as many quality checks as we should have."

It reminds us that simulation software—when used naively or incorrectly—can lead to a devastating result. A main part of the problem lies in the growing complications in the software.

Often, simulations are used for complicated problems in which verification of the correctness of the numerical results is not easy, and this makes simulation software prone to human mistakes, including programming errors (commonly referred to as 'bugs') and misuse. Even though it is impossible to prove the correctness of software[†] it is still possible to develop useful and reliable software. For example, there is a formal protocol called Software Quality Assurance (SQA) to ensure the reliability of software, as well as techniques to develop large-scale software such as Object-Oriented Programming (OOP) and design patterns (Gamma *et al.*, 1994). Every computational scientific work needs to address this issue, and the approach used in this study is described in Chapter 3.

Besides such human errors, there are two additional kinds of errors in every computer simulation: modeling errors and numerical errors (including discretization errors, convergence errors, and round-off errors). Modeling errors are a deviation of the mathematical

[†] *"Complete testing, erroneously used to mean 100% branch coverage. The notion is specific to a test selection criterion: i.e., testing is 'complete' when the tests specified by the criterion have been passed. Absolutely complete testing is impossible."*—from "Software Testing Techniques" by Beizer (1990).

description of the physical system (*i.e.*, model) from the real physical system. Some flow systems offer a better description than others. For example, a uniform-density, low Mach number flow is believed to be well described by the incompressible, uniform-density Navier-Stokes equations, and simulations employing such validated mathematical models are at least as accurate as thoughtfully and carefully conducted experiments. Indeed, some authors have attempted to estimate and quantify measurement errors from numerical data. However, for some types of flow systems, we do not yet have such a sufficiently high-fidelity mathematical description. One such example is chemically reacting flow systems, which is the topic of the present study. Although significant work has been done in this area, there has been only limited progress in obtaining reliable chemical kinetics models for the combustion of hydrocarbon fuels. There have been much theoretical, experimental, and numerical work on laminar flames and flamelets. However, theoretical work has been limited to cases with oversimplified assumptions, such as flames with no heat release, or infinitely thin flames. The numerical work has relied on reduced chemistry models or oversimplified hydrodynamic models.

Only few recent reports exist that used a detailed hydrocarbon combustion model with realistic fluid mechanical models. For example, Najm & Knio (2005) proposed an operator-splitting scheme for low Mach number, chemically reacting flows with GRI-Mech 1.2 that involves 32 species and 177 reactions[‡] to demonstrate their algorithm. Bell *et al.* (2005a) used the reaction mechanism of Glarborg *et al.* (2000) that includes 65 species and 447 elementary reactions to simulate a laminar diffusion flame. Many other simulations of combustion phenomena using multi-dimensional models have been reported using reduced chemistry models. For a simulation of a turbulent flame, Bell *et al.* (2005b) used a subset of GRI-Mech 1.2 mechanism that includes 20 species and 84 elementary reactions to simulate a turbulent V-flame. This is probably one of the biggest simulation in terms of the number of species and reactions used in turbulent combustion simulations, and while the resutls are very impressive, the numerically predicted angle of the V-flame is wider than that observed in the experiment by 10%. There are many chemical kinetics models proposed for methane combustion, but the lack of an appropriate framework—a multi-dimensional simulation of laboratory-scale flames that allows direct comparison of numerical data to experimental ones—limits our understanding of laminar flames, and consequently turbulent flames.

[‡]These numbers include argon whereas numbers in Table 1.1 do not.

Table 1.1: The chemical kinetics models. M is the total number of species involved, and K stands for the number of reactions in the mechanism. The reaction model number indicates reactants considered, *e.g.*, HCN indicates Hydrogen, Carbon (Hydrocarbons) and Nitrogen oxidation reaction set. Reduced mechanisms are indicated by lower-case model IDs. This list is by no means complete.

Model ID.	Name	M	K	References
HCN1	GRI-Mech 3.0	53	325	(Smith *et al.*)
HCN2	GRI-Mech 2.11	49	277	Predecessor of HCN1
HCN3	Glarborg00	65	447	(Glarborg *et al.*, 2000; Bell *et al.*, 2005a)
HC1	GRI-Mech 3.0	35	217	Sub-mechanism of HCN1
HC2	GRI-Mech 1.2	31	175	Predecessor of HCN2
HC3	SD05 (Rel. 03/10)	39	175	(San Diego mechanism)
HC4	DLW99	71	469	Based on HC2, (Davis *et al.*, 1999, 2002b)
HC5	Wang99	52	367	(Wang *et al.*, 1999)
HC6	Marinov99	57	383	(Marinov, 1999)
HC7	WF97	33	192	(Wang & Frenklach, 1997)
HC8	Tan94	78	473	(Tan *et al.*, 1994)
HC9	EDL92	30	171	(Egolfopoulos *et al.*, 1992)
hc1	Smooke92	26	83	(Smooke *et al.*, 1992; Day & Bell, 2000)
hc2	Smooke86	16	46	(Smooke *et al.*, 1986)
hc3	DRM-19	20	84	Subset of HC2, Bell *et al.* (2005b)
hc4	1-step model	4	1	(Westbrook & Dryer, 1981)
H1	GRI-Mech 3.0	9	28	Sub-mechanism of HC1
H2	SD05 (Rel. 03/10)	9	22	Sub-mechanism of HC3
H3	DLW99	9	28	Sub-mechanism of HC4
H4	YDR91	9	19	(Yetter *et al.*, 1991; Frouzakis *et al.*, 1998)
H5	MW88	9	37	(Maas & Warnatz, 1988)

This is very unfortunate because our everyday life relies heavily on combustion: furnance, gas turbine, and automobile engines, just to name a few. Presently in this country, slightly more than three quarters of our electricity supply relies on combustion[§].

There are several chemistry models for methane combustion available. For example, Egolfopoulos & Dimotakis (2001) used and compared several natural gas combustion models including those of Tan *et al.* (1994), GRI-Mech (Smith *et al.*), Wang & Frenklach (1997), Marinov (1999) and Wang *et al.* (1999). Bergthorson (2005) used San Diego mechanism, and a C-3 mechanism by Davis *et al.* (1999), in addition to GRI-Mech 3.0, to study laminar flame speed at various conditions. These mechanisms are summarized in Table 1.1. All are comprised of a seemingly sufficient set of elementary reactions. However, numerical

[§]Energy Information Administration, Form EIA-906, "Power Plant Report."

results obtained from each of these models vary significantly (Egolfopoulos & Dimotakis, 2001; Bergthorson, 2005). Many of them take elementary reaction parameters from experimental data and evaluate rate coefficients of a single reaction to assemble a complete set of elementary reactions. However, this approach can fail due to correlations in uncertainties in the parameters, and the best-fit values to the individual parameters do not necessarily comprises the best chemical kinetics model (Frenklach *et al.*, 1992). GRI-Mech takes a slightly different approach that is based on a systematic optimization method called solution mapping (Frenklach *et al.*, 1992), through which the mechanism is optimized to several independent flame conditions, including shock-tube ignition delay, methyl radical concentration, and laminar premixed flame speeds. As a consequence, this mechanism is regarded as the best methane combustion model without problem-specific fine tuning of parameters.

These chemical kinetics models have been primarily used in numerical studies that employ a one-dimensional formulation developed by Kee *et al.* (1988) that represents a significant advance in combustion research. Inclusion of these detailed methane combustion models in a multi-dimensional numerical simulation is computationally expensive, and numerical studies of laminar flames in multi-dimensions have typically used reduced mechanisms such as (Smooke *et al.*, 1986, 1992) that include a subset of all species involved and reduced reaction sets. It is not obvious if comparison of one-dimensional numerical solutions to laboratory-generated experimental data is valid, even when they agree. When they do not agree, it is not clear if the chemical kinetics model is the cause of error or the simplified fluid mechanics model in the one-dimensional model is responsible, or some combination of the two.

There are three problems here. One is that there has been no appropriate validation of chemical kinetics models that gives us confidence in these models and in the numerical results in studying flame behavior, or designing reaction systems. This is primarily because of the difficulty in realistic multidimensional simulations of hydrocarbon flames because of excessive computational costs. An algorithm and software that allow direct simulation of chemically reacting flows are required, and this is the second problem. The third problem is the reliability of the simulation software itself. Knupp & Salari (2003) collected appropriate procedures for testing programs for computational science and engineering, but attention has been paid to this subject only very recently.

1.2 Objective

The primary objective of the present study is to understand the validity and applicability of the GRI-Mech 3.0 chemistry model and to make progress toward a universal natural-gas (methane) combustion model that works for every fuel in every situation, including both rich and lean conditions, as well as under low or high pressure.

To achieve this goal, a computational framework that allows us to evaluate the accuracy of the chemistry model is required. An efficient and accurate algorithm to simulate a laboratory flame directly, including appropriate initial and boundary conditions, was developed. The spectral element method originally developed by Patera (1984) for incompressible flow will be extended to accommodate capabilities to simulate chemically reacting flows with large density variation efficiently. Second, error sources to computational simulations will be identified, and then the list of methods to quantify or identify errors stemming from each error source will be discussed. It will be subsequently shown that the code used in this study is correct and accurate with respect to the selected test criteria. Third, the developed computing framework will be used to study the behavior of a chemical-kinetics model. Namely the behavior of GRI-Mech at various strain and equivalence ratios will be studied for premixed methane flames. The aim is to clarify the validity of one-dimensional modeling of stagnation flow so that computationally simpler, one-dimensional models can be used judiciously. Through these developments, how aerodynamic effects such as strain, stretch, dilatation, and flow non-uniformity that creates flame front curvature can affect flame speed and flame behavior are investigated. These studies are of interest in their own light, but can also provides a good validation of the GRI-Mech 3.0 model, if the numerical results are in agreement with a theory or experimental prediction.

Chapter 2

The simulation methods

Simulation: *3. The technique of imitating the behaviour of some situation or process (whether economic, military, mechanical, etc.) by means of a suitably analogous situation or apparatus, esp. for the purpose of study or personnel training. Freq. attrib.* —Oxford English Dictionary (emphasis added by the author)

2.1 Introduction

As the definition of the word "simulation" suggests, for a numerical study to also be a simulation it must be a "suitably analogous" situation or apparatus. Many numerical studies do not include every detail of the situation or apparatus, or do not "imitate the behavior" at a reasonable accuracy (as inclusion of curved geometry, for example, usually deteriorates the accuracy of numerical methods). For example, Bell *et al.* (2005b) computed a laboratory-scale turbulent V-flame using a reduced methane chemistry model. Although this calculation used a large number of reaction and species sets compared to other turbulent combustion simulations, the numerically predicted angle of V-flame was wider than experiment by 10%. Use of reduced chemistry is certainly one cause of error, but more importantly, this might have been caused by not modeling the nozzle exit area where the flow around the anchoring rod is modified by the attached V-flame, as noted by the authors. On the other hand, the KIVA code (Amsden *et al.*, 1985, 1989; Amsden, 1993, 1997, 1999) developed at the Los Alamos National Laboratory over decades is a significant achievement in the simulation of mixing and combustion of internal combustion engines and used by many researchers including Han & Reitz (1995), Celik *et al.* (2000), and Sone & Menon (2003). Although the code has the capability to accomodate arbitrary-shaped cylinders as

well as valve and piston movements, it is very diffusive due to its first-order temporal and spatial accuracy, and its reliability is uncertain.

As the title of this thesis suggests, the present study *is* about a simulation of specific laboratory phenomena, and techniques were developed with this purpose in mind to model flames in a laboratory with a suitably analogous setup with sufficient accuracy. The detailed modeling of the experimental setup have not been done in previous works on numerical studies of laminar flames. The methodology used in this study is described in this chapter. First, the governing equations of chemically reacting flow are reviewed. Then the numerical method employed in this study, including discretization of the governing equations, is introduced. Details on the implementation of the algorithm will be deferred to the next chapter.

2.2 Mathematical models of reacting flows

2.2.1 The governing equations for axisymmetric flow at low Mach number

The governing equations of the fluid mechanics of chemically reacting flow are the compressible form of the Navier-Stokes equations. The unsteady equations have been used in many studies (Amsden *et al.*, 1989; Kim *et al.*, 1999; Haworth & Jansen, 2000). However, these studies are mostly concerned with turbulent reacting flow, and the trouble is that the laminar methane flames exhibit flame speeds far less than those of turbulent flames. Therefore, the flow time scale is usually far smaller than that of acoustic waves, and the explicit integration of equations for compressible flows leads to a very restrictive time step size compared to the time scale of the phenomena. In addition, if the compressible equations are used for low-speed flows, the pressure-gradient term becomes singular as the Mach number approaches to zero and some sort of corrections must be applied (Ramshaw *et al.*, 1985; Amsden *et al.*, 1989). To circumvent these issues, self-consistent equations for low-speed reacting flow have been derived (Majda & Sethian, 1985; McMurtry *et al.*, 1986) using low Mach number asymptotic analysis, and similar equations are derived and used by many others to study laminar flames. This formulation, often called a low Mach number formulation, or a zero Mach number formulation, effectively removes the propagation of sound waves as a means of equilibrating pressure, and thus allows larger timestep size, replacing this dynamical step by a Poisson-solver for pressure. Suppose for example, for the current

problem, the flame thickness is 1.0×10^{-4} m and 10 collocation points are necessary to resolve the flame. The advection timescale is $\Delta t^{\text{Adv}}_{\text{compress}} \sim 3.0 \times 10^{-8}$ sec. However, when a low Mach number formulation is employed, the speed of sound constraint disappears from the denominator of the advection timescale and $\Delta t^{\text{Adv}}_{\text{lowMach}} \sim 1.0 \times 10^{-5}$ sec. Significant savings in computational time can be achieved when the interaction of acoustic waves and flame is not of interest or importance.

The low Mach number formulation used in this study is conceptually different in how bulk viscosity is treated. The derivation of the following equations for density, velocity, temperature, and species used in the present study is described in Appendix A.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{2.1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p^* + \mathbf{L}^e(\mathbf{u}) + \mathbf{L}^i(\mathbf{u}) + \mathbf{f}, \tag{2.1b}$$

$$\frac{\partial Y_{\mathfrak{m}}}{\partial t} + \mathbf{u} \cdot \nabla Y_{\mathfrak{m}} = D_{\mathfrak{m}} \nabla^2 Y_{\mathfrak{m}} + \frac{1}{\rho} \nabla \rho D_{\mathfrak{m}} \cdot \nabla Y_{\mathfrak{m}} + \dot{\omega}_{\mathfrak{m}}, \tag{2.1c}$$

$$\rho C_{p,\text{mix}} \frac{DT}{Dt} = \lambda_T \nabla^2 T + \nabla \lambda_T \cdot \nabla T - \rho \sum_m h_{\mathfrak{m}} \dot{\omega}_{\mathfrak{m}}, \text{ and} \tag{2.1d}$$

$$\rho = p_0 / (\bar{R}_{\text{gas}} T), \tag{2.1e}$$

where ρ is the density, $\mathbf{u}$ is the velocity, T is the temperature, and $Y_{\mathfrak{m}}$ is the mass fraction of species $\mathfrak{m}$.

The viscous terms in the momentum equation are

$$\mathbf{L}^i(\mathbf{u}) = \frac{\mu}{\rho} \nabla^2 \mathbf{u} \tag{2.2a}$$

$$\mathbf{L}^e(\mathbf{u}) = \frac{\mu}{\rho} \nabla \left[\nabla \cdot \mathbf{u} \right] + \frac{\nabla \mu}{\rho} \cdot \left[\nabla \mathbf{u} + \nabla \mathbf{u}^t \right]. \tag{2.2b}$$

These equations exhibit a dependence on the external force, $\mathbf{f}$, the species-production rate, $\dot{\omega}_{\mathfrak{m}}$, the enthalpy of the species $\mathfrak{m}$, $h_{\mathfrak{m}}$, mixture-averaged specific heat capacity, $C_{p,\text{mix}}$, ambient pressure, p_0, and the specific gas constant, $\bar{R}_{\text{gas}}$—as well as on transport properties, μ and λ_T, which are the viscosity and thermal conductivity of the gas mixture, respectively, and $D_{\mathfrak{m}}$ the diffusion coefficient of species $\mathfrak{m}$.

In the momentum equation, p^* is the perturbational pressure, which is different from p_0

in the equation of state, and defined by

$$p^*(x,t) = (p(x,t) - p_0) - \left(\mu_{\mathrm{B}} - \frac{2}{3}\mu\right) \nabla \cdot \mathbf{u} \tag{2.3}$$

where p_0 is the leading-order pressure term, which is an ambient pressure and treated as a constant (101.3 kPa) throughout this study, and μ_{B} is the bulk viscosity. Note that p^* includes bulk-viscosity effects. It is possible to factor this term out, which leads to a different value of p^*, but one still obtains the same velocity field. This is because inclusion of the last term of Eqn.(2.3) in the momentum equation (2.1b), rather than in Eqn.(2.3), corrects p^* such that it will cancel the term when the momentum equations are integrated. When this term is comparable to the ambient pressure, p_0, the governing equation itself is invalid and the compressible form of the equations must be used. In the present study, *a posteriori* analysis shows the divergence is on the order of 10^3/s within the flame, and even when μ_{B}/μ is on the order of 10^3, this term is still on the order of 10, which is far less than the ambient pressure, and therefore, it seems this is a reasonable formulation for the current problem.

The species and temperature transport equations used here assume the Fickian diffusion model with mixture-averaged transport properties. No Soret, Dufour, or pressure-gradient diffusion effects are considered. Implications of these assumptions are discussed in Appendix A.

These equations seemingly overspecify the density field. There is an evolution equation, Eqn.(2.1a), also the equation of state, Eqn.(A.28e). On the other hand, there is no equation for the pressure (p^*). To resolve this imbalance of variables and equations, the pressure Poisson equation (PPE) is derived and employed. The derivation of PPE depends upon the discretization of the momentum equations and will be discussed later in the chapter.

In the present study, these unsteady equations are integrated until the numerical solution reaches steady state.

In addition to the governing equations for reacting flows at low Mach numbers, two additional models are used in this study. One is a model for an axisymmetric uniform density flow to study non-reacting flows, and the other is a one-dimensional model of stagnation flames developed by Kee *et al.* (1988).

2.2.2 Axisymmetric incompressible uniform-density flow

With the additional assumption of uniform density, the following equations can be derived from the above low Mach number equations. The governing equations for uniform-density flow are the Navier-Stokes equations with a divergence-free constraint on the velocity field, and those are the equations for which the spectral element method was originally developed by Patera (1984). These equations are later used to analyze cold (non-reacting) flows to evaluate discrepancies between simulations and experiments when no chemical reactions are involved:

$$\nabla \cdot \mathbf{u} = 0, \tag{2.4a}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{N}(\mathbf{u}) - \nabla \tilde{p} + \nu \mathbf{L}(\mathbf{u}). \tag{2.4b}$$

Here, $\mathbf{L}$ is a linear operator ∇^2, and $\mathbf{N}$ is a non-linear operator whose form is given later. In the momentum equation, $\tilde{p}$ denotes the scaled perturbational pressure, p^*, divided by density, and is frequently referred to as the kinematic pressure.

One of the difficulties in solving these equations numerically is the divergence-free constraint, as discussed in Gresho (1991). In the spectral element method, this constraint is satisfied through pressure projection along with a splitting method (Ianenko, 1971). Karniadakis *et al.* (1991) derived a high-order splitting method that preserves high-order temporal accuracy. However, more importantly, their splitting method is consistent with the divergence-free constraint and is used in this study as well.

2.2.3 Axisymmetric one-dimensional model for reacting flow

In an important contribution, Kee *et al.* (1988) proposed a further simplification to axisymmetric stagnation-flow modeling. In addition to the symmetry, they assumed a particular form of the streamfunction, $\Psi(x, r) = r^2 U(x)$, which is the leading order of the Taylor series of an arbitrary axisymmetric streamfunction. The third assumption is a constant pressure eigenvalue, *i.e.*, $1/r(\partial p/\partial r)$ is assumed constant throughout the domain. The first two assumptions effectively separate the variables in the definition of the velocity field,

$$u(x) = 2U(x)/\rho(x), \tag{2.5}$$

and

$$v(x,r) = -rU'(x)/\rho(x). \tag{2.6}$$

The last assumption makes it possible to decouple the axial momentum equations and the velocity is determined by the radial momentum balance only,

$$\Lambda \equiv \frac{1}{r}\frac{\partial p}{\partial r} = \frac{d}{dx}\left(\frac{2U}{\rho}\frac{dU}{dx}\right) - \frac{3}{\rho}\left(\frac{dU}{dx}\right)^2 - \frac{d}{dx}\left[\mu\frac{d}{dx}\left(\frac{1}{\rho}\frac{dU}{dx}\right)\right]. \tag{2.7}$$

This equation can be obtained by dividing the radial momentum equation, Eqn.(A.47), by r, and using the Stokes hypothesis ($\mu_{\mathrm{B}} = 0$). Let V be a spreading rate, dv/dr, which is related to the axial velocity gradient through the continuity equation. The above equation becomes

$$V = -\frac{1}{2\rho}\frac{d(\rho u)}{dx}, \tag{2.8a}$$

$$\Lambda = -\frac{d}{dx}(\rho u V) - 3\rho V^2 + \frac{d}{dx}\left(\mu\frac{dV}{dx}\right). \tag{2.8b}$$

These equations, along with four boundary conditions, typically u and V specified at both ends, yield a one-dimensional solution of the stagnation flame. The axial momentum equation may be used to recover the pressure field from the velocity field if desired:

$$\frac{\partial p}{\partial x} = -4U\frac{d}{dx}\left(\frac{U}{\rho}\right) - 2\mu\frac{d}{dx}\left(\frac{1}{\rho}\frac{dU}{dx}\right) + \frac{4}{3}\frac{d}{dx}\left[2\mu\frac{d}{dx}\left(\frac{U}{\rho}\right) + \frac{\mu}{\rho}\frac{dU}{dx}\right]. \tag{2.9}$$

This equation, along with the temperature and the species balance equations along the axis ($\partial/\partial r = 0$ due to axisymmetry), comprise the one-dimensional model. This formulation is relatively simple, and there are ready-to-use codes available such as CHEMKIN or more recently, Cantera (Goodwin, 2003) (both of which implement this model). It has been widely used in the combustion community partly because there have been no practical alternatives in studying combustion problems numerically, since combustion simulations are computer intensive, and inclusion of detailed chemical-kinetics models has been prohibitive in multidimensional simulations in the past.

Although widely used, there is a difficulty with this model. Quoting from the original paper right after Eqn.(6),

$$\frac{\partial}{\partial x}\left(\frac{1}{r}\frac{\partial p}{\partial r}\right) = \frac{1}{r}\frac{\partial}{\partial r}\left(\frac{\partial p}{\partial x}\right) = 0.$$

This is the logic behind assuming that the pressure eigenvalue, $\Lambda = 1/r(\partial p/\partial r)$, is constant. However, the last equality does not hold in general. Since

$$\frac{\partial}{\partial r}\left(\frac{\partial p}{\partial x}\right) \to 0 \tag{2.10}$$

as $r \to 0$, the entire fraction may be finite. When the streamfunction is given by the assumed form, $\partial/\partial x$ is only a function of x as given in Eqn. (2.9), and indeed Λ is a constant. Therefore, how well the one-dimensional model works depends on how well the assumed form of streamfunction is satisfied in a real flow. The consequence of this simplification will be discussed later.

In the present study, the Cantera software package (Goodwin, 2003) is used to solve the above one-dimensional model.

2.2.4 The chemical reaction model

In addition to flow modeling, chemistry source terms and the transport properties that appear in the governing equation must be modeled and evaluated. This part follows practice used in CHEMKIN / Cantera, with more details in Kee *et al.* (2003).

The source term, $\dot{\omega}_{\mathfrak{m}}$, in the species transport equations, Eqn.(2.1c), represents the creation and destruction of a particular species during the combustion process. Suppose there are K reactions and M species. The k-th reaction can be described by

$$\sum_{\mathfrak{m}\in\mathcal{M}} \nu'_{\mathfrak{m}k}\mathfrak{m} \Leftrightarrow \sum_{\mathfrak{m}\in\mathcal{M}} \nu''_{\mathfrak{m}k}\mathfrak{m} \tag{2.11}$$

where $\nu_{\mathfrak{m}k}$ denotes stoichiometric coefficients of species $\mathfrak{m}$ in reaction k, and the set $\mathcal{M}$ contains all species relating to the given reaction system ($M = |\mathcal{M}|$). The superscript (r) denotes reactants and (p) denotes products.

Then the rate-of-progress variable of the k-th reaction, q_k, is

$$q_k = k_{fk}\prod_{\mathfrak{m}\in\mathcal{M}}[\mathfrak{m}]^{\nu'_{\mathfrak{m}k}} - k_{rk}\prod_{\mathfrak{m}\in\mathcal{M}}[\mathfrak{m}]^{\nu''_{\mathfrak{m}k}} \tag{2.12}$$

where k_{fk} and k_{rk} are the forward and backward reaction rates, and are usually expressed in Arrhenius forms,

$$k_f = AT^n \exp\left(-E_A/R_{\mathrm{u}}T\right), \tag{2.13}$$

with $[\mathfrak{m}]$ the molar concentration of species $\mathfrak{m}$. The reaction index k is suppressed from this expression for conciseness, but parameters A, n, and E_A are all reaction-dependent. Each one of these parameters is usually supplied by reaction models such as GRI-Mech 3.0. Although the backward reaction rate may be obtained by the same formula, it is more accurate to use the equilibrium constant K_c to obtain K_r through (*e.g.*, Denbigh, 1955; Turns, 2000)

$$K_c = \frac{k_f}{k_r}, \tag{2.14}$$

where

$$K_c = K_p \left(\frac{p_{\mathrm{ref}}}{R_{\mathrm{u}}T}\right)^s, \tag{2.15}$$

where $s = \sum \nu^{(p)} - \sum \nu^{(r)}$, and

$$K_p = \exp\left(\sum_{\mathfrak{m}\in\mathcal{M}} \left[\nu_{\mathfrak{m}}^{(r)} \frac{g_{\mathfrak{m}}}{R_{\mathrm{u}}T} - \nu_{\mathfrak{m}}^{(p)} \frac{g_{\mathfrak{m}}}{R_{\mathrm{u}}T}\right]\right). \tag{2.16}$$

The molar production rate is

$$\dot{\omega}_{\mathfrak{m}}^c = \sum_{k=1}^{K} \left(\nu_{\mathfrak{m}k}^{(p)} - \nu_{\mathfrak{m}k}^{(r)}\right) q_k. \tag{2.17}$$

The mass fraction production rate can be obtained by

$$\dot{\omega}_{\mathfrak{m}} = \frac{W_{\mathfrak{m}}}{\rho} \dot{\omega}_{\mathfrak{m}}^c. \tag{2.18}$$

2.2.5 The mixture-averaged transport model

In addition to a chemical-kinetics model, transport properties that appear in the equations must be modeled and evaluated. Models used in the present study are described here for completeness, and more details can be found in Kee *et al.* (1986). The Wilke formula is

used to evaluate the viscosity of the given gas mixture,

$$\mu = \sum_{\mathfrak{m}\in\mathcal{M}} \frac{X_{\mathfrak{m}}\mu_{\mathfrak{m}}}{\sum_{\mathfrak{n}} X_{\mathfrak{n}}\Phi_{\mathfrak{m}\mathfrak{n}}}, \tag{2.19}$$

where

$$\Phi_{\mathfrak{m}\mathfrak{n}} = \frac{1}{\sqrt{8}}\left(1+\frac{W_{\mathfrak{m}}}{W_{\mathfrak{n}}}\right)^{-1/2}\left(1+\left(\frac{\mu_{\mathfrak{m}}}{\mu_{\mathfrak{n}}}\right)^{1/2}\left(\frac{W_{\mathfrak{n}}}{W_{\mathfrak{m}}}\right)^{1/4}\right)^{2}. \tag{2.20}$$

$X_{\mathfrak{m}}$ is the mole fraction of species $\mathfrak{m}$, $\mu_{\mathfrak{m}}$ is the dynamic viscosity of species $\mathfrak{m}$, which is given by kinetic theory,

$$\mu_{\mathfrak{m}} = \frac{5}{16}\frac{\sqrt{\pi m_{\mathfrak{m}} k_B}}{\pi\sigma_{\mathfrak{m}}^2\Omega^{(2,2)*}}, \tag{2.21}$$

where $\sigma_{\mathfrak{m}}$ is the Lennard-Jones collision diameter, $m_{\mathfrak{m}}$ is the molecular mass ($W_{\mathfrak{m}}/N_A$ where N_A is the Avogadro number), and k_B is the Boltzmann constant. The collision integral $\Omega^{(2,2)*}$ values are evaluated through the tables given in Monchick & Mason (1961).

The thermal conductivity of a gas mixture is computed using a combination averaging formula:

$$\lambda_T = \frac{1}{2}\left(\sum_m X_{\mathfrak{m}}\lambda_{T\mathfrak{m}} + \frac{1}{\sum_m X_{\mathfrak{m}}/\lambda_{T\mathfrak{m}}}\right). \tag{2.22}$$

The thermal conductivity of each individual species can be obtained by

$$\lambda_{T\mathfrak{m}} = \frac{\mu_{\mathfrak{m}}}{W_{\mathfrak{m}}}\left(f_{\text{trans}}C_{v,\text{trans}} + f_{\text{rot}}C_{v,\text{rot}} + f_{\text{vib}}C_{v,\text{vib}}\right), \tag{2.23}$$

where

$$f_{\text{trans}} = \frac{5}{2}\left(1-\frac{2}{\pi}\frac{C_{v,\text{rot}}}{C_{v,\text{trans}}}\frac{A}{B}\right), \tag{2.24}$$

$$f_{\text{rot}} = \frac{\rho D_{mm}}{\mu_{\mathfrak{m}}}\left(1+\frac{2}{\pi}\frac{A}{B}\right), \tag{2.25}$$

and

$$f_{\text{vib}} = \frac{\rho D_{mm}}{\mu_{\mathfrak{m}}}. \tag{2.26}$$

The constants that appear in these expressions are

$$A = \frac{5}{2} - f_{\text{vib}}, \tag{2.27}$$

Table 2.1: The values of C_v

	monatomic	linear	nonlinear
$C_{v,\mathrm{trans}}/R_\mathrm{u}$	3/2	3/2	3/2
$C_{v,\mathrm{rot}}/R_\mathrm{u}$	0	1	3/2
$C_{v,\mathrm{vib}}/R_\mathrm{u}$	0	$Cv/R_\mathrm{u} - 5/2$	$Cv/R_\mathrm{u} - 3$

and

$$B = Z_\mathrm{rot} + \frac{2}{\pi}\left(\frac{5}{3}\frac{C_{v,\mathrm{rot}}}{R_\mathrm{u}} + f_\mathrm{vib}\right), \tag{2.28}$$

where Z_rot is the rotational relaxation collision number, and its value at 298 K is given as an input.

The molar heat capacities in Eqn.(2.23) depend on the geometry of molecules and are collected in Table 2.1.

The self-diffusion coefficient is given by,

$$D_{\mathfrak{m}\mathfrak{m}} = \frac{3}{16}\frac{\sqrt{2\pi k_B^3 T^3/m_\mathfrak{m}}}{p_0 \pi \sigma_k^2 \Omega^{(1,1)*}}. \tag{2.29}$$

The mixture-averaged diffusion coefficient is given by Mathur *et al.* (1967) (see Kee *et al.*, 1986)

$$D_\mathfrak{m} = \frac{\bar{W} - X_\mathfrak{m} W_\mathfrak{m}}{\bar{W}\sum_{\mathfrak{n}\neq\mathfrak{m}} X_\mathfrak{n}/D_{\mathfrak{m}\mathfrak{n}}}, \tag{2.30}$$

where $D_{\mathfrak{m}\mathfrak{n}}$ is the binary diffusion coefficient between species $\mathfrak{m}$ and $\mathfrak{n}$,

$$D_{\mathfrak{m}\mathfrak{n}} = \frac{3}{16}\frac{\sqrt{2\pi k_B^3 T^3/m_{\mathfrak{m}\mathfrak{n}}}}{p\pi\sigma_{\mathfrak{m}\mathfrak{n}}\Omega^{(1,1)*}}, \tag{2.31}$$

where $m_{\mathfrak{m}\mathfrak{n}}$ is the reduced molar mass for the species pair and is given by

$$m_{\mathfrak{m}\mathfrak{n}} = \frac{m_\mathfrak{m} m_\mathfrak{n}}{m_\mathfrak{m} + m_\mathfrak{n}}. \tag{2.32}$$

2.3 The numerical method

2.3.1 Introduction

There are many methods for solving partial differential equations numerically. However, for the purpose of this study, the requirements are: flexibility to handle complex geometry as set up in a laboratory, efficiency to make it possible to integrate computationally expensive reacting flow equations, and accuracy to manage numerical error so that the errors in the chemical kinetics models can be evaluated and assessed. The spectral element method, originally proposed by Patera (1984) for incompressible flows (*e.g.*, Henderson & Karniadakis, 1995; Henderson & Barkley, 1996; Henderson, 1999a,b; Blackburn & Henderson, 1999; Tomboulides & Orszag, 2000; Blackburn & Lopez, 2002), can be adapted to satisfy all these requirements after its applicability is extended to accommodate flows with large density variations. Although the original method was intended for an incompressible flow, Tomboulides *et al.* (1997) and Tomboulides & Orszag (1998) later extended it to variable-density low Mach number flow. However their approach is still not sufficiently efficient to simulate methane flames with detailed combustion and transport models because their algorithm solves the reaction-diffusion equations without operator splitting. A new algorithm has been developed and is presented here.

2.3.2 Expansion basis

In the spectral element method for two-dimensional problems, the approximate solution is expanded in a given expansion basis as follows,

$$u(x,y) = \sum_{i,j} u_{i,j} h_i(x) h_j(y), \tag{2.33}$$

where $h_i(x)$ is a compactly supported Lagrange polynomial based on the roots of Jacobi polynomials and has the following properties:

$$h_i(x_j) = \delta_{ij}, \tag{2.34}$$

$$\int_{\Omega^e} u(x)dx \approx \sum_i \omega_i u(x_i), \tag{2.35}$$

and

$$\int_{\Omega^e} h_i(x)h_j(x)dx \approx \begin{cases} \omega_i & i = j \\ 0 & i \neq j \end{cases}, \tag{2.36}$$

where Ω^e is a finite size domain over which the basis functions, $h_i(x)$, have support.

The basis function $h_i(x)$ can be constructed using many different orthogonal polynomials, but Legendre polynomials with Gauss-Lobatto quadrature are most often used (Henderson & Karniadakis, 1995; Blackburn & Sherwin, 2004), and they will also be used in this study (hereafter called GLL—Gauss-Lobatto-Legendre—basis) except for the expansion in the radial direction within elements that are adjacent to the axis. For this special case, a new basis function has been developed and will be described next.

2.3.3 Polar axis treatment for axisymmetric flow

Simulation of fluid flow in an axisymmetric domain requires a proper treatment of the axis singularity. There are essentially two issues: One is that the governing equation itself contains a singularity when written in polar-coordinate form, and the other is that there are certain requirements in the behavior of each azimuthal Fourier mode, and those requirements must be satisfied by each expansion basis function. These issues are addressed in the context of the spectral method by several authors (Leonard & Wray, 1982; Matsushima & Marcus, 1995; Mohseni & Colonius, 2000) (see Boyd, 2000), but the second issue has been mostly ignored in the spectral element method (Tomboulides *et al.*, 1997; Blackburn & Sherwin, 2004). In those studies that employed the spectral element method in axisymmetric coordinates, only the leading order behavior is specified. Higher-order conditions are ignored with the expectation that they will be satisfied in the process of convergence. However, although there are cases in which high-order conditions can be satisfied without specifying them, there are cases in which the smooth numerical solution does not satisfy some particular required property. To this end, a new basis function that incorporates the correct behavior has been developed.

An arbitrary function in polar coordinates can be expanded as a Fourier series in θ,

$$f(r,\theta) = \sum_{m=0}^{\infty} \left(f_m(r)\cos(m\theta) + g_m(r)\sin(m\theta) \right). \tag{2.37}$$

If $f(r,\theta)$ is a scalar and the function is analytic at $r = 0$, the following conditions must be

satisfied (Boyd, 2000):

(i) $f_m(r)$ and $g_m(r)$ have m-th order zeros at $r = 0$.

(ii) If m is even, then $f_m(r)$ and $g_m(r)$ are both symmetric about $r = 0$ and their power series contain only even powers of r.

(iii) If m is odd, then $f_m(r)$ and $g_m(r)$ are both antisymmetric about $r = 0$ and their power series contain only odd powers of r.

When $f(r, \theta)$ is the axial velocity in the cylindrical coordinates or is the product of r with the radial or tangential velocity in polar coordinates, the same conditions apply. For axisymmetric problems, as considered in this paper, the requirement translates to the fact that scalars and axial velocity must be even functions about the axis, while radial velocity must be an odd function about the axis, and we need to implement this property in the basis function itself.

In the standard coordinate, $\xi \in [-1, 1]$, mapping to the physical coordinate is provided by isoparametric mapping (Karniadakis & Sherwin, 1999). The approximate solution is expanded in the following form:

$$f(\xi_z, \xi_r) = \sum_{i=0}^{Q-1} \sum_{j=0}^{Q-1} \hat{f}_{ij} h_i(\xi_z) h_j^r(\xi_r) \tag{2.38}$$

where $h_i(z)$ are Lagrange polynomials of order P $(= Q - 1)$. For the expansion of all elements in the axial direction and for the expansion in the radial direction (except for those elements adjacent to the axis), GLL collocation points are used to construct the Lagrange polynomials such that $h_m(\xi_n) = \delta_{mn}$, where $\xi_m (m = 0, 1, \ldots, Q - 1)$ are a set of Q GLL points. This is a standard basis in the spectral element method and further details are given in Karniadakis & Sherwin (1999). For $h_m^r(\xi)$, depending on the parity requirements for the approximate solution, either $h_m^{\text{even}}(\xi)$ or $h_m^{\text{odd}}(\xi)$ are used—which are even and odd functions of ξ, respectively—or $h_m(\xi)$ is used if there is no parity requirement.

For the radial expansion of elements adjacent to the axis, let $\eta = [(\xi + 1)/2]^{1/2} \in [0, 1]$. Following Leonard & Wray (1982), and Matsushima & Marcus (1995), a function with even parity about $\eta = 0$ can be expressed as

$$h_m^{\text{even}}(\eta) = h_m(2\eta^2 - 1). \tag{2.39}$$

Leonard & Wray (1982) used a quadratic basis function to obtain the desired parity and the divergence-free constraint in their numerical study using a spectral method. Later, Matsushima & Marcus (1995) derived a more general form of basis functions with parity that can represent any non-symmetric smooth function with the required parity property. The basis function used by Leonard & Wray (1982) is a special case of the one derived by Matsushima & Marcus (1995).

Effectively, we construct a Lagrange polynomial using quadratic monomials (x^2) to obtain a desired parity, and a required matching condition at the outer boundary. However, this is still a GLL approximation in the original polynomial space, and we can achieve spectral accuracy, as shown later. Since $\{L_m(x)\}_{m=0,\ldots,Q-1}$ is a complete basis for a space of polynomials of order P, this basis function spans a space of even polynomials of order $2P$. The new basis functions are not orthogonal analytically (so are the original GLL polynomials), but they both satisfy the discrete orthogonality conditions.

The basis function $h_m(\eta)$ is an even function in η and $h_m(\eta_n) = \delta_{mn}$ so that the discrete orthogonality condition is satisfied.

For the odd function basis, one may use $h_m^{\text{even}}(\eta)$ as a building block, and then

$$h_m^{\text{odd}}(\eta) = \frac{\eta}{\eta_m} h_m^{\text{even}}(\eta) \tag{2.40}$$

satisfies all the requirements.

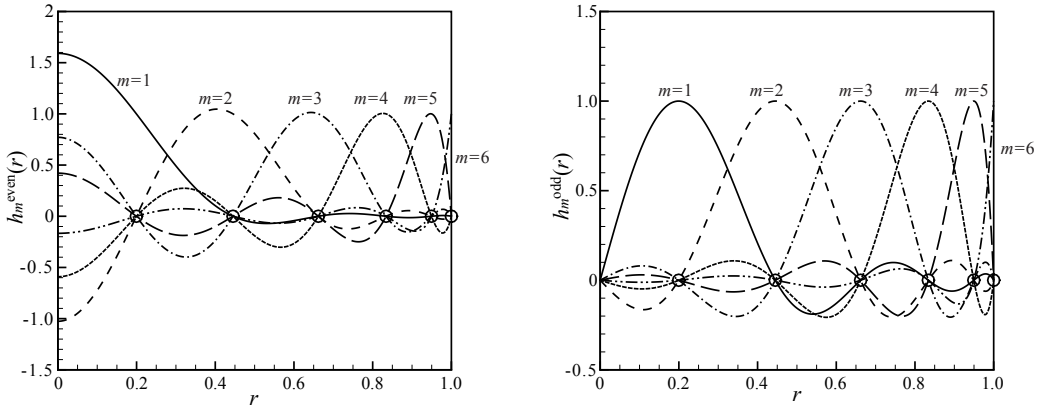
Simple arithmetic shows that the collocation derivative matrices for these bases are

$$D_{ij}^{\text{even}} = 4\frac{r_i}{R^2} D_{ij}, \tag{2.41}$$

and

$$D_{ij}^{\text{odd}} = \frac{1}{r_i}\delta_{ij} + \frac{r_i}{r_j} D_{ij}^{\text{even}} \tag{2.42}$$

where D_{ij} is the collocation differentiation matrix for the original basis, and R is the length of the radial domain in physical space. The even and odd basis functions can be constructed through Eqns.(2.39) and (2.40) once the base basis function, h_m in Eqn.(2.39) is selected. The Gauss-Radau-Legendre (GRL) polynomial has the desired property of containing only one boundary collocation point so that it allows connectivity conditions to be incorporated at the outer element boundary while it spans all polynomial space less than the given



(a) Even function basis ($Q = 6$)

(b) Odd function basis ($Q = 6$)

Figure 2.1: The radial expansion bases for elements adjacent to the axis. Note that each basis function satisfies the correct parity requirements. Each basis function consists of up to $(2Q-1)$-th order polynomials and satisfies the discrete orthogonality property.

polynomial order. In addition, the GRL basis conveniently avoids evaluating singular terms in the governing equation by avoiding a collocation point on the axis.

The Gaussian quadrature weights are $w_i^{\text{even}} = w^{\text{odd}} = (R^2/4)w_i$. These new bases are shown in Fig. 2.1, for the case of $Q = 6$, and they will be referred to as Gauss-Radau-Legendre with parity (GRLp) bases in this study. As can be seen in the figure, the collocation points associated with this basis function have fewer clustered collocation points toward the axis, where the function expanded is expected to be smooth. Therefore, it is slightly computatinally advantageous compared to other basis functions used in earlier works on spectral element method for axisymmetric problems (Tomboulides *et al.*, 1997; Blackburn & Sherwin, 2004).

$$
\begin{aligned}
&\int_0^R f^{\text{even}}(r) r dr \\
&= R^2 \int_0^1 \sum_m f_m h_m^{\text{even}}(\eta) \eta d\eta \\
&= R^2 \sum_m f_m \int_0^1 h_m^{\text{even}}(\eta) \eta d\eta \\
&= \frac{R^2}{4} \sum_m f_m \int_{-1}^1 h_m(\xi) d\xi \\
&= \frac{R^2}{4} \sum_m f_m \left(\sum_n w_n h_m(\xi_n) \right) \\
&= \frac{R^2}{4} \sum_m f_m w_m \\
&= \sum_m f_m w_m^{\text{even}}.
\end{aligned}
$$

The same is true for the odd basis:

$$
\begin{aligned}
&\int_0^R f^{\text{odd}}(r) r dr \\
&= R^2 \int_0^1 \sum_m f_m h_m^{\text{odd}}(\eta) \eta d\eta \\
&= R^2 \sum_m f_m \int_0^1 h_m^{\text{odd}}(\eta) \eta d\eta \\
&= R^2 \sum_m f_m \int_0^1 \frac{\eta}{\eta_m} h_m^{\text{even}}(\eta) \eta d\eta \\
&= \frac{R^2}{4} \sum_m f_m \int_{-1}^1 \frac{\eta(\xi)}{\eta_m} h_m(\xi) d\xi \\
&= \frac{R^2}{4} \sum_m f_m \left(\sum_n w_n \frac{\eta_n}{\eta_m} h_m(\xi_n) \right) \\
&= \frac{R^2}{4} \sum_m f_m w_m \\
&= \sum_m f_m w_m^{\text{even}}.
\end{aligned}
$$

Development of basis functions with appropriate parity and desired behavior for the non-axisymmetric case, *i.e.*, $m > 0$, near the axis is a straightforward extension of the approach described here and the work of Matsushima & Marcus (1995), using one-sided

Table 2.2: The mixed stiffly stable scheme coefficients (Karniadakis *et al.*, 1991)

Coefficient	γ_0	α_0	α_1	α_2	β_0	β_1	β_2
1st order	1	1	0	0	1	0	0
2nd order	3/2	2	-1/2	0	2	-1	0
3rd order	11/6	3	-3/2	1/3	3	-3	1

Jacobi polynomials.

2.3.4 The spectral element method for uniform-density flows

We first introduce the spectral element method for uniform-density flow, as we will use the same building blocks later to construct a method for reacting flows.

First, the momentum equations are discretized in time using a mixed stiffly stable scheme (Karniadakis *et al.*, 1991). The idea of a mixed stiffly stable scheme is to use an extrapolation to estimate the implicit flux of the nonlinear term in the context of a stable backward-differentiation scheme.

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \sum_{q=0}^{J_e-1} \alpha_q \mathbf{u}^{n-q}}{\Delta t} = \sum_{q=0}^{J_e-1} \beta_q \mathbf{N}(\mathbf{u}^{n-q}) - \nabla p^{n+1} + \nu \mathbf{L}(\mathbf{u}^{n+1}) \tag{2.43}$$

where $\mathbf{N}(\mathbf{u}) = -\mathbf{u} \cdot \nabla \mathbf{u}$, and $\mathbf{L}(\mathbf{u}) = \nabla^2 \mathbf{u}$. In the above equations, α_q, β_q, and γ_0 are the weighting coefficients for stiffly stable time-integration method of order J_e, and the values are tabulated in Table 2.2.

Splitting this equation into terms gives,

$$\widehat{\mathbf{u}} - \sum_{q=0}^{J_e-1} \alpha_q \mathbf{u}^{n-q} = -\Delta t \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{u} \cdot \nabla \mathbf{u}\right)^{n-q} \tag{2.44a}$$

$$\widehat{\widehat{\mathbf{u}}} - \widehat{\mathbf{u}} = -\Delta t \nabla p^{n+1} \tag{2.44b}$$

$$\gamma_0 \mathbf{u}^{n+1} - \widehat{\widehat{\mathbf{u}}} = \nu \Delta t \nabla^2 \mathbf{u}^{n+1}, \tag{2.44c}$$

where $\widehat{\mathbf{u}}$ and $\widehat{\widehat{\mathbf{u}}}$ are intermediate velocities in the time-stepping scheme, defined by Eqns.(2.44a) and (2.44b), respectively.

By taking the divergence of the equation for the pressure projection, Eqn. (2.44b), the pressure Poisson equation is obtained. Note that $\nabla \cdot \widehat{\widehat{\mathbf{u}}} = 0$ by taking the divergence of the

last equation. Finally, the governing equations are integrated by the following four steps:

$$\widehat{\mathbf{u}} - \sum_{q=0}^{J_e-1} \alpha_q \mathbf{u}^{n-q} = -\Delta t \sum_{q=0}^{J_e-1} \beta_q \left(\mathbf{u} \cdot \nabla \mathbf{u}\right)^{n-q} \tag{2.45a}$$

$$\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \widehat{\mathbf{u}} \tag{2.45b}$$

$$\widehat{\widehat{\mathbf{u}}} - \widehat{\mathbf{u}} = -\nabla p^{n+1} \Delta t \tag{2.45c}$$

$$\gamma_0 \mathbf{u}^{n+1} - \widehat{\widehat{\mathbf{u}}} = \nu \Delta t \nabla^2 \mathbf{u}^{n+1}. \tag{2.45d}$$

It may be observed that the building blocks for the spectral element method for incompressible uniform density flows are the explicit advection equation,

$$\frac{\partial u}{\partial t} = f(u), \tag{2.46}$$

and the implicit Helmholtz equation,

$$\nabla^2 u - \lambda u = f(u). \tag{2.47}$$

The former is solved by an explicit collocation method while the latter is solved by the Galerkin method.

2.3.5 The spectral element method for variable-density flows

When extending the spectral element method to the chemically reacting flows with large density variations, additional difficulties arise.

First, unlike in the compressible counterpart, pressure must be obtained by deriving and solving the pressure Poisson equation rather than the equation of state. This is because pressure in the equation of state is not the same as that in the momentum equation. Second, when solving the compressible equations explicitly, the continuity equation can be omitted in favor of the species-conservation equations and the constraint, $\sum_{\mathfrak{m}} Y_{\mathfrak{m}} = 1$. Then the density field can be obtained simply by adding partial densities, *i.e.*, $\rho = \sum_{\mathfrak{m}} \rho_{\mathfrak{m}}$. Again, this is not possible in the low Mach number case as the continuity equation is needed to derive the pressure Poisson equation, and therefore, one of the species conservation equations is redundant and must be abandoned. Third, the equation of state is now a constraint between

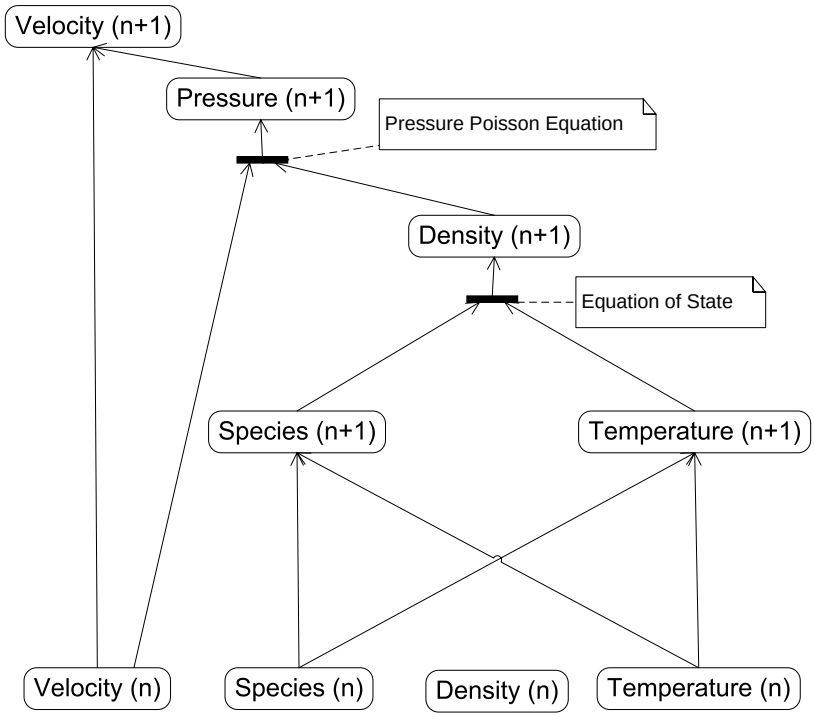


Figure 2.2: Integration dependency for low Mach number formulation

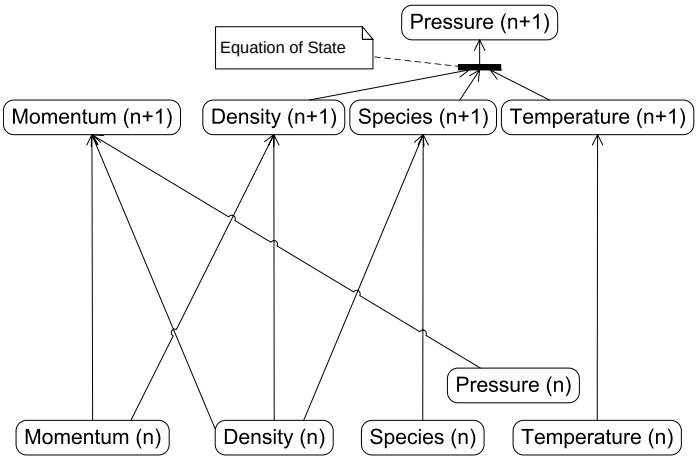


Figure 2.3: Integration dependency for explicit compressible formulation

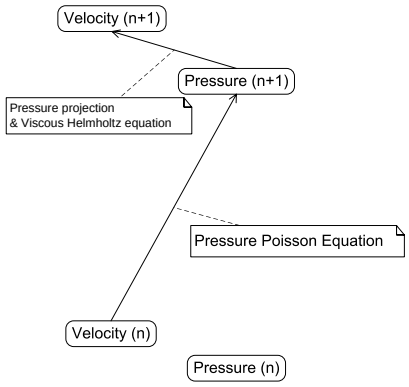


Figure 2.4: Integration dependency for uniform density formulation

density, compositions through the specific gas constant, and temperature—and among these unknowns, only density does not have a formula that updates it in time. Therefore, the equation of state must be used to specify the density field, given the temperature and mass fractions. This is a major difference from the work of Day & Bell (2000), in which the continuity equation was used to update density while the equation of state is incorporated

in the divergence constraint as a penalty barrier.

The updating scheme is summarized in Fig. 2.2. For comparison, the same dependency diagrams for a compressible formulation (Fig. 2.3) and a uniform-density formulation (Fig. 2.4) are shown here as well. As can be seen from this diagram, a density update must be obtained before integrating the momentum equations. Therefore, temperature and species must be updated first.

In addition, the left hand side operator of the pressure Poisson equation now contains the reciprocal of the density, $1/\rho$, and a technique for solving such equations must be developed. One could choose not to include this $1/\rho$ factor inside the operator by multiplying through ρ to the momentum equations before taking the divergence of the entire equation. However, this formulation limits the size of the timestep, Δt, compared to the current formulation and was not used in this study.

The discrete form of the governing equations is introduced in the order in which they are updated:

$$\frac{\gamma_0 T^{n+1} - \sum_{q=0}^{J_e-1} \alpha_q T^{n-q}}{\Delta t} = \sum_{q=0}^{J_e-1} \beta_q \left(-\mathbf{u}\cdot\nabla T + \left(\frac{1}{\rho C_{p,\mathrm{mix}}}\right)\nabla\lambda_T\cdot\nabla T\right)^{n-q} + \alpha^n \nabla^2 T^{n+1} \tag{2.48}$$

$$\frac{\gamma_0 Y_{\mathfrak{m}}^{n+1} - \sum_{q=0}^{J_e-1} \alpha_q Y_{\mathfrak{m}}^{n-q}}{\Delta t} = \sum_{q=0}^{J_e-1} \beta_q \left(-\mathbf{u}\cdot\nabla Y_{\mathfrak{m}} + \frac{1}{\rho^n}\nabla\rho D_{\mathfrak{m}}\cdot\nabla Y_{\mathfrak{m}}\right)^{n-q} + D_{\mathfrak{m}}^n \nabla^2 Y_{\mathfrak{m}}^{n+1} \tag{2.49}$$

for $\mathfrak{m} \in \mathcal{M}'$ where $\mathcal{M}'$ contains all species in $\mathcal{M}$ except N_2. Then,

$$Y_{N_2} = 1 - \sum_{\mathfrak{m}\in\mathcal{M}'} Y_{\mathfrak{m}}, \tag{2.50}$$

and

$$\rho^{n+1} = \frac{p_0}{R_u T^{n+1}}\bar{W} = \frac{p_0}{R_u T^{n+1}} \frac{1}{\sum_{\mathfrak{m}\in\mathcal{M}}(Y_{\mathfrak{m}}^{n+1}/W_{\mathfrak{m}})}. \tag{2.51}$$

Once the updated density, temperature, and mass fractions are obtained, the rest follows the procedure for incompressible non-uniform density flows, which will be described next. Now that species and temperature fields are updated, transport properties can be updated

at this time, *i.e.*,

$$\mu^{n+1} = \mu(Y_{\mathfrak{m}}^{n+1}, T^{n+1}) \tag{2.52a}$$

$$\lambda_T^{n+1} = \lambda_T(Y_{\mathfrak{m}}^{n+1}, T^{n+1}) \tag{2.52b}$$

$$D_{\mathfrak{m}}^{n+1} = D_{\mathfrak{m}}(Y_{\mathfrak{m}}^{n+1}, T^{n+1}, p_0) \tag{2.52c}$$

Once all thermodynamic states at the new time-level are obtained, the momentum equations are finally integrated,

$$\widehat{\mathbf{u}} - \sum_{q=0}^{J_e-1} \alpha_q \mathbf{u}^{n-q} = \Delta t \sum_{q=0}^{J_e-1} \alpha_q \left(-\mathbf{u}\cdot\nabla\mathbf{u} + \mathbf{L}^e(\mathbf{u}) + \mathbf{f}\right)^{n-q} \tag{2.53a}$$

$$\nabla\cdot\left(\frac{1}{\rho}\nabla p^*\right) = \frac{1}{\Delta t}\left(\nabla\cdot\widehat{\mathbf{u}} - \nabla\cdot\widehat{\widehat{\mathbf{u}}}\right) \tag{2.53b}$$

$$\widehat{\widehat{\mathbf{u}}} - \widehat{\mathbf{u}} = -\Delta t\,\nabla p^* \tag{2.53c}$$

$$\gamma_0\mathbf{u}^{n+1} - \widehat{\widehat{\mathbf{u}}} = \frac{\mu}{\rho}\Delta t\,\nabla^2\mathbf{u}^{n+1}, \tag{2.53d}$$

where, again as is in the uniform-density equations, $\widehat{\mathbf{u}}$ and $\widehat{\widehat{\mathbf{u}}}$ are the intermediate velocities in the time-stepping scheme, defined by Eqns.(2.53a) and (2.53c), respectively.

In the above expressions, J_e in summations is the order of the time-integration scheme, and appropriate coefficients (α_q, β_q, and γ_0) for each case are recorded in Table 2.2. In contrast with the uniform-density case, the second term on the right-hand side of the pressure Poisson equation does not disappear. Now, to obtain the unknown quantity that appears in the right hand side of Eqn.(2.53b), we use Eqn.(2.53d),

$$\nabla\cdot\widehat{\widehat{\mathbf{u}}} = \gamma_0 Q^{n+1} - \Delta t\nabla\cdot\left[\frac{\mu}{\rho}\Big(\nabla\left(\nabla\cdot\mathbf{u}\right) - \nabla\times\nabla\times\mathbf{u}\Big)\right], \tag{2.54}$$

where Q^{n+1} is the divergence constraint that must be satisfied at time t^{n+1}, which must be obtained through the continuity equation,

$$Q^{n+1} \equiv \nabla\cdot\mathbf{u}^{n+1} = -\frac{1}{\rho}\frac{D\rho}{Dt} = -\frac{1}{\rho}\left(\frac{\gamma_0\rho^{n+1} - \sum_{q=0}^{J_e-1}\alpha_q\rho^{n-q}}{\Delta t} + \sum_{q=0}^{J_e-1}\beta_q(\mathbf{u}\cdot\nabla\rho)^{n-q}\right). \tag{2.55}$$

Other studies (Tomboulides *et al.*, 1997; Day & Bell, 2000) used the Lagrange derivative of the equation of state to decompose $D\rho/Dt$ into DT/Dt and $DY_{\mathfrak{m}}/Dt$, and used the tem-

perature transport equation and the species transport equations to obtain Q^{n+1}. However this approach is costly and accumulates round-off errors.

2.3.6 Boundary condition for the pressure Poisson equation

The pressure Poisson equation requires boundary conditions. Typically, uniform-pressure boundary condition is specified at the outflow where the ambient pressure can be specified. For other boundaries, such as inflow and wall, where the velocity components are specified, the Neumann boundary condition for the pressure field must be derived from the momentum equation. Assuming the steady boundary condition, we have,

$$\frac{1}{\rho}\frac{\partial p^*}{\partial n} = \mathbf{n}\cdot\left(\mathbf{N}(\mathbf{u}) + \mathbf{L}(\mathbf{u})\right), \tag{2.56}$$

where $\mathbf{N}(\mathbf{u}) = -\mathbf{u}\cdot\nabla\mathbf{u}$.

Using the vector identity,

$$\nabla^2\mathbf{u} = \nabla\left(\nabla\cdot\mathbf{u}\right) - \nabla\times\nabla\times\mathbf{u}, \tag{2.57}$$

the viscous term can be rewritten as,

$$\mathbf{L}(\mathbf{u}) = \frac{\mu}{\rho}\left(2\nabla Q - \nabla\times\nabla\times\mathbf{u}\right) + \frac{2}{\rho}\nabla\mu\cdot\mathbf{S}, \tag{2.58}$$

where $\mathbf{S} = (\nabla\mathbf{u} + \nabla\mathbf{u}^T)/2$.

2.3.7 Helmholtz equations

Solving Eqns. (2.48), (2.49), (2.53b), and (2.53d) requires solving the Helmholtz equations. We define the inner-product to describe Galerkin formulation,

$$(\psi, \Phi) = \int_\Omega \psi(\mathbf{x})\Phi(\mathbf{x})d\Omega, \tag{2.59}$$

$$(\psi, \Phi)_w = \int_\Omega \psi(\mathbf{x})\Phi(\mathbf{x})w(\mathbf{x})d\Omega, \tag{2.60}$$

and

$$\langle\psi, \Phi\rangle = \int_{\partial\Omega^{\mathcal{N}}} \psi(\mathbf{x})\nabla\Phi(\mathbf{x})\cdot\mathbf{n}dS. \tag{2.61}$$

2.3.7.1 Cartesian coordinate

In this case,

$$\nabla^2 u - \lambda^2 u = f(u), \tag{2.62}$$

or in differential form,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - \lambda u = f(u). \tag{2.63}$$

We multiply the equation by a test function $v(x, y)$ and integrate over the entire domain to obtain,

$$(\nabla v, \nabla u) + \lambda(v, u) = -(v, f) + \langle v, u \rangle. \tag{2.64}$$

Using the basis functions described earlier, this equation can be cast into the linear system.

$$[\mathbf{L} + \lambda \mathbf{M}]\, \mathbf{u} = -\mathbf{M}\mathbf{f} + \mathbf{u}^{\delta N} \tag{2.65}$$

where

$$L_{ij} = \int_\Omega h_i'(\mathbf{x}) h_j'(\mathbf{x}) d\Omega. \tag{2.66}$$

2.3.7.2 Cylindrical coordinate

Again, we start with the same equation,

$$\nabla^2 u - \lambda^2 u = f(u), \tag{2.67}$$

which in differential form is,

$$\frac{\partial^2 u}{\partial z^2} + \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) - \lambda^2 u = f(u). \tag{2.68}$$

We multiply this equation through by r to obtain (Blackburn & Sherwin, 2004),

$$\frac{\partial}{\partial z}\left(r\frac{\partial u}{\partial z}\right) + \frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) - \lambda^2 r u = r f(u). \tag{2.69}$$

Again we apply the method of weighted residuals with a test function $v(z, r)$,

$$(\nabla v, \nabla u)_r + \lambda^2 (v, u)_r = -(v, f)_r + \langle v, u \rangle_r. \tag{2.70}$$

As it appears in the pressure Poisson equation, an equation with a variable factor inside the operator needs to be solved:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla u \right) - \lambda^2 u = f(u), \tag{2.71}$$

which in differential form is,

$$\frac{\partial}{\partial z} \left(\frac{r}{\rho} \frac{\partial u}{\partial z} \right) + \frac{\partial}{\partial r} \left(\frac{r}{\rho} \frac{\partial u}{\partial r} \right) - \lambda^2 r u = r f(u). \tag{2.72}$$

Again, the factor r has been applied to the entire equation. Applying the method of weighted residuals with a test function $v(z, r)$ then yields,

$$(\nabla v, \nabla u)_{r/\rho} + \lambda^2 (v, u)_r = -(v, f)_r + \langle v, u \rangle_r. \tag{2.73}$$

In the exact Galarkin formulation, the weight term must be evaluated by expansion, *i.e.*, the stiffness matrix should be:

$$\mathbf{L}_{ij} = \int_{\Omega^e} \frac{\phi_i'(x)\phi_j'(x)}{\sum_{l=1}^{Q} \rho_l \phi_l(x)} d\Omega^e.$$

However, using the property of the Lagrange polynomial, we have,

$$\mathbf{L}_{ij} = \sum_{l=1}^{Q} \omega_l \frac{\phi_i'(x_l)\phi_j'(x_l)}{\rho_l}.$$

This form of the stiffness matrix is used in this study.

2.3.8 Solution method for the linear system

Now that a matrix system has been obtained for the viscous Helmholtz equations, or pressure Poisson equations, the linear system may be solved directly, or iteratively. In the spectral element method, the stiffness matrix usually has a localized structure and the static condensation technique is the most efficient technique for solving the equation.

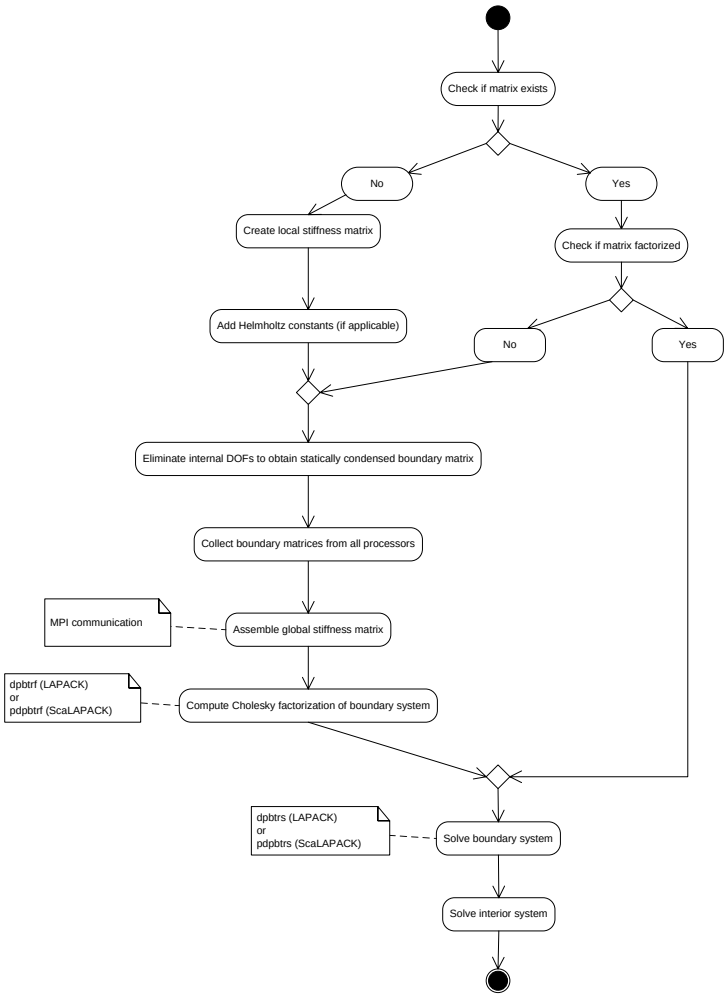


Figure 2.5: The activity diagram (flow chart) of the stiffness matrix construction and solution. Note that the Cholesky factorization is the most expensive process.

Given a matrix equation, $\mathbf{Au} = \mathbf{f}$, decompose it into a block form,

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}. \tag{2.74}$$

By multiplying

$$\begin{bmatrix} I & -A_{12}A_{22}^{-1} \\ 0 & I \end{bmatrix} \tag{2.75}$$

from the left, this equation can be factored into the block triangle form:

$$\begin{bmatrix} A_{11}^* & 0 \\ A_{21} & A_{22} \end{bmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 - A_{12}A_{22}^{-1}f_2 \\ f_2 \end{pmatrix} \tag{2.76}$$

where $A_{11}^* = A_{11} - A_{12}A_{22}^{-1}A_{21}$, which is known as the Schur complement (Karniadakis & Sherwin, 1999).

By assigning the boundary degrees of freedom to u_1 and the internal degrees of freedom to u_2, the boundary (shared) degrees of freedom can be solved first, then the rest forms a set of K independent matrix equations, where K is the number of elements. The matrix A_{11}^* is a symmetric banded matrix by construction whereas the matrix A_{22} is a block-structured matrix, each of which is a positive definite matrix.

2.3.9 Computationally efficient integration of the diffusion terms

Parallelization of the algorithms and their efficient implementation is crucial in obtaining numerical solution of reacting flows. The static condensation technique described in the previous section allows a natural decomposition of the entire degrees of freedom into those on the boundary of elements and those not on the boundary. Since the interior degrees of freedom are fully decompled from the boundary degrees of freedom, the interior points can be solved with perfect scalability up to the number of elements in the computational domain. On the other hand, the matrix for the boundary degrees of freedom (the Schur complement, A_{11}^* in Eqn. 2.76) couples all degrees of freedom on element boundaries as well as domain boundaries. This poses a challenge in solving the Schur matrix efficiently, particulaly using message-passing techniques. Since, for the spectral element method, the Schur complement is usually a symmetric positive definite banded matrix, this matrix can

be solved by either an iterative method, such as conjugate gradient method, or a direct method, such as Gaussian elimination. Fischer *et al.* (1988) implemented a parallel iterative technique using a conjugate gradient and multigrid method. Subsequent work is documented in Fischer & Patera (1991), Fischer & Rønquist (1994), and Deville *et al.* (2002). However, for reacting flows, the condition number of the matrix can be significantly larger than that for incompressible flow due to large density variations, and therefore the direct method is used in the present work. `dpbtrf` and `dpbtrs` in LAPACK or `pdpbtrf` and `pdpbtrs` in ScaLAPACK libraries, both of which implement Cholesky factorization and backward substitution, can be used for this purpose. The Cholesky factorization is the most expensive operation when solving the matrix Eqn. (2.76) among the processes shown in Fig. 2.5. To gain further computational efficiency, Eqns.(2.48) and (2.49) are rewritten to improve performance.

$$\frac{T^{n+1}-T^n}{\Delta t} = \left(-\mathbf{u}\cdot\nabla T + \left(\frac{1}{\rho C_{p,\mathrm{mix}}}\right)\nabla\lambda_T\cdot\nabla T + (\alpha^n-\alpha^0)\nabla^2 T\right)^n + \alpha^0\nabla^2 T^{n+1} \quad (2.77)$$

and

$$\frac{Y_{\mathfrak{m}}^{n+1}-Y_{\mathfrak{m}}^n}{\Delta t} = \left(-\mathbf{u}\cdot\nabla Y_{\mathfrak{m}} + \frac{1}{\rho^n}\nabla\rho D_{\mathfrak{m}}\cdot\nabla Y_{\mathfrak{m}} + (D_{\mathfrak{m}}^n - D_{\mathfrak{m}}^0)\nabla^2 Y_{\mathfrak{m}}\right)^n + D_{\mathfrak{m}}^0\nabla^2 Y_{\mathfrak{m}}^{n+1}, \quad (2.78)$$

where α^0 and $D_{\mathfrak{m}}^0$ are the initial values of the thermal diffusion coefficient and the diffusion coefficient of each species, respectively.

By factoring out the time-dependent part of the diffusion operator, factorization of the matrix is required only once at the beginning of iterations. $\|\alpha^n-\alpha^0\|$ (or $\|D_{\mathfrak{m}}^n - D_{\mathfrak{m}}^0\|$) is computed at every timestep, and when it exceeds a predetermined criterion, α^0 (or $D_{\mathfrak{m}}^0$) is updated and the matrix is factorized again.

Fig. 2.6 shows execution speed per iteration per number of collocation points for reacting flow. Without the efficient method presented here, the code experiences a slowdown in execution speed after $n_p = 4$, when the ScaLAPACK library (Blackford *et al.*, 1997) is not used. When ScaLAPACK is used, it shows good scalability; however it shows a significant overhead and overall, the code does not run any faster than when ScaLAPACK is not used. When the efficient method described here is used, the code scales well both with and without

the ScaLAPACK library. In particular, use of ScaLAPACK almost halves execution time.

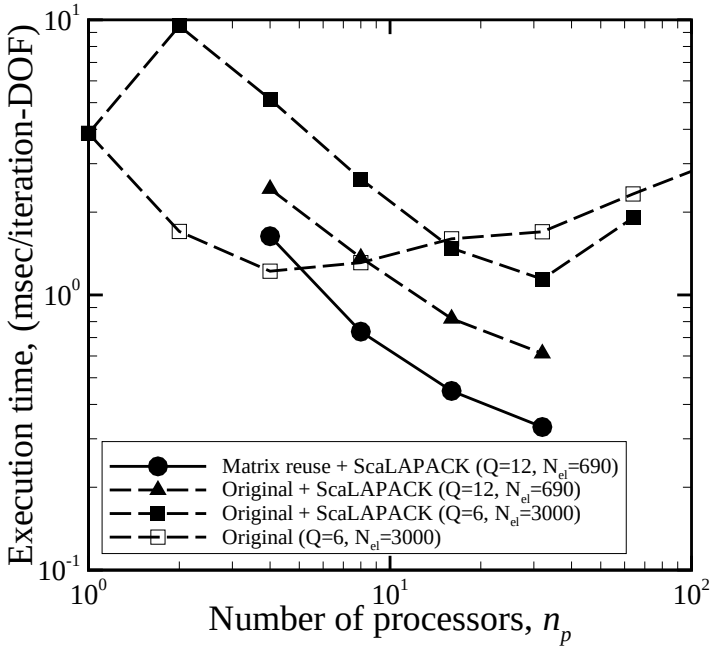


Figure 2.6: Computational performance of the Omega code used in the present study. Without the efficient integration, the code does not scale beyond $n_p = 4$. When the new method is employed, along with high-order polynomial basis, the code scales well up to $n_p = 16$ and speed up can be observed until $n_p = 32$. Use of ScaLAPACK enables the code to execute significantly faster. The benchmark test is one of Phase II simulations reported in Chapter 4. The execution time was measured on SHC cluster at the Caltech Center for Advanced Computing Research (CACR) with the code compiled with pathscale compilers.

Chapter 3

Software implementation, verification, and validation

An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software environment and the complete set of instructions which generated the figures.—D. Donoho

3.1 Introduction

In the field of computational science, it is important to have an accurate simulation environment so that we can be confident of the results and can actually rely on them in designing new products or in building better experiments.

As briefly mentioned in Chapter 1, care must be taken to ensure the correctness of the result obtained by computations. There are three kinds of error in every numerical study: modeling, numerical, and human errors.

$$\epsilon_{\text{total}} = f_{\text{error}}(\epsilon_{\text{modeling}}, \epsilon_{\text{numerical}}, \epsilon_{\text{human}}) \tag{3.1}$$

By driving these three error sources to zero, the total error should converge to zero, and the process by which the errors are small enough so that numerical solutions can be trusted is called verification or validation, depending on the type of error and the process and information employed.

The modeling error is an error in the governing equation that describes the physical system. For example, the uniform-density low Mach number flow is well described by the incompressible Navier-Stokes equations, *i.e.*, Eqn. (2.4), which are considered to offer a very

accurate description of the flow. However, for a chemically reacting system, the kinetics mechanism models may bring in more uncertainty than fidelity. Besides chemical kinetics models, transport models and coefficients may contain relatively large errors. There is a process that ensures the correctness or appropriateness of the model used in the computation, which is called validation. A validation is usually performed by comparing the obtained numerical solution to suitable experiments, and it checks if the agreement between them is reasonable. In other words, a validation is a process to quantify the modeling error.

The numerical error can be divided into three types, according to Ortega (1990): discretization error, convergence error, and rounding error. The discretization error arises whenever we approximate continuum equations by discrete ones and is the difference between the solutions to the continuum equations and the discrete equations. The convergence error arises when an infinite series is approximated by a finite sum. The rounding error—sometimes called quantization, approximation, or truncation error—is due to the inability of computers to represent real numbers to arbitrary precision. For modern computers, 64-bit floating point description is sufficient to make this error insignificant; 32-bit floating point descriptions may be used in a limited situation with caution.

The discretization error can usually be computed theoretically, while convergence error can be estimated by so-called convergence tests. There usually is no necessity to estimate rounding error, but if desired, one can do so by systematically driving discretization and convergence errors down.

The last kind of error, human error, includes both programmer and user error. Bugs in the coding process, wrong specification of the input parameters, and wrong usage of the code are typical examples of this type of error. This type of error is hard to quantify and is usually the biggest concern of all three. The programming part of the human error, at least the one that affects the accuracy of the numerical results, can be mostly eliminated by going through processes called regression testing (Lakos, 1996) and code verification (Roy, 2005).

Reducing user error is another big challenge in every field of engineering. A good strategy is to use a standardized and easy-to-use human interface. For example, the FORTRAN style inputs, that force users to align inputs exactly as they are read in, is a common source of user errors.

Definition 1 (Verification) *A verification is a process that ensures* $\epsilon_{\text{numerical}}$ *and* ϵ_{human} *are smaller than a tolerance criterion. The verification can be further divided into two categories: code verification that quantifies the programming error and the discretization error, and solution verification that quantifies the convergence error and the user error.*

Definition 2 (Validation) *A validation is a process by which* $\epsilon_{\text{modeling}}$ *is quantified. Validation is possible only when other two kinds of errors, namely* $\epsilon_{\text{numerical}}$ *and* ϵ_{human}*, are known and smaller than* $\epsilon_{\text{modeling}}$*. Therefore, the verification process must be conducted before going through the validation process.*

Models such as turbulence models, transport models, or chemical-kinetics models are ones that must be validated; codes, including an algorithm and its implementation, should be verified but cannot be validated. This fact is important because, quite often, inappropriate verification is used to certify a code. Customarily, numerical data are compared against *similar* experiments to claim that the code is "validated." As will be shown in a later chapter of this thesis, modeling the experimental apparatus without full details is usually not appropriate for validation, let alone the fact that the comparison to experimental data can never provide verification of any kind. Another naive verification often reported is to compare multidimensional simulation results against simpler numerical models such as a one-dimensional model. These may be regarded as a validation of simpler models via multidimensional simulations, but not the opposite.

We will discuss how the current program is implemented and how the object-oriented design can help minimize the probability of bugs and errors in the code, compared to the conventional function-oriented design in the first section.

Another important factor is speed or efficiency. The time it takes to obtain computational results can be broken down to three components: development time, compilation time, and execution time. Although the object-oriented technique was originally adopted to reduce any chance of introducing coding bugs in the code in the present study, it also helped reduce the development and maintenance time significantly as the evolving code became more complex.

There are some drawbacks of coding in C++. One of them is that the resulting code may not be as fast as C or FORTRAN programs in execution time. Another problem is that many, if not all, compilers are not fully compatible with the ANSI C++ standard,

and portability issues arise even when the code is written in a way that conforms to the standard.

3.2 Implementation

3.2.1 The structure of the Omega code

The Omega code developed as part of this project contains approximately 60,000 lines of source code, including header and implementation files, plus external libraries such as MPI (Gropp *et al.*, 1994, 1999) and LAPACK (Anderson *et al.*, 1999; Blackford *et al.*, 1997), which are not counted. Although traditionally computational scientists have preferred the use of FORTRAN programming languages due to performance reasons (Dowd & Severance, 1998), such a function-oriented technique is more error-prone. Object-oriented programming has an edge over function-oriented programming in keeping the bug density in the code small when the techniques are properly used, and therefore, the result is more scientifically reliable.

The implementation of the Omega code follows the protocols for a large-scale programming project, such as described in Lakos (1996), Stroustrup (1991), and Gamma *et al.* (1994).

3.2.1.1 Package structure

First, the static structure of the code is shown in Fig. 3.1. Each package is a collection of source files organized as a physically cohesive unit (Lakos, 1996). For example, the "domain" package is responsible for managing computational domain of the problem, consisting of 19 header files and 16 source files. As can be seen from the diagram, the packages are structured so that the dependency among packages are acyclic and unambiguously levelizable[†], *i.e.*, a level n package may depend on components of level $n-1$ or below only. Such dependency is important not only in implementing codes efficiently but also in maintaining software reliably, as each component can be tested independent of any other packages of the same level or above. These diagrams are described in a language called Unified Modeling Language (UML) and interested readers are encouraged to refer to Miles & Hamilton

[†]Such words as "levelizable" or "levelization" or their meaning may not be found in the Oxford English Dictionary but they are defined and used in Lakos (1996).

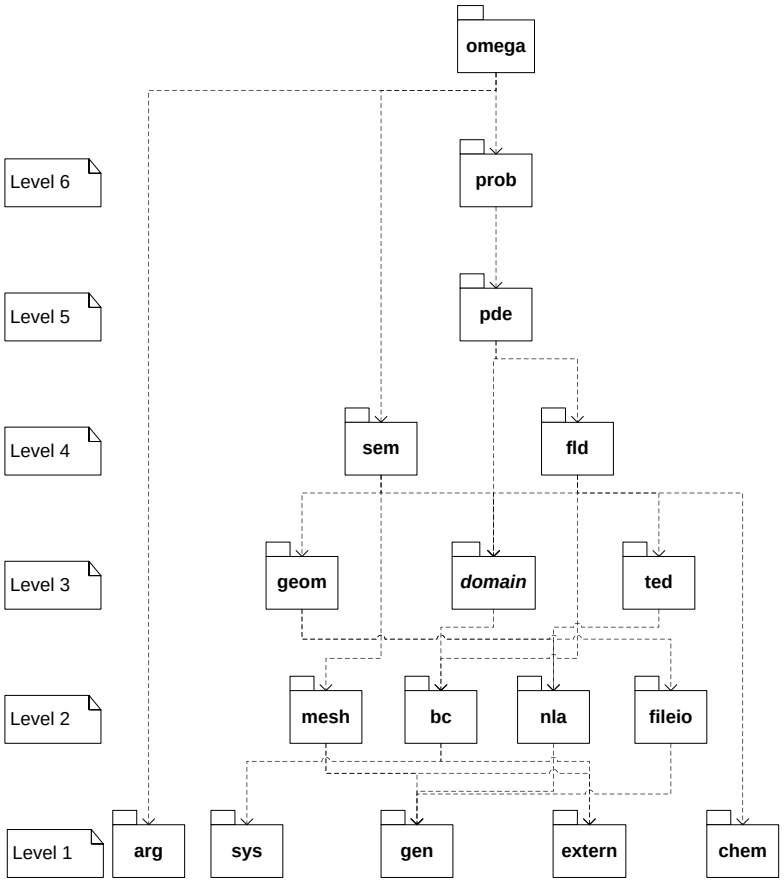


Figure 3.1: A static structure of the packages used in the *Omega* code. Note each package is acyclic and fully levelized, meaning that it can clearly define what other packages it depends on without cyclic dependency. Summary of responsibility of each package is described in Table 3.1.

(2006).

3.2.1.2 Domain structure

One of the problems with so many flow solvers written by so many people is the lack of consistency. Ideally, the whole CFD community should have a single program that computes every flow problem including aerodynamics, supersonic flow (including shockwaves), and low-speed or turbulent combustion. Traditionally, that was not an option because such an approach would make the software less efficient by loading many unnecessary capabilities. However, by not doing so, a significant amount of development time and code verification time is wasted. Although the Omega code does not have such a multipurpose capability currently, it is designed with such extensions in mind. In designing such multi-purpose

Table 3.1: Responsibility of each package

Name	Level	Description
omega		Driver program
prob	6	Problem (*e.g.*, laminar flame) description and specification
pde	5	Partial and ordinary differential equations
sem	4	Spectral element method
fld	4	Field variables such as pressure and velocity
geom	3	Geometry of computational domain
domain	3	Abstract computational domain and coordinate system
ted	3	time-evolving data for multi-step time-integration
mesh	2	Computational mesh within element or domain (1D & 2D)
bc	2	Boundary conditions
nla	2	Numerical linear algebra
fileio	2	Input / output functions
arg	1	Commandline argument processing unit
sys	1	Definition of system dependent variables
gen	1	Generic library (*e.g.*, smart pointers)
extern	1	Wrapper for external library functions
chem	1	Chemistry and transport properties

codes, it is important to extract common structure among different methods and capability and implement them efficiently. Design patterns (Gamma *et al.*, 1994) can be very useful for this purpose.

An example is the bridge pattern used in implementing the computational domain in the Omega code, as in Fig. 3.2. A client who needs to compute the divergence of the velocity field does not need to know in which coordinate system the velocity is defined. All he has to know is that the divergence is defined. How the divergence is computed is passed to the actual coordinate representation such as Cartesian or Cylindrical, for example. Then this coordinate system dictates how to compute the divergence in terms of derivative operators, *i.e.*, $\partial u/\partial x + \partial v/\partial y$ for Cartesian and $\partial u/\partial x + \partial v/\partial y + v/y$ for Cylindrical. These differentiations along coordinates ($\partial/\partial x$ and $\partial/\partial y$) are then evaluated in the actual numerical method implementation, such as the spectral element method (SEM) or finite difference method (FDM). In so doing, one can implement different coordinate systems, dimensions of the problem, and discretization methods in a single code with a uniform structure and interface.

Using this structure, the code can be extended beyond a two-dimensional axisymmetric

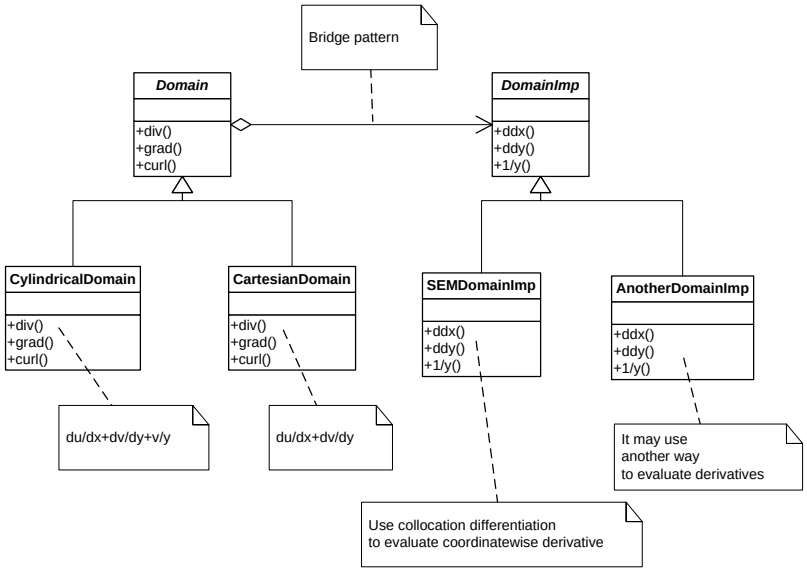


Figure 3.2: This is how the bridge pattern is used in the Omega code. The use of the bridge pattern allows decoupling of the coordinate representation (Cartesian or Cylindrical) from the discretization methods.

problem solver. In fact, the present code has a capability to handle problems in two-dimensional Cartesian coordinates. In addition, it should be easy to implement other discretization techniques, as necessary. One obvious advantage of having a computer program that solves problems in several different coordinate systems is its breadth and the capability to solve different kinds of problems. A less-obvious advantage, but certainly a compelling reason to choose such an implementation, is that it makes verification easy and makes the code more reliable. For example, by sharing many parts of the code when an erroneous outcome is identified that appears only when the code is used to solve problems in a particular coordinate system, then the "bug" is most likely to be in the part that is responsible for that specific coordinate. Since verification can be done in various problems in different coordinate systems, exercising many common components, it gives the user more confidence in the correctness of the code. It also contributes to reducing user errors by maintaining a uniform user interface.

3.2.1.3 Boundary / boundary condition structure

Incorporating boundary conditions is one of the most difficult parts of implementing flow solvers. This difficulty stems in part from the fact that the boundary conditions are as-

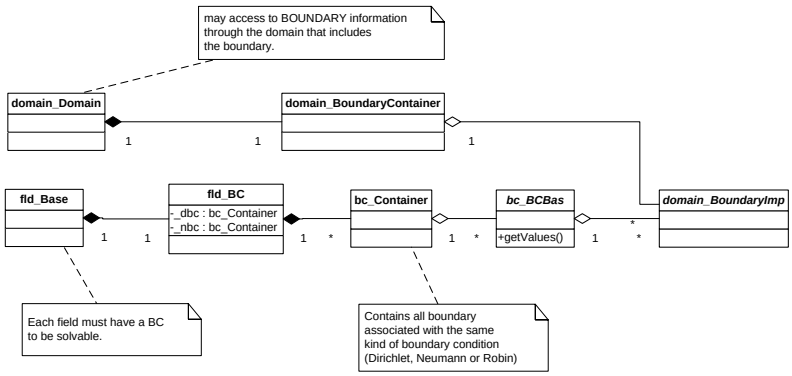


Figure 3.3: Static structure of boundary and boundary conditions

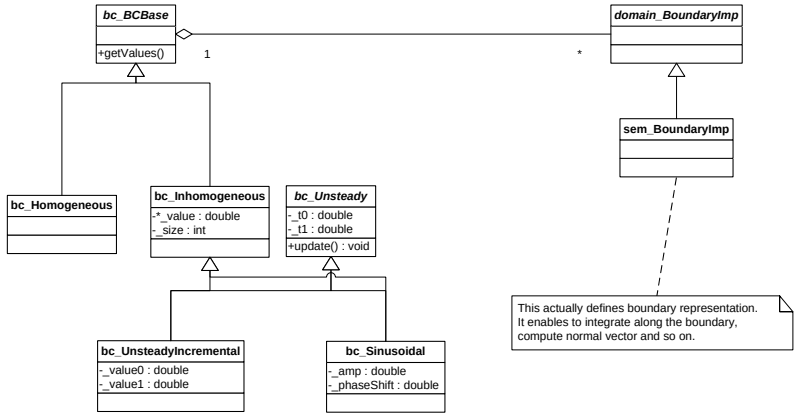


Figure 3.4: Use of bridge pattern in structuring boundary and boundary conditions

sociated with the boundary on which the conditions are defined, as well as conditions by which field values such as velocity or pressure are constrained. To make matters worse, the way in which boundary conditions are applied can depend on discretization methods. This three-way interaction makes the design of the boundary condition implementation a difficult task.

Shown in Fig. 3.3 is the static structure of the boundary and boundary conditions relating to domain and field variables. `fld_Base` is an object that represents a field variable such as velocity, and contains its own boundary conditions, `fld_BC`. The `fld_BC` object then includes two different containers, namely Dirichlet boundary conditions and Neumann boundary conditions. Each container contains as many `bc_BCBase` objects as necessary when that type of boundary condition is used in the problem specification. Each `bc_BCBase` object then contains a pointer or a reference to the boundary object that defines the boundary of the computational domain and numerical value (conditions) associated with this boundary

(and the field). In the other branch, `domain_Domain` is a representation of a computational domain, which contains a `domain_BoundaryContainer` that contains all boundary objects (`domain_BoundaryImp`) which define the boundary of the computational domain. Fig. 3.4 shows the interaction of boundary conditions with their discrete representation. Again the bridge pattern is used to separate the type of boundary conditions—such as homogeneous, inhomogeneous, or unsteady—to a particular representation of the domain—such as the spectral element method.

3.2.2 Procedure in solving matrix equations

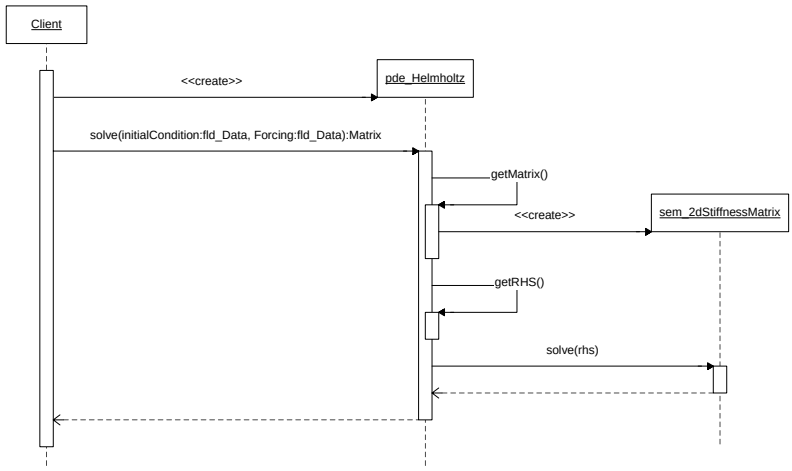


Figure 3.5: The sequence on how the client solves the Helmholtz equation

Shown in Fig. 3.5 is the sequence of how the Helmholtz equations are solved numerically. When a client requests the Helmholtz equation to be solved with a given right-hand side, the `pde_Helmholtz` object forms an appropriate operator matrix as well as the right-hand side vector, and solves the matrix equations. As described in the previous chapter (see page 30), the static condensation and the direct inversion of each resulting matrix is used to solve the linear system. Although an iterative method, such as a conjugate gradient, scales well in a parallel-computing environment, the condition number of the matrix system can be as large as 700 million for one of the cases investigated here, and an iterative method is not a viable option. While preconditioning was not explored in this study, we proceed with the direct method. Fig. 3.6 shows the elaborated sequence of processes used in obtaining the operator matrix for the Helmholtz equation, and Fig. 3.7 shows the sequence of solving the

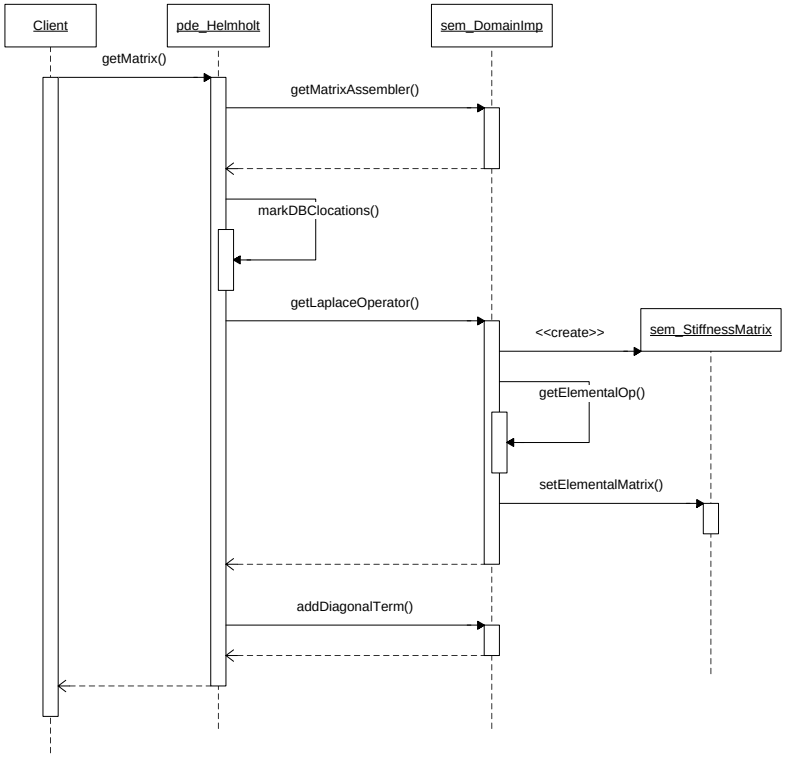


Figure 3.6: When the `getMatrix` method is used, this is the sequence that follows to obtain a weak form of the Laplace operator. Note the `getMatrix` method only computes elemental-level matrices, and assembly of the global operator due to direct stiffness summation is deferred until the assembled matrix is needed.

matrix equation.

3.2.3 Evaluation of transport properties

As discussed in the previous chapter, the viscosity and the self-diffusion coefficients of each species have a dependence on the collision integrals. Since they are a function of temperature only, it is customary to evaluate these transport properties *a priori*, and to tabulate or store them in terms of polynomial coefficients by fitting a polynomial instead of evaluating Eqn.(2.21) or Eqn.(2.31) directly during computation, to save computational time. In the present study, the polynomial fitting approach is adopted following Cantera (Goodwin, 2003), and the viscosity and diffusivities are expressed as follows:

$$\mu_{\mathfrak{m}}(T) = T^{\alpha}\left(\sum_{n=0}^{3}\mu_{\mathfrak{m},n}\left(\log(T)\right)^{n}\right) \tag{3.2}$$

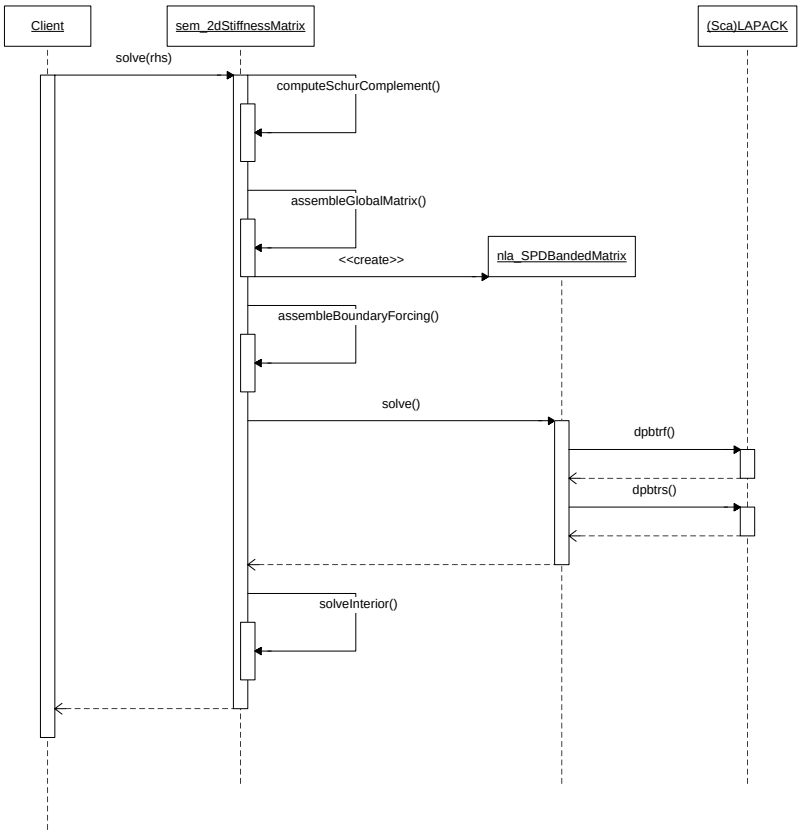


Figure 3.7: When the solve method in `sem_2dStiffnessMatrix` is used, the matrix object tries to factorize the matrix and assemble the global matrix when necessary. For example, `computeSchurComplement` computes A_{11}^* in Eqn.(2.76) at each element level, followed by `assembleGlobalMatrix` that assembles all contributions to A_{11}^* from each element to compute a symmetric positive definite banded matrix. A LAPACK function, `dpbtrf` (or `pdpbtrf` if ScaLAPACK is used), is responsible for Cholesky factorization of A_{11}^*.

$$D_{\mathtt{pq}}(T) = T^{1+\alpha}\left(\sum_{n=0}^{3} D_{\mathtt{pq},n}\left(\log(T)\right)^n\right). \tag{3.3}$$

$\alpha = 0.69$ is used after optimization for the temperature range of 300 K to 2200 K.

3.2.4 Chemistry ODE and Jacobian estimation

The chemical source terms in the species-transport and temperature-transport equations require special treatment because of their stiffness. The temporal evolution of a chemically reacting system is described by chemistry ODEs that arise after splitting the chemical source

terms from the entire governing equations for reacting flows:

$$\frac{dY_{\mathfrak{m}}}{dt} = \dot{\omega}_{\mathfrak{m}} \tag{3.4}$$

$$\frac{dT}{dt} = \dot{\omega}_T = -\frac{\sum_m h_{\mathfrak{m}} \dot{\omega}_{\mathfrak{m}}}{C_{p,\mathrm{mix}}}. \tag{3.5}$$

The right-hand side source terms are evaluated by a Fuego-generated C source code (Aivazis, 2002; Hung, 2003). Fuego is an object-oriented toolkit for chemical kinetics applications developed by Michael Aivazis. It parses a CHEMKIN format chemical mechanism file and produces a C source code that is CHEMKIN link-compatible, but provides increased efficiency by eliminating loops, conditional statements, and other computational overheads. CVODE (Cohen & Hindmarsh, 1994) is used to integrate these equations. Although CVODE can integrate stiff ODEs without a user-supplied Jacobian routine, it is important to have a good estimate of the Jacobian to integrate stiff ODEs such as this one efficiently.

From the Law of Mass Action, we obtain the rate of reaction in molar form,

$$\hat{\omega}_{\mathfrak{m}}(\mathbf{c}, T) = \sum_{k=1}^{K} \hat{\omega}_{\mathfrak{m},k}(\mathbf{c}, T), \tag{3.6}$$

where $\mathbf{c}$ denotes a set of concentrations of each species. This along with a chain rule of derivatives yield the following Jacobian structure in symbolic form:

$$\mathbf{J} = \left(\begin{array}{c|c} \frac{\partial \dot{\omega}_{\mathfrak{m}}}{\partial Y_{\mathfrak{n}}} & \frac{\partial \dot{\omega}_{\mathfrak{m}}}{\partial T} \\ \hline \frac{\partial \dot{\omega}_T}{\partial Y_{\mathfrak{n}}} & \frac{\partial \dot{\omega}_T}{\partial T} \end{array} \right). \tag{3.7}$$

Let the upper-left submatrix of $\mathbf{J}$ be $\mathbf{J}^0_{mn}$, which is

$$\mathbf{J}^0_{mn} \equiv \frac{\partial \dot{\omega}_{\mathfrak{m}}}{\partial Y_{\mathfrak{n}}} = \mathbf{J}^Y_{mn} + \frac{\bar{W}}{W_{\mathfrak{n}}} \left(\dot{\omega}_{\mathfrak{m}} - \sum_l \mathbf{J}^Y_{ml} Y_{\mathfrak{l}} \right) \tag{3.8}$$

$$\frac{\partial \dot{\omega}_T}{\partial Y_{\mathfrak{n}}} = -\frac{1}{C_{p,\mathrm{mix}}} \left(\sum_m h_{\mathfrak{m}} \mathbf{J}^0_{mn} + \dot{\omega}_T C_{p,\mathfrak{n}} \right) \tag{3.9}$$

where

$$\mathbf{J}^Y_{mn} = \frac{W_{\mathtt{m}}}{W_{\mathtt{n}}}\mathbf{J}^c_{mn}, \tag{3.10}$$

and

$$\mathbf{J}^c_{mn} = \frac{\partial \hat{\omega}_{\mathtt{m}}(\mathbf{c}, T)}{\partial c_n}, \tag{3.11}$$

where m is an index of species `m`. Eqn.(3.10) is supplied by the `fejay_` function in the *Fuego*-generated C source code.

Symbolic evaluation of $\partial\dot{\omega}_{\mathtt{m}}/\partial T$ and $\partial\dot{\omega}_T/\partial T$ terms is more complicated as the explicit form of such derivatives is quite lengthy. We use finite-difference approximations to the last column of the Jacobian matrix.

By using this almost symbolic Jacobian, about a 20% reduction of the number of calls to the function that computes chemical source terms and a 100% improvement in execution speed was achieved in one case.

3.3 Code verification

To make sure that the code does what the developer intends it to do, the importance of verification can never be overemphasized. Within the context of the three errors in numerical simulations mentioned earlier, verification is conducted to quantify errors stemming from numerical errors and the programming part of human error. First, a convergence study of component solvers will be described to make sure that the error decays as we refine spatial and temporal sampling intervals, so that we have a control of discretization errors.

Throughout this study, the error is measured by the L^2 norm,

$$\|f(x,y)\| = \left(\int_\Omega \left(f(x,y)\right)^2 d\Omega\right)^{1/2} \approx \left(\sum_{i,j} w_{i,j} f^2_{i,j}\right)^{1/2}, \tag{3.12}$$

unless otherwise noted.

3.3.1 Verification of components

As described in the previous chapter, the flow solvers for the incompressible Navier-Stokes equations make extensive use of the Poisson equation and the Helmholtz equations. In this section, we will investigate the convergence rate of the Helmholtz equation solver to ensure

that the advertised rate of convergence is achieved.

Fig.(3.8) shows a convergence study of the Helmholtz solver in cylindrical coordinates, for a synthesized problem. The solution is $u(z,r) = \sin(\pi z)\cos(\pi r)$ with even radial parity, and the boundary condition and the forcing term are such that the solution satisfies the Helmholtz equation, $\nabla^2 u + \lambda u = f$ where $\lambda = -\pi^2 \sin(\pi x)\cos(\pi y)$.

In the case of h-refinement, in which polynomial orders are kept constant while the number of elements is increased (thus the element size being refined), it is expected to converge at fifth order; and for p-refinement, in which elements are fixed and polynomial orders are increased, exponential convergence is expected. This property can be seen in Fig.(3.8).

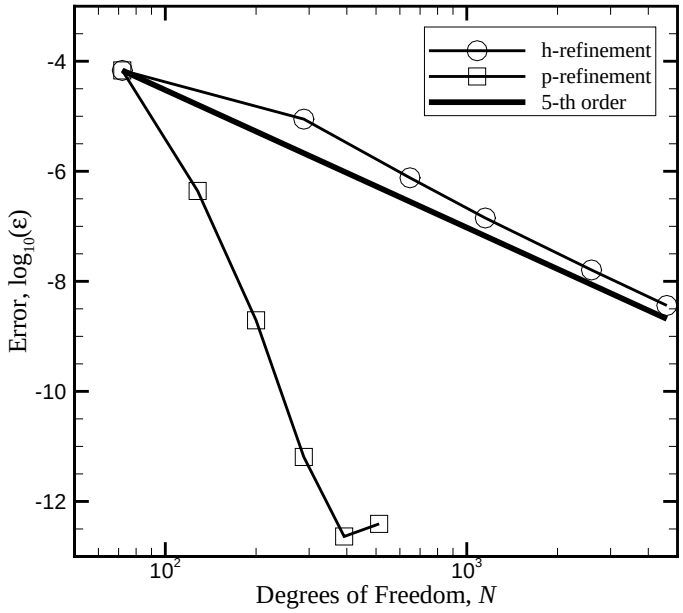


Figure 3.8: h-refinement case shows fifth-order convergence while p-refinement shows faster convergence.

3.3.2 Verification by the method of exact solution

One of the advantages of being able to solve Cartesian problems in addition to cylindrical ones within the same software is that it makes it possible to conduct verification tests that are otherwise not possible. Kovasznay flow in a Cartesian coordinate system is one of the

Table 3.2: Convergence data of Helmholtz equation solver

K	Q	N	N_{Net}	Error
2	6	72	32+34	6.81278e-005
8	6	288	128+103	8.87486e-006
18	6	648	288+208	7.61115e-007
32	6	1152	512+349	1.41611e-007
72	6	2592	1152+739	1.61071e-008
128	6	4608	2048+1273	3.64283e-009
2	8	128	72+48	4.39157e-007
2	10	200	128+62	1.96032e-009
2	12	288	200+76	6.43544e-012
2	14	392	288+90	2.30857e-013
2	16	512	392+104	3.9094e-013

few exact solutions of the Navier-Stokes equations that exercise every term in the equation (Kovasznay, 1948).

The Kovasznay flow is described by

$$u(x, y; \lambda) = 1 - \exp(\lambda x)\cos(2\pi y) \tag{3.13}$$

$$v(x, y; \lambda) = \frac{\lambda}{2\pi}\exp(\lambda x)\sin(2\pi y) \tag{3.14}$$

$$p(x; \lambda) = 1 - \exp(2\lambda x)/2 \tag{3.15}$$

where

$$\lambda = Re/2 - \sqrt{Re^2/4 + 4\pi^2}. \tag{3.16}$$

This solution has been used in verification in the past (Karniadakis *et al.*, 1991; Blackburn & Sherwin, 2004). In particular, Blackburn & Sherwin (2004) used this exact solution to verify their implementation in three-dimensional cylindrical coordinates by translating this solution in two-dimensional Cartesian coordinates into a three-dimensional cylindrical coordinate system.

As shown in Fig. 3.9, the expected rate of convergence is observed using h-refinement with seventh-order accuracy, as well as the high-order convergence using p-refinement.

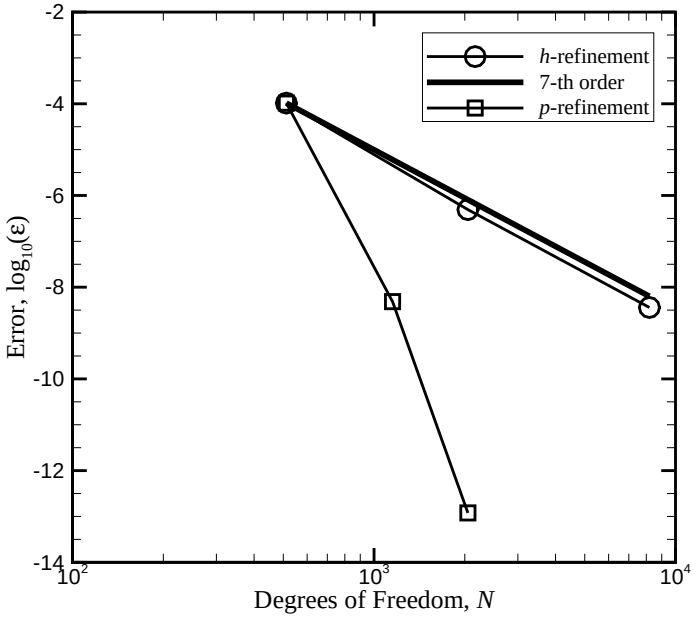


Figure 3.9: h-refinement case shows seventh-order convergence while p-refinement case shows faster convergence.

3.3.3 Verification of the code via the method of manufactured solution

The verification of a code through an exact solution is very powerful. However, quite often there is no exact solution available in closed form without oversimplifications. The method of manufactured solution is universally applicable to verification of a code whether or not an exact solution is available (Roy, 2005). In the present study, an exact solution based on the Kovasznay flow solution is used to create a problem with an exact solution:

$$\rho(z,r) = \exp(\lambda z/2) \tag{3.17a}$$

$$u(z,r) = \exp(-\lambda z/2) - \exp(\lambda z/2)\cos(2\pi r) \tag{3.17b}$$

$$v(z,r) = \lambda \exp(\lambda z/2)\sin(2\pi r)/(2\pi) \tag{3.17c}$$

$$p(z) = (1 - \exp(2az))/2 \tag{3.17d}$$

$$T(z,r) = T_0/(\exp(C_0 z)(C_1 + C_2\cos(2\pi r) + C_3\sin(2\pi z))) \tag{3.17e}$$

$$Y_{\mathrm{H2}} = Y_{10} + Y_{11}\sin(2\pi z) + Y_{12}\cos(2\pi r) \tag{3.17f}$$

$$Y_{\mathrm{N2}} = 1 - Y_{10} - Y_{11}\sin(2\pi z) - Y_{12}\cos(2\pi r) \tag{3.17g}$$

where $T_0 = -688.0862031843639$, $C_0 = (5 - \sqrt{25 + 4\pi^2})/2$, $C_1 = -9.815178$, $C_2 = C_3 = -1.299873$, $Y_{10} = 0.3$, $Y_{11} = Y_{12} = 0.05$ are used.

This solution satisfies the divergence constraint, $\nabla \cdot \rho\mathbf{u} = 0$, and the mass-fraction constraint, $\sum Y_{\mathfrak{m}} = 1$. The exact solution of temperature is somewhat complicated due to its coupling to density and mass fraction fields through the equation of state. The required forcing term to balance these manufactured solution on the governing equations is computed using *Mathematica*®, and the C source code generated by *Mathematica*®is used to evaluate the forcing terms.

Figure 3.10 shows the decay of errors for both h-refinement, in which the number of elements is increased with fixed polynomial order, and p-refinement, in which polynomial order is increased with the number of elements fixed.

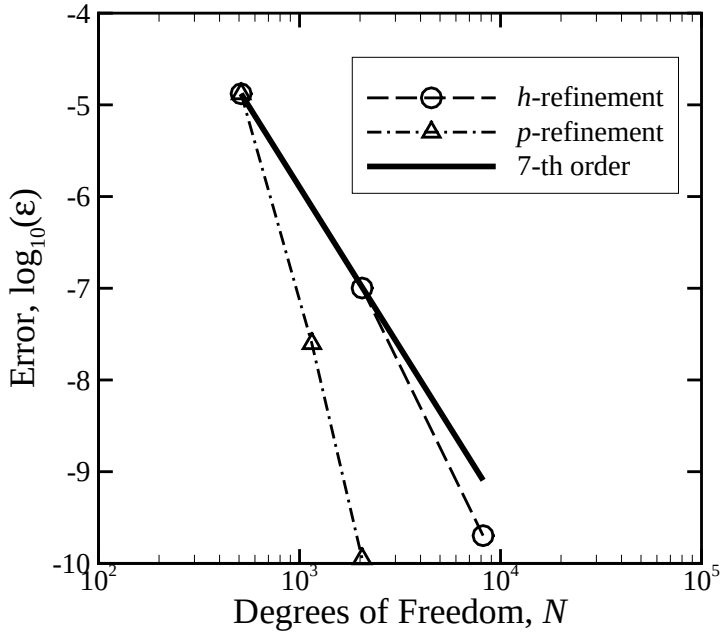


Figure 3.10: The method of manufactured solution provides verification of the code. h-refinement case shows seventh-order convergence while p-refinement case shows faster convergence. Both error decay rates are in line with the expected rate.

3.3.4 Verification of the code against another numerical solution

Although this test is not exactly verification in the sense of the definition given above, it is still a useful comparison to gain confidence on the newly developed code. Frouzakis *et al.*

(1998) reported non-premixed opposed-jet hydrogen flames at a Reynolds number of 100, using a spectral element method and the hydrogen oxidation kinetic mechanism by Yetter *et al.* (1991) (H4 mechanism in Table 1.1). The opposed jet flame is very similar in nature to a stagnation flame, and therefore, is an appropriate test case for the present work. We have set up a computational domain using the same elements, as reported in Frouzakis *et al.* (1998), with the same polynomial order. Shown in Fig. 3.11 is the computational domain used for this study. Hydrogen diluted with nitrogen is introduced from the right between $0 \leq r \leq 0.005\,\mathrm{m}$ with the inlet velocity of $0.15925\,\mathrm{m/s}$, and air is introduced from the left at the same velocity. The top part of the domain ($r = 0.02\,\mathrm{m}$) is an outflow boundary condition, and the rest is an isothermal wall, kept at $300\,\mathrm{K}$.

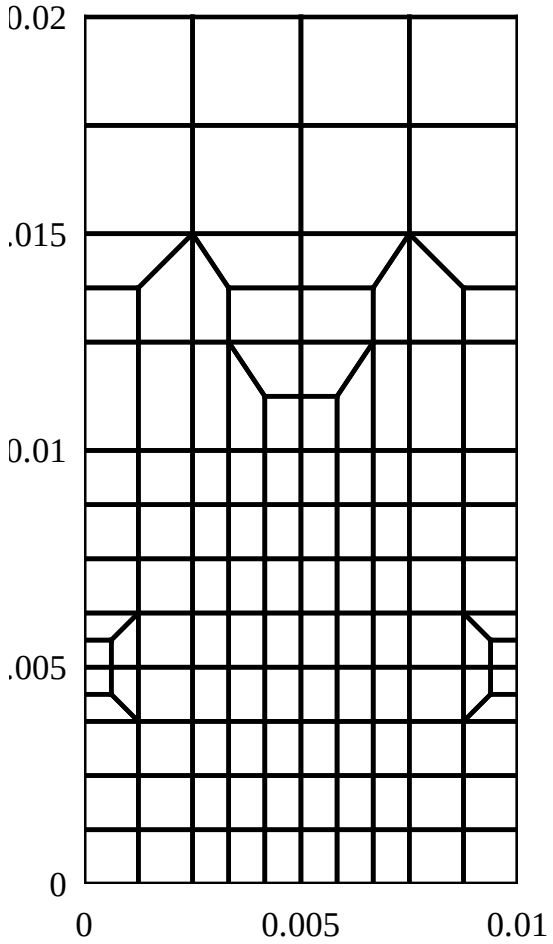


Figure 3.11: Computational domain for opposed-jet hydrogen flame study taken from Frouzakis *et al.* (1998). It includes 120 elements; each contains 6 by 6 collocation points.

Shown in Figs. 3.12–3.15 are comparisons of hydrogen and hydroxyl radicals, temperature, and axial velocity profiles between the Omega code, developed in the present work, and the two-dimensional numerical solution by Frouzakis *et al.* (1998). One-dimensional solutions using CHEMKIN, reported in Frouzakis *et al.* (1998) are also plotted in each figure for reference. The same hydrogen kinetics model (Yetter *et al.*, 1991) is used in all cases. Despite some differences in the formulation and transport models, the two two-dimensional solutions are in good agreemeent.

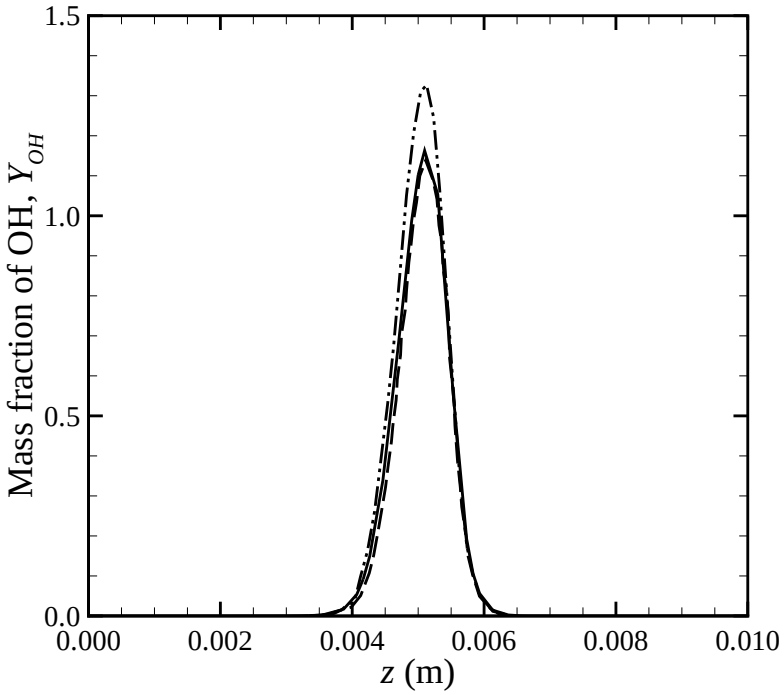


Figure 3.12: The mass fraction of the hydroxyl radical profile is compared among the Omega code (solid line) developed in the present work, the two-dimensional solution reported in Frouzakis *et al.* (1998) (dashed line), and the one-dimensional solution using CHEMKIN also reported in Frouzakis *et al.* (1998) (double-dot-dashed line).

3.4 The validation of the model

Validation refers to comparison to experimental data or a theoretical prediction that can support the correctness of the numerical solution. For example, it is easy to verify that the numerical solution actually satisfies the set of equations by looking at the residuals after substituting the numerical solutions to the equations. However, this does not mean that the numerical solution is the right one; there might be a modeling error. In addition to

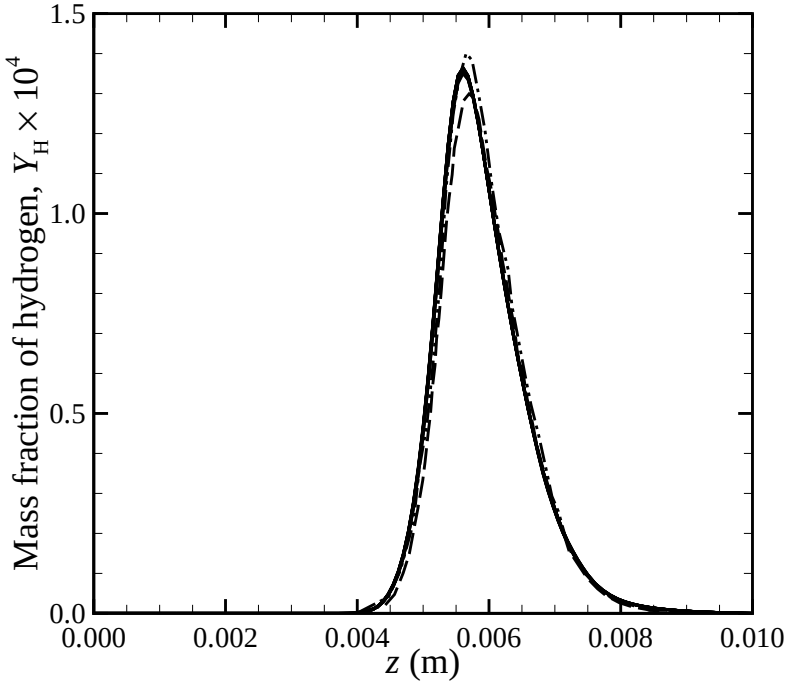


Figure 3.13: Comparison of the mass fraction of hydrogen atom profile. (Legend as in Fig. 3.12.

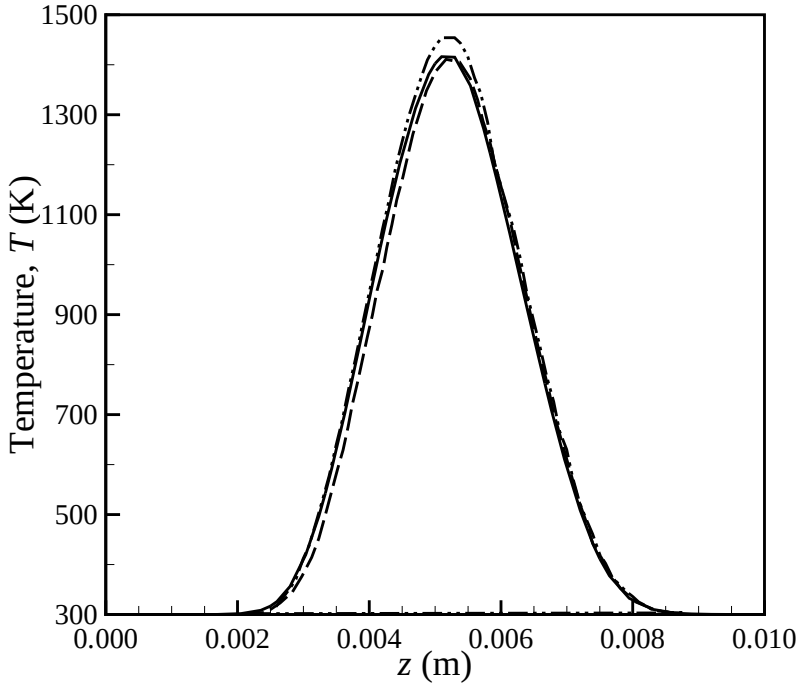


Figure 3.14: Comparison of the temperature profile. (Legend as in Fig. 3.12.)

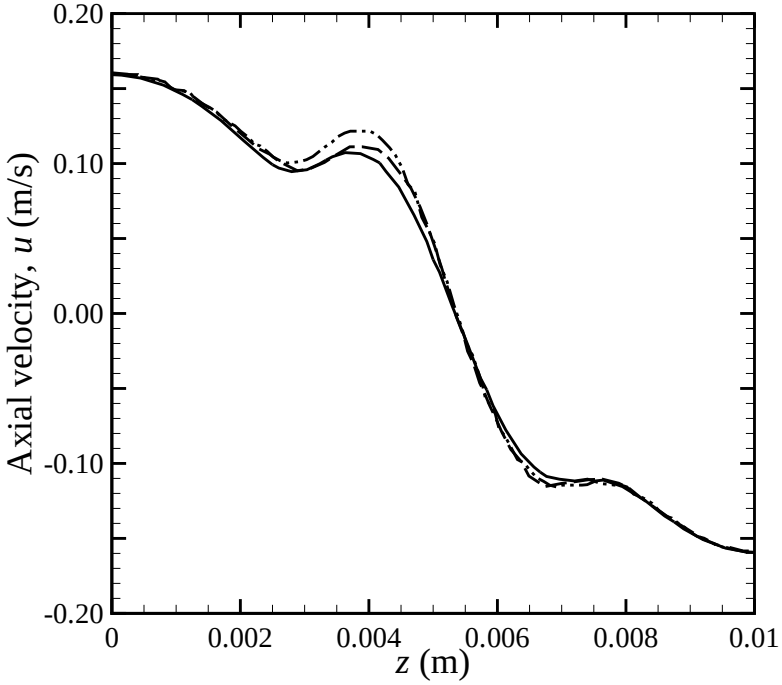


Figure 3.15: Comparison of the axial velocity profile. (Legend as in Fig. 3.12.)

modeling of fluid mechanics, there are three model components used in the numerical study of reacting flows: thermodynamics (such as enthalpy and heat of formation), the transport model and coefficients, and the chemical kinetics. Although each of the three components needs to be improved, and is improving today, it is expected that the chemical kinetics model contains more uncertainty and error (Williams, 2000). The validation of reacting-flow models is one of the main themes of the present study and will be discussed in the following chapter.

It is worth noting that the modeling error, $\epsilon_{\text{modeling}}$ in Eqn. (3.1), can be quantified only by simulating experiments directly. For example, if the study of an ideal stagnation flame that extendes to infinity in radial direction with perfectly flat flame shape is considered, and this ideal setting is modeled in both experiments and numerical study, it is impossible to quantify $\epsilon_{\text{modeling}}$ because experiments represent a different situation since such an ideal flame cannot be realized in the laboratory. However, by simulating flames observed in a laboratory directly, instead of modeling idealized flames, it is possible to quantify the modeling error that appears in mathematical models, $\epsilon_{\text{modeling}}$. The availability of the recent high-quality experimental data and all necessary experimental implementation details from

the work of Bergthorson *et al.* (2005b) and Bergthorson (2005), along with the necessary computational resources and support, permitted this approach to be adopted in the present study.