University of Kentucky
UKnowledge

Theses and Dissertations--Mechanical Engineering

Mechanical Engineering

2015

LOW-ORDER DISCRETE DYNAMICAL SYSTEM FOR H_2-AIR FINITE-RATE COMBUSTION PROCESS

Wenwei Zeng
University of Kentucky, wenweizengchina@gmail.com

Right click to open a feedback form in a new tab to let us know how this document benefits you.

Recommended Citation

Zeng, Wenwei, "LOW-ORDER DISCRETE DYNAMICAL SYSTEM FOR H_2-AIR FINITE-RATE COMBUSTION PROCESS" (2015). *Theses and Dissertations--Mechanical Engineering*. 73.
https://uknowledge.uky.edu/me_etds/73

This Master's Thesis is brought to you for free and open access by the Mechanical Engineering at UKnowledge. It has been accepted for inclusion in Theses and Dissertations--Mechanical Engineering by an authorized administrator of UKnowledge. For more information, please contact UKnowledge@lsv.uky.edu.

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Wenwei Zeng, Student

Dr. James M. McDonough, Major Professor

Dr. Haluk E. Karaca, Director of Graduate Studies

LOW-ORDER DISCRETE DYNAMICAL SYSTEM FOR H_2-AIR FINITE-RATE COMBUSTION PROCESS

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Mechanical Engineering in the College of Engineering
at the University of Kentucky

by

Wenwei Zeng

Lexington, Kentucky

Director: Dr. J. M. McDonough, Professor of Mechanical Engineering and Mathematics
Co-Directors: Dr. José Graña-Otero, Professor of Mechanical Engineering

Lexington, Kentucky

2015

Copyright © Wenwei Zeng 2015

ABSTRACT OF THESIS

LOW-ORDER DISCRETE DYNAMICAL SYSTEM FOR H_2-AIR FINITE-RATE COMBUSTION PROCESS

A low-order discrete dynamical system (DDS) for finite-rate chemistry of H_2-air combustion is derived in 3D. Fourier series with a single wavevector are employed to represent dependent variables of subgrid-scale (SGS) behaviors for applications to large-eddy simulation (LES). A Galerkin approximation is applied to the governing equations for comprising the DDS. Regime maps are employed to aid qualitative determination of useful values for bifurcation parameters of the DDS. Both isotropic and anisotropic assumptions are employed when constructing regime maps and studying bifurcation parameters sequences. For H_2-air reactions, two reduced chemical mechanisms are studied via the DDS. As input to the DDS, physical quantities from experimental turbulent flow are used. Numerical solutions consisting of time series of velocities, species mass fractions, temperature, and the sum of mass fractions are analyzed. Numerical solutions are compared with experimental data at selected spatial locations within the experimental flame to check whether this model is suitable for an entire flame field. The comparisons show the DDS can mimic turbulent combustion behaviors in a qualitative sense, and the time-averaged computed results of some species are quantitatively close to experimental data.

KEYWORDS: Discrete Dynamical System, Subgrid-scale Model, Diffusion flame, Finite Rate Chemistry

Wenwei Zeng
December 14, 2015

Low-order Discrete Dynamical System For H_2-air Finite-rate Combustion Process

by

Wenwei Zeng

Dr. J. M. McDonough

Director of Thesis

Dr. José Graña-Otero

Co-Director of Thesis

Dr. Haluk Karaca,

Director of Graduate Studies

December 14, 2015

Acknowledgements

Learning is an endless process, and studying is one part of my life. Studying and making research in the graduate school was an valuable experience for me, and it's an honor for me to work with Dr. James McDonough. Meticulous is a mark of his life, and he believes that both mathematics derivation and technical writing are pieces of artwork. Finally, I become a qualified researcher with the help from Dr. McDonough, and this thesis is a product that I worked on in the past three years. I want to thank Dr. Jose Grana-Otero, who instructed me a lot in combustion area. I also want to thank Dr. Alexandre Martin, who never complained my bad manner in communication.

I do appreciate my parents, Xianglian Li and Guiqiu Zeng, who sacrificed a lot and provided never-ending support for me in my life. I grew up in a small village in China, with limited financial resources from my family, as my parents are farmers and laborers. Without their support, my sisters and me could not become first-generation college students, let alone continued graduate studies in the USA. Thanks for my sister, Linli Zeng and Emily Zeng, they are always willing to impart their experience and suggestions to me when I was in plight.

Last, I want to acknowledge my friends Rick Fu, Tingting Tang, Weiyun Liu, Paul Zhang, Zach Li, Martin Weng. They are closet friends that I want to brothers and sisters, and I hope our friendship could last forever. It's a very good time for me to stay and work with them.

Table of Contents

Acknowledgements **iii**

List of Tables **vii**

List of Figures **viii**

1 Introduction and Background **1**

1.1 Background 1

1.2 Finite-rate Models 3

1.3 Numerical Approach 7

1.3.1 Effects of Turbulence 7

1.3.2 Need for Parallelization 8

1.4 Motivation 10

1.5 Outline of Thesis 25

2 Discrete Dynamical System Model Analysis **26**

2.1 Governing Equations 26

2.2 Construction of the Dynamical Systems 28

2.2.1 Galerkin approximation to the governing equations 30

2.2.2 Euler integration and discrete dynamical systems 37

2.2.3 Homogeneous and isotropic assumptions 41

2.2.4 Inhomogeneity and anisotropy 43

2.3 Finite-Rate Chemistry Reduced Mechanism 43

2.3.1 Investigation on N_2 Chemistry 45
2.4 Discrete Dynamical System Model for Specific Species Reaction . . . 47
2.5 Temperature Model 49

3 Experimental Data **51**
3.1 Flame Axis 55
3.2 Maximum Temperature 56
3.3 Flame Boundary 57
3.4 Initial Condition 58

4 Results and Discussion **61**
4.1 Numerical Solution and Discussion 61
4.2 Regime Maps 61
4.2.1 β vs. γ in Homogeneous and Isotropic Assumption 62
4.2.2 β vs. γ with Reacting N_2 Mechanism 65
4.2.3 β vs. γ in Inhomogeneous and Anisotropic Conditions 65
4.3 Temperature-Phase Portraits 68
4.4 Numerical Results 71
4.4.1 Validation of model 72
4.4.2 Solution comparisons with experimental data 74
4.5 Discrepancy Analysis 93

5 Conclusions and Future Work **97**

Appendices **101**
A Reduced Chemical Mechanism Fortran Code 101
B Regime Maps Parallel Computing Fortran Code 123

Bibliography **176**

Vita **189**

List of Tables

2.1 Rate Coefficients for Reduced Mechanism of H_2-Air Reaction 44

2.2 Rate Coefficients for Thermal NO_x Mechanism 47

3.1 Experimental mean values for temperature and mass fractions: flame axis 56

3.2 Experimental mean values for temperature and mass fractions: maximum temperature 57

3.3 Experimental mean values for temperature and mass fractions: flame boundary 58

3.4 Initial conditions for temperature and mass fractions 59

3.5 Experimental mean values for temperature and mass fractions 60

4.1 Values of parameters with homogeneous and isotropic assumption . . 63

4.2 Value of parameters with inhomogeneous and anisotropic conditions . 66

4.3 Solution and comparison at location: $x/D = 2.5$, $r = 0$ 74

4.4 Solution and comparison at location: $x/D = 2.5$, $r = 6.75$ 79

4.5 Solution and comparison at location: $x/D = 10$, $r = 9$ 82

4.6 Solution and comparison at location: $x/D = 20$, $r = 10.5$ 86

4.7 Solution and comparison at location: $x/D = 30$, $r = 7.5$ 90

List of Figures

3.1 Burner Structure and Diffusion Flames Structures 52

3.2 Global flame temperature distribution 52

3.3 Temperature and species concentration distribution on specific heights 54

4.1 Regime map with Homogenous and Isotropic Assumption: (a) β vs. γ at location $x/D = 2.5$, $r = 0$; (b) zoom at interested region; (c) color table for flow states in regime maps 64

4.2 Regime map with inhomogeneous and anisotropic condition: (a) β_2 vs. γ_2 at location $x/D = 2.5$, $r = 0$; (b) β_3 vs. γ_3 at location $x/D = 2.5$, $r = 0$; (c) β_2 vs. γ_2 at location $x/D = 2.5$, $r = 6.75$; (d) β_3 vs. γ_3 at location $x/D = 2.5$, $r = 6.75$; (e) β_2 vs. γ_2 at location $x/D = 10$, $r = 9$; (f) β_3 vs. γ_3 at location $x/D = 10$, $r = 9$; (g) β_2 vs. γ_2 at location $x/D = 20$, $r = 10.5$; (h) β_3 vs. γ_3 at location $x/D = 20$, $r = 10.5$; (i) β_2 vs. γ_2 at location $x/D = 30$, $r = 7.5$; (j) β_3 vs. γ_3 at location $x/D = 30$, $r = 7.5$; (k) color table for flow states 67

4.3 Temperature-Species Concentration Phase Portraits 69

4.4 Sum of mass fractions in Time 73

4.5 Numerical solution and comparison at location: $x/D = 2.5$ with $r = 0$ 76

4.6 Numerical solution and comparison at location: $x/D = 2.5$, $r = 6.75$. 80

4.7 Numerical solution and comparison at location: $x/D = 10$, $r = 9$ . . . 83

4.8 Numerical solution and comparison at location: $x/D = 20$, $r = 10.5$. 87

4.9 Numerical solution and comparison at location: $x/D = 30$, $r = 7.5$ . . 91

Chapter 1 Introduction and Background

1.1 Background

Current targets of combustion research include optimizing combustor operation, monitoring the combustion process, and reducing the severe consequences of instabilities. There are several aspects to improve combustion system performance, such as reducing the levels of pollutant emissions, and smoothing the pattern factor correlation parameters at the combustor exhaust. In terms of instabilities, the purpose is to extend the stability region by reducing the level of oscillation that is caused by combustion processes. As combustion systems pursue high performance and meet increasingly stringent air pollution standards, combustion equipment design and operation become more complex, as noted by Docqier and Candel [1]. In the past decade, low pollutant emission has became a feature for new combustion equipments. In addition, for a specific fuel, optimizing the performance of combustion system is necessary for seeking a balance between low emissions and operating performance, as discussed by Richards *et al.* [2]. Thus, investigation on understanding and monitoring combustion processes and system is more and more important.

In order to optimize design of combustion devices, a long-term goal of research on the combustion process is to develop mathematical flame models. Turbulent flames are very complicated; thus, a stepwise approach starting with simple flame configurations seems to be a reasonable approach. The essential quantities for an overall characterization of the chemical state within a flame include temperature and major species concentrations and their fluctuations, and these variables should be diagnosed

by detection techniques. This area of research relies strongly on the availability of appropriate measuring techniques, which can verify the results of flame simulations or discover shortcomings in the models applied, see Meier *et al.* [3].

Combustion models can be divided into two main groups according to the assumptions on the reaction kinetics, namely, infinitely-fast chemistry and finite-rate chemistry. Although hydrocarbon fuels and pure hydrogen exhibit fast chemistry, it is clear from experiments and theoretical investigation that the assumption of infinite-rate chemistry for them is not appropriate if highly accurate results are sought, as noted by Neuter *et al.* [4]. This is because minor species in a reaction are more sensitive than are major species relative to finite-rate chemistry, but may play a significant role in the reactions path.

There are three types of combustion flames in terms of mixing type: premixed, non-premixed and partially premixed flames. In premixed flames, the fuel and oxidizer are mixed well prior to ignition, and these flames are not limited only to gaseous fuels, but also to pre-vaporised fuels. Non-premixed flames represent a special class of combustion, wherein fuel and oxidizer enter the combustion chamber separately. Diffusion and mixing of the two flows bring the reactants together; then reaction occurs. Mixing is the key characteristic for non-premixed flames. Partially premixed flames have different flammability characteristics from those of non-premixed and fully premixed flames, they are non-uniform with respect to fuel-air mixing and produce two or more reactions. Partially premixed flames are formed when a fuel flow is mixed with a less than stoichiometric quality of oxidizer before the reaction zone, and the fuel and oxidizer are mixed well in the reaction zone, as described by Aggarwal [5]. The partially-premixed regions are formed by gaseous fuel leaks and in evaporating liquids, as mentioned by Puri *et al.* [6].

1.2 Finite-rate Models

Many years of research associated with finite-rate chemistry in turbulent combustion has focused on the first two of the above mentioned flame types. Neuber *et al.* [4] investigated finite-rate chemistry in non-premixed turbulent flames, and they studied turbulent N_2 diluted H_2 diffusion flames by means of laser spectroscopic methods and a numerical combustion model; results shown that theoretical spatial maxima of the mean OH mole fraction matched with experimental data well in magnitude. Bray *et al.* [7] conducted research on premixed turbulent combustion with finite-rate chemistry and presumed probability density function (PDF), and they investigated the sensitivity of prediction of mean reaction rates in turbulent premixed flames to presumed PDF shape. They compared three different presumed PDF shapes with direct numerical simulation (DNS) data, and the comparison showed the beta function and twin delta function PDFs make significant mistakes, while the PDF based on unstrained laminar flame properties agrees well with the DNS data. Dunn *et al.* [8] applied numerical calculations with a particle based PDF method to study detailed scalar structure measurements of highly sheared turbulent premixed flames on a piloted premixed jet burner. They found that as shear rates increase, the finite-rate chemistry effects decrease gradually in reactiveness, and a particle-based PDF model with modified mixing frequency was able to predict the measured flames with finite-rate chemistry effects. Lindstedt *et al.* [9] employed perturbation approaches to introduce finite-rate chemistry effects and provided an assessment of uncertainties in the formation chemistry of NO_x. They pointed out that formation of nitric oxide is kinetically controlled, and that calculation procedures are able to predict interactions between turbulence and finite-rate chemistry over a wide range of Damköhler numbers. Irannezhad [10] performed a numerical analysis of a laboratory low-swirl stabilized flame via a large-eddy simulation model together with a finite-rate chem-

istry combustion model. They found that computational domain and inlet boundary conditions have significant effects on flame stabilization mechanisms in a numerical simulation.

Chemical-kinetic mechanisms play an important role in combustion research, and they can be used to study autoignition, deflagrations, detonations and diffusion flames, etc. On the basis of rate parameters of elementary reaction in combustion processes, description of the elementary reaction steps have moved forward extremely in the last couple decades through the development of detailed chemical-kinetic mechanisms. Healy *et al.* [11] developed a detailed chemical kinetic mechanism to study the oxidation of mixtures of $CH_4/C_2H_6/C_3H_8$/n-C_4H_{10}/n-C_5H_{12} at high pressure and intermediate to high temperatures. Wang [12] developed an skeletal mechanism from a detailed mechanisms for hydrogen (H_2) and C_1-C_4 hydrocarbons. He made a systematical error analysis and and checked the chemical reality of the skeletal mechanism, the numerical results showed that the skeletal mechanisms exhibited very good performance in prediction of ignition for hydrogen, methane, ethylene, ethane, propene, propane and n-butane. Westbrook *et al.* [13] developed detailed chemical kinetic reaction mechanisms for the pyrolysis and oxidation of nine n-alkanes larger than n-heptane in both low temperature and high temperature reaction pathways, and these mechanisms have been validated by the experimental data from different sources.

Dixon-Lewis et al. [14] applied a "composite flux" method to analyze the solution of multi-radical, premixed laminar flames, and they discussed the radical recombination regions of hydrogen and lean hydrocarbon flames, and the full zones of both rich and lean hydrogen flames. Gardner [15] focused on the combustion chemistry of nitrogen, sulfur, and chlorine in gas phase. Frenklach *et al.* [16] utilized a method of systematic optimization solution mapping to determine an optimal set of parameters for a methane combustion mechanism. Dagaut *et al.* [17] studied the oxidation of TR0 kerosene in a jet-stirred reactor (JSR), and performed a kinetic analysis to identify

the dominant reaction steps of the mechanism. In the conditions of intermediate temperature and high pressure, the solution showed HO_2 radicals play an important role as chain carriers, which lead to the formation of the branching agent H_2O_2. Marinov *et al.* [18] performed detailed chemical kinetic modeling to analyze aromatic and polyaromatic hydrocarbon pathways in methane and ethane premixed flames. Curran *et al.* [19] developed a detailed chemical kinetic mechanism to investigate the oxidation of n-heptane in flow reactors, shock tubes, and rapid compression machines. The sensitivity analysis indicated that a low-temperature chemistry is very sensitive to formation of stable olefin species. Ranzi *et al.* [20] discussed kinetic modeling and application of extended kinetic schemes, and argued that extension requires only a relatively limited set of independent elementary kinetic parameters. They combined the low- and high- temperature mechanisms of the oxidation process into an extended kinetic scheme to simulate oxidation of natural gas, commercial gasolines and jet fuels, and provided several examples to demonstrate the reliability and effectiveness of these mechanistic schemes. Simmie [21] reviewed the status of detailed chemical kinetic models for intermediate- to high-temperature oxidation and ignition combustion of hydrocarbons, and validated these models with experiments.

Different chemical mechanisms focus on different specific applications; and within the uncertainties of mechanism fundamental studies, the elementary reaction rate parameters must match fundamental rate measurement and computations. The importance of elementary reaction is different in different applications, and the more detailed the chemical mechanism, the broader the range of application. Thus, some mechanisms for combustion processes contain thousands of elementary reaction steps and hundreds of chemical species, as mentioned by Petrova and Williams [22]. The disadvantages of complete mechanisms is that detailed mechanisms in combustion applications require significant computational sources, especially for turbulent combustion; and this is prohibited by limited computational facilities nowadays. For

example, there are 38 elementary reactions in the formation of water, and the analysis and solution of these elementary reaction is a difficult and time-consuming process, as shown by Warnatz *et al.* [23]. It has been shown that many of elementary reactions have minor effects on the reaction process, and these reactions can be ignored in an approximation (see Petrova and Williams [22]). An appropriate reduced mechanism can reveal most important information of chemical reactions and significantly reduce computing processes. Investigating and determining appropriate reduced mechanisms for the DDS model at the non-premixed flames is one part of task in this work.

Numerical computation of combustion processes is rapidly growing with the development of high-performance computer power. Reasonable accuracy must be assured before employing the numerical results, and it is a good approach to achieve the accuracy by basing the numerical calculation on a correct detailed chemical-kinetic mechanism. This thesis focuses on hydrogen-air chemical reactions; there are fewer species and fewer elementary reactions involved than any other fuel oxidation chemical mechanism. Thus, it is easier to obtain reasonable accuracy in hydrogen-air combustion mechanisms. In the well-known San Diego mechanism (Saxena and Williams) [24], eight species and twenty one reversible elementary reactions in the hydrogen oxidation mechanism with reasonable rate parameters for all of these steps have been applied. However, even in this relatively simple mechanism, combustion processes in turbulent flow and complex geometric configurations with a full chemical mechanism are beyond the current computational capabilities for three space dimensions. Therefore, developing a systematically reduced hydrogen-oxygen mechanism that has sufficient accuracy to produce reliable numerical results is very helpful.

1.3 Numerical Approach

1.3.1 Effects of Turbulence

In the past several decades, numbers of reduced mechanisms for hydrogen-oxygen chemistry have been derived from one specific combustion process, such as developed separately for diffusion flames, premixes flames, and for autoignition. For example, Williams [25] derived separate reductions for autoignition, similarly, Mauss [26] and Seshadri [27] obtained separate reductions for laminar deflagration. Fernández-Galisteo [28] [29] presented a one-step overall mechanism systematically that derived for sufficiently lean deflagrations. Guthiel [30] and Balarkrishnan [31] provided reductions for laminar diffusion flames. These reduced mechanisms can efficiently solve one specific problem, and they are limited to only one combustion process. However, autoignition and flames may occur simultaneously in turbulent combustion, or in the transition from deflagration to detonation; and reduced chemistry is required due to computer limitations for numerical computation. To fill this need, Boivin *et al.* [32] developed a systematically reduced chemistry that encompasses autoignition and flames for hydrogen-air chemistry. In terms of general computational approaches, sufficiently accurate reduced chemistry that contains all of these combustion processes is needed because it is hard to predict, in advance, what manner combustion will develop. The systematically reduced description of hydrogen-oxygen chemistry derived in [32] can be applied to all of these combustion processes with acceptable accuracy.

The present thesis applies a small, yet detailed mechanism to simulate non-premixed turbulent hydrogen-air co-flow from a nozzle, and the numerical results will be verified with experimental data. The chemical mechanisms presented here include two individual ones: first is hydrogen-oxygen chemistry with N_2 dilution; and second is reacting N_2 producing NO_x emission. This N_2 dilution mechanism contains 15 elementary reactions and 8 species as shown in [24]; and the reacting N_2 mechanisms

contains the nitrogen Zeldovich mechanism, including 21 elementary reactions and 10 species.

1.3.2 Need for Parallelization

The validation of the mechanisms in the context of a turbulent non-premixed jet flame includes use of averaged values and fluctuations of velocity, temperature and the major species, such as H_2, O_2, H_2O, N_2. The aim of this work is to obtain a clear understanding of combustion chemistry scheme under the conditions mentioned in the abstract, as well as to provide a mechanism for use by those investigators in need of manageable and reliable chemical-kinetic descriptions for H_2-fuel. This mathematical model is coded in Fortran 77, with computational work performed on the University of Kentucky high-performance computer (HPC). In order to obtain solutions more quickly, parallelization is applied to the code. The goal of parallel computing is to reduce execution time, resulting in the ability to solve problems that would not be possible with a corresponding serial program; but this leads to a need for more CPUs, memory resources, and many problems scale well to only a limited number of processors. (see MPI course of Dartmouth College [33]).

A high-performance computer, or supercomputer, is a computer at the frontline of contemporary processing capacity, particularly with respect to speed of calculation, which can happen at speeds of nanoseconds, and available random access memory. With increasing computationally-intensive tasks and the requirement of decreasing run time, supercomputers play a more and more important role in computational science fields. Supercomputers are applied in a variety of applications in scientific research and industry design/production, including quantum mechanics, climate research, weather forecasting, oil and gas exploration, molecular modeling and physical simulation. The current thesis focuses on modeling a combustion process, which is a specific application of physical simulation in computational fluid dynamics. The cor-

responding parallel structure in the code is used to improve computational efficiency with potential application in a complete LES code.

There are two types of problem decomposition in the context of parallel programming: domain decomposition and functional decomposition, which can be applied separately or together. In domain decomposition, data are divided into similar size pieces and mapped to different processors; one process can communicate with other processors when necessary; and the processor works only on data assigned to it. In functional decomposition, a program is decomposed into a number of small tasks; the pieces of data require different processing times, and the computation performance is limited by the slowest process (see [33]).

There are two typical ways to parallelize a programming language. One is directive based parallel programming language, with Open Multi-Processing (OpenMP) being the most widely used. This programming language utilizes directives to tell processors how to distribute data and work across the process, and it appears as comments in an originally serial source code. It is usually implemented on shared memory architectures. A second one is message passing interface (MPI), which passes messages to send/receive data between processes, and each process has its own local variables. It can be used on either shared or distributed memory architectures (see [33]).

The characteristics of MPI and OpenMP are established by their own structures. The advantages of MPI include: MPI can run on either shared or distributed memory architectures, so it can solve a wider range of problems than OpenMP; each process has its own local variables; distributed memory computers are much cheaper than large shared-memory computers. However, MPI needs more programming changes to go from serial to parallel versions, and it is harder to debug than OpenMP; moreover, its performance is limited by communication networks between the nodes. In contrast to MPI, OpenMP is easier to program and debug, and its directives can be added incrementally, allowing gradual parallelization. In OpenMP, a program can be run as

a serial code and the serial code statements do not need modification; moreover the code is easier to understood and more easily maintained; and it is mostly used for loop parallelization, as noted by [33].

In the University of Kentucky HPC center has 16 cores per node. Thus, OpenMP can use 16 cores at one time, at most, in this local shared memory system. In terms of theoretical calculation of computation time, 16 cores can speed calculation of parallelization parts to 16 times, and for a relatively small task, 16 times the speed of the original one is fast enough. The whole execution time in OpenMP includes computation time, idle time (waiting for data from other processors) and communication time (time processors take to send and receive messages). The idle time and communication time will delay the whole calculation time. Thus, minimizing communication by reducing the number of messages, and reducing the amount of data passed in messages, is an efficient approach to reduce communication time. In this thesis, the computational load of each task is not large; therefore OpenMP is a good choice for parallel computing here.

1.4 Motivation

It is not a wise choice to directly simulate a turbulent combustion process, such as directly calculating discrete momentum equations, even with a parallelization programming. More efforts have been spent on improving numerical analysis of the interaction between chemistry and turbulence for combustion flames especially for turbulent flames, in the last several decades since there is still a large amount of computational work in turbulent combustion processes without a simplified turbulence model, as mentioned by Carbonell *et al.* [34]. In the following several paragraphs, this thesis will review the history of turbulence model development, and discuss both advantages and disadvantages of each model to provide a setting and motivation for the method studied in this thesis.

The earliest recognition of turbulence as a physical phenomenon began at the time of Da Vinci (*circa* 1500). But there was no substantial progress in understanding turbulence until Boussinesq provided the Boussinesq hypothesis [35] in the late 19^{th} century. His hypothesis mentioned that "*turbulent stresses are linearly proportional to mean strain rates, with the constant of proportionality being the (non-physical) eddy viscosity*", which is very popular in most turbulence models, at least in part due to the analogy with Newton's law of (physical) viscosity. The Navier–Stokes (N.–S.) equations are a classical mathematical description of motion of fluid substances, and it is believed that these equations properly illustrate the complexities of turbulent behavior, see Lerner and Trigg [36]. Nevertheless, analytical solutions to the N.–S. equations, even for the simplest turbulent flows, do not exist due to the nonlinearity that is caused by convective acceleration. Mixtures of chaos and order, and the wide range of length and time scales in turbulent flow, make turbulence "*one of the seven most important open problems in mathematics,*" as claimed by the Clay Mathematics Institute. This institute has offered a one-million US dollar prize to the researcher who first proves long-time existence of high Reynolds number (hence turbulent) solutions to the N.–S. equations [37]. Turbulence is one of the toughest problems in classical physics; some scientists view turbulence as "*chief outstanding problem of our subject,*" first stated by Lamb [38] in the second edition of *Hydrodynamics*; and "*invention of the Devil on the seventh day of creation,*" declared by Bradshaw [39] in 1994.

The works of Reynolds, Prandtl, Taylor and others emphasized that statistical approaches were the only possibility for analyzing the randomness of turbulent flows, and this opinion was very popular in the development of turbulence modeling procedures for simplifying the N.–S. equations, see details in McDonough [40]. The modeled equations describe the statistical evolution of the flow containing terms that cannot be obtained from the N.–S. equations, and therefore they require modeling. With increasing complexity of turbulent flow, in order to get reasonable numerical

results, improvement of turbulence models is needed.

Typically, there are three main approaches to turbulence calculation: direct numerical simulation (DNS), large-eddy simulation (LES), and Reynolds-average Navier–Stokes (RANS) modeling. In DNS, the N.–S. equations are numerically highly resolved to accurately simulate the flow, and its computational cost is high as all length scales and time scales have to be resolved; this approach is limited to flow with low to moderate Reynolds number. In LES, the equations are solved with a filtered velocity field, which represents the larger-scale parts; the remaining parts of the smaller-scale motions are not directly represented, which are calculated later with a sub-grid scale (SGS) model, presented by Lesieur [41] and Kravchenko [42]. LES applies local spatial filtering to all appropriate variables (spatial rather than temporal) and the *LES decomposition* of $\boldsymbol{u}(\boldsymbol{x}, t)$ is

$$\boldsymbol{u}(\boldsymbol{x}, t) = \tilde{\boldsymbol{u}}(\boldsymbol{x}, t) + \boldsymbol{u}'(\boldsymbol{x}, t). \tag{1.1}$$

In this decomposition $\tilde{\boldsymbol{u}}$ is usually termed the large- or resolved-scale part of the solution, and $\boldsymbol{u}'$ is called the small-scale, or subgrid-scale, or unresolved part.

Approaches based on RANS models are the most prevalent currently, and in RANS all scales of the solution must be modeled under a RANS formalism; but the time mean quantities are directly computed, as showed by Speziale [43]. The *Reynolds decomposition* of $\boldsymbol{u}(\boldsymbol{x}, t)$ is

$$u(\boldsymbol{x}, t) = \bar{u}(\boldsymbol{x}) + u'(\boldsymbol{x}, t), \tag{1.2}$$

where $u'(\boldsymbol{x}, t))$ is termed the "fluctuating part," and $\bar{u}(\boldsymbol{x})$ is the time averaged value and thus, independent of time. The total arithmetic of a model affects its calculation expense, and McDonough [40] summarized the predicted arithmetic for these three models. DNS requires no modeling, which has a total arithmetic scaling at least

as Re^3; and the total arithmetic will scale no worse than Re^2 for LES; and the total arithmetic is at most a weak function of Re for RANS. This thesis will briefly introduce the history and development of the three classical turbulence models.

The flow variables, velocity and pressure are a function of space and time in a complete description of a turbulent flow, which can only be obtained exactly by numerically solving the N.–S. equations with DNS. Moin and Mahesh [44] stressed that DNS was a research tool, and it was not appropriate for a brute-force solution to the N.–S. equations for engineering problems. They discussed related numerical issues, such as boundary conditions, and spatial and temporal discretization, and used DNS data to evaluate accuracy of experimental measurements.

Direct simulation eliminates the need for ad hoc models, and the advanced justification for completed numerical resolution is that the statistics of the large scale can be found at low Reynolds number and vary little with Reynolds number (see Rogallo [45]). DNS is a very useful tool for turbulence research, and it complements the time-trusted methodology of experimental research. Fox and Lilly [46] started the foundations research of turbulence in two dimension at the National Center for Atmospheric Research in the year 1971. They mentioned that numerical simulation of turbulent flow was important in practical geophysics, and it was useful as a turbulence theory test. They believed that directly simulating the larger scales of motion, and only considering the small unresolved scales for their gross statistical interactions with larger scales, was a more useful method for practical applications. Orszag and Patterson [47] used a 32^3 computation of three-dimensional homogeneous, isotropic turbulence in incompressible fluid at a Re_λ of 35 as an initial DNS application in 1972. They presented a preliminary report of these numerical simulations, and compared the results with predictions of turbulence theories; and the calculations demonstrated that spectral methods could be used to perform large-scale computations of turbulence in 3-D. Rogallo [45] made significant progress in direct simulation methods,

and he extended the Orszag and Patterson algorithm from homogeneous, isotropic turbulence to incompressible fluid subjected to uniform deformation and rotation in 1981. His study provided the results of irrotational strain, shear, rotation, and relaxation toward isotropy following axisymmetric strain, and he compared the numerical results with linear theory and experimental data. Rogallo applied the computed results to assess accuracy of models that were used in the closure of the Reynolds-stress equations, and his work set the standard for DNS of homogenous turbulence.

Due to absence of flow boundaries, homogeneous flows are easier to achieve in numerical computation than inhomogeneous flows. The earliest computation for inhomogeneous flows were only performed in one dimension. Coarse-grid computations of free-shear layers without wall-bounded turbulence was performed by Riley and Metcalfe [48] in the late 1970s. Direct numerical methods have been successful for unbounded flows, where viscosity was used to set the scale of dissipative eddies; however, it has not been successfully for wall-bounded flows, such as a turbulent channel flow, see Rogallo and Moin [49]. Later, in 1987, Kim *et al.* [50] performed a direct simulation of a turbulent channel flow, where all essential turbulence scales of motion were resolved on the computational grid, and no SGS model was used. Kim *et al.* reported the computed results of turbulence statistics and compared them with the existing experimental data at a comparable Reynolds number. They also investigated the behavior of turbulence correlations near the wall, and presented a number of statistical correlations, which were complementary to the existing experimental data for the first time. Moser and Moin [51] then simulated a low Reynolds number curved turbulent channel flow by direct numerical solution of the N.–S. equations. The resulting flow fields were used to study the effects of streamline curvature via comparing the concave and convex sides of the channel. Since then, channel flow has became a fundamental phenomenon for wall-bounded turbulence study.

Subsequent studies of wall-bounded turbulence in channel configurations included

other phenomena, such as heat transfer, rotation, transverse curvature, and transpiration. Kasagi *et al.* [52] carried out a fully-developed thermal field DNS in a two-dimensional turbulent channel flow to investigate air heat transfer. Statistical data such as root-mean-square temperature fluctuations, turbulent heat fluxes, turbulent Prandtl number, and dissipation time scales were obtained in their research. These researchers also calculated budget equations of temperature variance, dissipation rate, and turbulent heat fluxes. Kristoffersson and Anderson [53] performed a DNS of fully-developed pressure-driven turbulent flow in a rotating channel. They found that the number of vortex pairs tended to increase with rotation number. Neves and Moin [54] identified effects of transverse curvature by comparing their simulated results with those of plane channel simulation of Kim [50], and found that as curvature increases, skin friction increases, and the slope of the logarithmic region decreases. Neves and Moin [55] also studied effects of convex transverse curvature on wall pressure fluctuations through DNS. Sumitani and Kasagi [56] employed a DNS of fully-developed turbulent channel flow to analyze heat transfer with uniform wall injection and suction. Spalart [57] made progress on boundary-layer simulations by developing a method to compute turbulent sink-flow boundary-layers with favorable pressures gradient. Spalart [58] also provided numerical simulation of a turbulent boundary layer on a flat plate with zero pressure gradient; and his boundary-layer data have been widely used in the turbulent flat-plate boundary layer field by scholars and engineers.

Moin and Mahesh [44] described the history of DNS development, and they stated that DNS of compressible turbulent flows has followed the steps of incompressible flows. Feiereisen *et al.* [59] accomplished the initial study of DNS of homogeneous compressible turbulent shear flow with low Reynolds number in the early 1980s. A decade later, a series of studies about homogeneous compressible turbulent flows was carried out. Erlebacher *et al.* [60] investigated compressible turbulent flows at low

turbulent Mach numbers, and Sarkar *et al.* [61] scrutinized compressibility effects on homogeneous turbulent shear flow. Lee et al. [62] studied the existence of eddy shocklets in 3-D compressible turbulent flow. Blaisdell *et al.* [63] checked compressibility effects within decaying isotropic turbulence and homogeneous turbulent shear flow, and their work increased understanding of compressible turbulence and helped development of turbulence models for compressible flows. For wall-bounded flow analysis, Coleman *et al.* [64] performed a study of compressible supersonic turbulent flow with isothermal walls in a plane channel; Rai et al. [65] first described a compressible, turbulent, supersonic, spatially evolving boundary-layer flow by DNS.

DNS is the optimal method for flow with low to moderate Reynolds numbers; however, the goal to simulate a turbulent flame by DNS is very difficult to achieve due to large computational expenses and memory requirements. The ranges of scales in turbulent flows increase rapidly with Reynolds number, and most problems in engineering applications have too wide a range of scales to be directly computed by DNS. There are several options for turbulent flow models with high Reynolds numbers besides DNS, such as RANS and LES.

In terms of additional partial differential equations that one must solve beyond those of N.–S. equations, there are zero-equation, one-equation, and two-equation models, and half–equation models for the classification of RANS methods. The half–equation models contain a single ordinary differential equation in their formulation, see Wilcox [66]. The k-ε models are the most widely used two-equation models, where k is turbulence kinetic energy and ε is turbulence kinetic energy dissipation rate. The formula for *turbulence kinetic energy* is

$$k = \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right), \tag{1.3}$$

and ε expressed in Cartesian tensor form as,

$$\varepsilon = 2\nu\overline{s'_{ij}s'_{ij}}, \tag{1.4}$$

where ν is kinematic viscosity, and s'_{ij} are components of the fluctuating strain rate tensor given by

$$s'_{ij} = \frac{1}{2}\left(\frac{\partial u'_i}{\partial x'_j} + \frac{\partial u'_j}{\partial x'_i}\right). \tag{1.5}$$

Solutions obtained from k-ε methods are not very close to experimental results (see, e.g. Freitas [67]), but they tend to be somewhat better than these of zero- and one-equation models. However, k-ε methods have been widely used in predicting many flow behaviors for industrial applications, such as engine performance. In addition, they currently comprise the most widely used turbulence models for combustion system optimization in industry.

Retracing the history of RANS, Boussinesq [68] first introduced *turbulent eddy viscosity* in his hypothesis, which is the basis for a simple time-averaged turbulence closure. In the later 19th century, a paper served as a landmark contribution to the development of fluid mechanics was published by Osborne Reynolds [69], which put forward the concept of Reynolds averaging. Thanks to the pioneering work of Prandtl [70], a practical turbulent flow calculation based on RANS equations with an eddy-viscosity model was first successfully achieved. Prandtl introduced the concept of *mixing-length theory* to determine eddy viscosity, and attempted to formally derive turbulent eddy viscosity appearing in the Boussinesq hypothesis. The closed form solutions of mixing length models are very successful for turbulent pipe and channel flows. Many researchers, especially von Kármán [71] [72], made further investigations on the mixing-length approach, and greatly moved forward research progress on mixing length models before the mid-20th century. At this time, researchers realized that the basic assumptions of mixing-length were unrealistic, because turbulent flow

scales did not show clear-cut separation characteristics. In order to develop more general models, eddy viscosity was applied to turbulent kinetic energy by Prandtl [73]. This was the predecessor of one-equation model of turbulence, that is the so-called k-l models, wherein the turbulent length scale l is obtained from specified empirical data; and the turbulent kinetic energy k comes from a modeled transport equation. These models did not work for all eddy-viscosity models, and they were unable to accurately mimic body forces, streamline curvature, and history effects of individual Reynolds-stress components.

In the early 1950s, Rotta [74] introduced the important theory of a full Reynolds-stress turbulence closure, and mentioned the correlations between fluctuations of pressure and velocity derivatives. This theory was a landmark contribution to turbulence modeling, and it changed the course of Reynolds-stress modeling permanently. The approach in [74] was based on the Reynolds-stress transport equation, and it was regarded as a second-order or second-moment closure. This Reynolds-stress closure traced both history and nonlocal effects on the evolution of the Reynolds-stress tensor. However, there were six additional transport equations in this second-order closure for individual components of the Reynolds-stress tensor, and the computer capacity was not sufficient to treat complex flows, especially combusting flows, based on such a closure at that time. With the development of high-speed computers, Daly and Harlow [75], and Donaldson [76] employed the second-order closure models again in the 1970s. Launder *et al.* [77] significantly improved earlier work by Rotta [74]. Later, a two-equation model, the so-called k-ε model, was obtained by reducing Launder's model and supplementing an eddy-viscosity representation for the Reynolds stress. The two-equation model, k-ε model, is still one of the most commonly-used turbulence models for engineering applications due to its low computational effort requirement.

Various second-order closures were proposed after the Launder *et al.* work. Lumley [78] believed that second-order closure modeling made it possible to treat many

practical situations, and he introduced significant contributions to modeling of the pressure-strain correlation and buoyancy terms. Speziale [79] examined physical properties of the commonly-used second-order closure models for rotating turbulent flows, and compared the results with solutions of the N.–S. equations of fully-developed turbulent channel flow in a rapidly rotating structure. Pope [80] developed a Langevin model appropriate to constant property turbulent flows from a general transport equation which was consistent with the second-order turbulence closure models. Haworth and Pope [81] attempted to determine the form of a second-order tensor appearing in the general model equation, and they evaluated this tensor by considering evolution of Reynolds stresses in homogeneous flows. Speziale [82] considered the zero-, one-, and two-equation models along with second-order closures, and discussed the development of models from two approaches: the continuum mechanics approach, and the statistical mechanical approach.

There was another two-equation model, the re-normalisation group (RNG) k-ε model, which was developed by Yakhot *et al.* [83]. This model was used to account for effects of smaller scales of motion in the N.–S equations. Unlike the standard k-ε model, which determined the eddy viscosity from a single turbulence length scale and calculated the turbulent diffusion at the specified scale, the RNG approach applied a modified form of the ε equation, which accounted for different scales of motion through changes to the production term. Wang *et al.* [84] proposed a generalized RNG closure model based on *dimensionality* of flow strain rate to improve predictions of turbulence qualities for compressible flows. The numerical turbulence energy of internal combustion engine flows was improved significantly, and the calculated tip penetrations matched the experimental data well, in this generalized RNG closure model. Two years later, Wang *et al.* [85] applied the RNG turbulence model to simulate non-reacting flows in a single-cylinder PFI engine. They compared the velocity fields of numerical results with experimental data from particle image velocimetry

(PIV) measurements, and good agreement was found between them.

RANS methods have been successful in predicting some gross features of combustion, such as profiles of combustor exit temperatures, whereas, these are unable to predict transient phenomena, such as flameout, relight and combustion instabilities in gas turbines and afterburners, cycle-to-cycle variations in IC engines, and pollutant formation, as noted by Fedina and Fureby [86]. The time-dependent nature of such flows can be resolved with LES, which provides a very good natural framework for simulation of performance of combustion equipment. In contrast, the goal of RANS modeling in such situations is just to produce averaged scalar fluxes whose overall effect is close to a "smearing" over time of the actual physics. This is unacceptable in many combustion studies and applications .

Typical chemical reaction rates can be expressed in the well-known form of the Arrhenius law (see, e.g. Warnatz *et al.* [23]):

$$k(T) = AT^n \exp\left(\frac{-E_a}{R_0 T}\right), \tag{1.6}$$

where E_a is activation energy; R_0 is the universal gas constant; T is absolute temperature; and A and n are empirical constants. This form is extremely nonlinear, and must be averaged in the context of the RANS formalism, or filtered in the typical LES, as noted by McDonough [40]. It is clear that

$$\begin{aligned} \overline{k}(T) &= \overline{AT^n \exp\left(\frac{-E_a}{R_0 T}\right)} \\ &\neq A\overline{T}^n \exp\left(\frac{-E_a}{R_0 \overline{T}}\right) = k(\overline{T})\,, \end{aligned}$$

and the lack of equality is so severe that the second formula on the right simply cannot be used. Moreover, most chemical reactions are sensitive to concentration of chemical species; thus, we cannot directly use time averaged species concentrations

to replace temporal fluctuations of species concentrations.

LES is an alternative method to RANS in combustion modeling. In many commercial CFD software suites, LES is now an available option for practical engineering application, even though its arithmetic is close to Re^2 which is still a challenge for modern computing machinery, as noted by McDonough [40]. In LES, the large-scale, energy-carrying motion is directly resolved on the grid, while the small-scale is modeled. The usual LES decomposition has been mentioned earlier. It is important to note that the resolved and unresolved scales depend on both space and time, and this is a major distinction and advantage in comparison with the Reynolds decomposition and resulting RANS methods.

The history of LES can be traced back to 1960s, when a meteorologist, Smagorinsky [87], proposed what is now called the Smagorinsky model. The Smagorinsky model is simple to implement and will stabilized a computation, but it failed in the prediction of atmospheric and oceanic flows for which it was intended, since it dissipated the large-scale too much. In the 1970s, the concept of spectral eddy viscosity was developed by a physicist, Robert Kraichnan, and this concept allowed modeling to proceed beyond the separation of scales assumption. Lesieur and Métais [41] implemented Kraichnan's spectral eddy viscosity in physical space to get a structure-function model, and applied a double filtering to dynamically determine subgrid-scale model constants. They also used scale-similarity models to replace the eddy-viscosity assumption. Moin and Kim [88] examined flow structure and studied statistical properties of flow as well as its time-dependent features in the vicinity of the wall of the fully-developed plane channel flow in LES. Then, Rogallo and Moin [49] used the results to study physics of near-wall turbulence. Meneveau and Katz [89] reviewed models that were based on scale-invariance properties of high-Reynolds number turbulence in the inertial range for LES, and evaluated model performance in numerical simulations.

A book which limits itself to the case of incompressible fluid, *Large Eddy Sim-*

ulation for Incompressible Flows–An Introduction published in 2001 by Sagaut *et al.* [90] exhaustively described all of sub-grid modeling methods for simulating the large scales of incompressible turbulence. Branley and Jones [91] applied LES to a calculation of a turbulent hydrogen diffusion flame with a conserved-scalar formalism; their simulated results showed LES can produce good agreement with measurements of variables, such as mean velocity, Reynolds stress and fluxes. Volavý *et al.* [92] investigated the most-used sub-grid scale (SGS) models, such as Smagorinsky model, dynamical Smagorinsky, sub-grid kinetic energy, dynamic sub-grid kinetic and mixed models, and they compared results of these models with the corresponding DNS data. Mahle *et al.* [93] applied an approximate deconvolution as an implicit SGS model for LES of turbulent combusting shear layers with hydrogen chemistry, and they studied influence of detailed diffusion mechanisms on laminar flamelets. Pitsch [94] discussed fundamental differences between RANS and LES combustion models for non-premixed and premixed turbulent combustion. He investigated LES modeling issues, and proposed ways to improve future LES model. Fureby [95] applied different LES models, such as the flamelet progress-variable model, the thickened flame model, the eddy dissipation concept model, and the partially-stirred reactor model to examine the performance of a swirl-stabilized premixed flame in a laboratory gas turbine combustor. The comparison between LES models and experimental data illustrated that all four LES models result in reasonable predictions of flow and combustion physics.

Mathew [96] summarized the general formulation and discussed the common modeling for flows without reaction in LES, and he also made an extension to flows with combustion in LES. The only thing typical SGS models do is to produce enough dissipation to control the aliasing effects arising from under resolution on large scales. The transfer of energy is disrupted because the smallest resolved scales of motion are much larger than the dissipation scales. To treat this LES problem, the unresolved

scales in LES should be the order of the grid spacing, or even smaller; thus the model is called subgrid-scale model. A sub-grid model provides a method to calculate sub-grid stresses which are the terms arising from the nonlinear convection terms in the momentum equation. There are two classes of SGS model: functional models and structural models. The function models are simpler than structural models, which provide dissipation as a model for the transfer of energy to small-scales. The structural models provide an estimate of the full field to find the sub-grid scales, and calculate SGS stress.

Most effort has been spent on investigating SGS modeling in LES studies, especially in the context of finite-rate chemistry. Moussaed *et al.* [97] investigated the effects of a dynamic SGS model in variational multiscale LES simulations. They used a variational projection operator and finite-volume cell agglomeration to obtain the separation between the largest and the smallest resolved scales. Gubba *et al.* [98] derived a dynamic SGS model for LES of turbulent premixed flames of stoichiometric propane/air mixtures in a vented combustion chamber. This model was based on fractal theory and a flame wrinkling factor, and the simulation results showed good agreement with experimental measurements.

With a single universal constant, the common eddy viscosity SGS stress models were unable to correctly represent various turbulent fields in rotating or sheared flows, or in transitional regimes, as discussed by Germano *et al.* [99]. These authors computed the model's coefficient dynamically as the calculation progresses instead of setting it a priori. This model was based on an algebraic identification between SGS stresses and resolved turbulent stresses. The results for LES of transitional and turbulent channel flow using this proposed model showed good agreement with direct simulation data.

We show a typical LES-like decomposition of solution variables formalism here

again, but with somewhat different notation:

$$\boldsymbol{Q}(\boldsymbol{x},t) = \boldsymbol{q}(\boldsymbol{x},t) + \boldsymbol{q}^*(\boldsymbol{x},t), \quad \boldsymbol{x} \in \mathbb{R}^d, \quad d = 2,3, \tag{1.7}$$

where $\boldsymbol{q}(\boldsymbol{x},t)$ is the resolved large-scale part, and $\boldsymbol{q}^*(\boldsymbol{x},t)$ is the unsolved small-scale part. Zeng *et al.* [100] used a basic hypothesis for SGS model, where the small-scale variables $\boldsymbol{q}^*(\boldsymbol{x},t)$ in Eq. (1.7) are expressed as

$$q_i^* = A_i M_i, \qquad i = 1,2,\ldots,N_v, \tag{1.8}$$

with N_v being the total number of dependent variables; q_i^* is the i^{th} component of the N_v small-scale dependent variables; the A_is are amplitudes derived from scaling laws of Kolmogorov (see, e.g., Frisch [101]); and the M_{is} are chaotic maps that can exhibit bifurcations leading to a strange attractor which produces small-scale turbulent temporal fluctuations locally in space and time [40]. McDonough and Zhang [102] [103] proposed this method as a 2-D SGS model; we extend it to 3D for LES in this thesis. This method includes the well-known logistic map, which was first presented by May [104] in the 1970s. The logistic map,

$$m^{(n+1)} = \beta m^{(n)}(1 - m^{(n)}), \tag{1.9}$$

is a widely-used, simple model with complicated dynamics. Later, Frisch [101] discussed a simple quadratic map,

$$x^{(n+1)} = 1 - 2\big(x^{(n)}\big)^2, \tag{1.10}$$

which can be transformed to Eq. (1.9), and called this equation the *'poor man's Navier–Stokes equation'*, as this equation has low calculational expense when presenting features including temporal behaviors of the partial differential equation. McDonough and Huang [105] derived the 2-D 'poor man's Navier–Stokes equation' directly from the analytical partial differential equations via a Galerkin procedure, and

Polly [107] investigated the corresponding 3-D case,

$$a^{(n+1)} = \beta_1 a^{(n)}(1 - a^{(n)}) - \gamma_{12} a^{(n)} b^{(n)} - \gamma_{13} a^{(n)} c^{(n)}, \tag{1.11a}$$

$$b^{(n+1)} = \beta_2 b^{(n)}(1 - b^{(n)}) - \gamma_{21} a^{(n)} b^{(n)} - \gamma_{23} b^{(n)} c^{(n)}, \tag{1.11b}$$

$$c^{(n+1)} = \beta_3 c^{(n)}(1 - c^{(n)}) - \gamma_{31} c^{(n)} a^{(n)} - \gamma_{32} c^{(n)} b^{(n)}. \tag{1.11c}$$

1.5 Outline of Thesis

In this work, we use an analogous approach to derive a finite-rate chemistry SGS model in 3D, which includes the momentum and thermal energy equations similar to earlier studies in 2D reported in [102] [103] [105]. Zeng *et al.* [100] presented a preliminary exploration of behaviors of this discrete dynamical system (DDS) for a specific reduced-kinetics mechanism of H_2-air combustion at a single point. We will investigate this SGS model at multiple points, corresponding to the experimental data of Schneider *et al.* [106], and assess validity of this model based on comparisons with these data.

In the next several chapters, we present the governing equations and assumptions for the SGS model, derive the corresponding DDS for two different reduced mechanisms via a single-mode Galerkin approximation, and finally present results and compare numerical solutions with experimental data. Both isotropic and anisotropic conditions will be applied to calculate the bifurcation parameters in the DDS model; a physical temperature model and a scaling method for constructing small scales of motion is applied; and the effects of NO_x in high-temperature combustion will be taken into consideration.

Chapter 2 Discrete Dynamical System Model Analysis

Anthony *et al.* [108] discuss dynamical systems and briefly mention their classification. A *dynamical system* is a four-tuple $\{\boldsymbol{T}, \boldsymbol{X}, \boldsymbol{A}, \boldsymbol{S}\}$, which includes a *time set*, $\boldsymbol{T}$, the *state-space*, $\boldsymbol{X}$, the set of *initial states*, $\boldsymbol{A}$, and a *family of motions*, $\boldsymbol{S}$. When *time set*, $\boldsymbol{T}= \mathbb{R}^+= [0, \infty)$ the system is a *continuous-time dynamical system*; and when $T= \mathbb{N}= \{0, 1, 2, 3, ...\}$, the system is a *discrete-time dynamical system*. When the *state-space* $\boldsymbol{X}$ is a finite-dimensional normed linear space, the system is a *finite-dimensional dynamical system*; otherwise, the system is an *infinite-dimensional dynamical systems*.

Discrete dynamical systems (DDS) are commonly used in many fields such as biology, ecology, economics, engineering, physics, finance, etc. In this thesis, the variables in a three-dimensional, first order, nonlinear system of difference equations will be analyzed.

2.1 Governing Equations

In this section, the derivation of a general discrete dynamical system beginning with the governing equations of combustion chemistry, which include mass conservation, momentum balance, and energy and species transport equations is performed. They are

$$\rho_t + \nabla \cdot (\rho \boldsymbol{U}) = 0, \tag{2.1a}$$

$$\rho\frac{D\boldsymbol{U}}{Dt} = -\nabla p + \nabla\cdot(\mu\nabla\boldsymbol{U}) + \rho\boldsymbol{g}, \tag{2.1b}$$

$$\rho c_p\frac{DT}{Dt} = \nabla\cdot(\lambda\nabla T) + \sum_{i=1}^{N_s} c_{p_i}D_iW_i\nabla\left(\frac{\rho Y_i}{W_i}\right)\cdot\nabla T - \sum_{i=1}^{N_s} h_i\dot{\omega}_i, \tag{2.1c}$$

$$\rho\frac{D(Y_i)}{Dt} = \nabla\cdot(\rho D_i\nabla Y_i) + \dot{\omega}_i, \qquad i = 1,\dots N_s. \tag{2.1d}$$

Here,

$$\dot{\omega}_i = W_i\sum_{j=1}^{N_r}(\nu''_{i,j} - \nu'_{i,j})\omega_j\,, \tag{2.2}$$

with

$$\omega_j = k_{f,j}\prod_{l=1}^{N_s}\left(\frac{\rho Y_l}{W_l}\right)^{\upsilon'_{i,j}} - k_{b,j}\prod_{l=1}^{N_s}\left(\frac{\rho Y_l}{W_l}\right)^{\upsilon''_{i,j}}. \tag{2.3}$$

These equations hold on a 3-D spatial domain $\Omega\in\mathbb{R}^3$ during a specified time interval $t\in(t_0,t_f)$, and $\boldsymbol{U}=(u,v,w)^T$; D/Dt is the substantial derivative; ∇ is the gradient operator; $\boldsymbol{g}$ is the body-force acceleration vector, ρ is density, and p is the pressure. T is temperature; Y_i is the mass fraction, and h_i is specific enthalpy, of species i. The transport properties include (dynamic) viscosity μ, thermal conductivity λ, and the binary diffusion coefficient D_i of species i in the ambient background gas. Here, c_{p_i} and W_i are the specific heat capacity and relative molecular mass of species i, respectively; $\nu'_{i,j}$ and $\nu''_{i,j}$ are stoichiometric coefficients of reactants and products corresponding to species i in reaction j. N_s and N_r are the number of species and reactions, respectively. Finally, $k_{f,j}$ and $k_{b,j}$ are the forward and backward reaction rate coefficients of the j^{th} reaction. The reaction rate expression was shown in Eq. (1.6), and the specific form for the j^{th} reaction is:

$$k_j = A_jT^{n_j}\exp\left(\frac{-E_j}{R_0T}\right).$$

Recall that in LES the large-scale part has been resolved directly, so we now propose to construct corresponding DDS SGS models from the governing equations. We would

then add the SGS solutions of the latter to the resolved solution to construct an approximation to the complete solution. But in the present work, because the DDSs are computed locally, and away from boundaries, in the absence of resolved-scale information, we will not analyze boundary conditions for the chosen spatial domain.

2.2 Construction of the Dynamical Systems

In this subsection, the deviation of the DDS model will be presented thoroughly. We begin with the mass conservation equation Eq. (2.1a) and the Navier–Stoker Eqs. (2.1b)

$$\rho_t + \nabla \cdot (\rho \boldsymbol{U}) = 0, \tag{2.4}$$

$$\rho \frac{D\boldsymbol{U}}{Dt} = -\nabla p + \nabla \cdot (\mu \nabla \boldsymbol{U}) + \rho \boldsymbol{g}. \tag{2.5}$$

For incompressible flow, the density is identically constant, while gas density in a combustion flow varies with pressure, temperature and species concentration of the combustion process. Generally, a flow is regarded as incompressible if the Mach number is less than 0.3 (in this work, the Mach number is less than 0.1), which means no more than $\sim 10\%$ error will be incurred due to changes in flow density, and the divergence-free condition is satisfied, as noted by McDonough [110]. On the other side, the temperature is always changing during chemical reactions processes, and we could not treat the gas density as a constant in this diffusion flames. However, the divergence free condition is still valid when a Leray projection [110] is applied on the conservation equation.

With the divergence-free constraint, the Eq. (2.4) will be replaced with.

$$\nabla \cdot \boldsymbol{U} = 0. \tag{2.6}$$

We express the 3-D dimensionless form of Eqs. (2.5) and Eq. (2.6) in the absence

of body forces in the horizontal plane, and via a typical scaling of dependent and independent variables as,

$$u_x + v_y + w_z = 0, \tag{2.7a}$$

$$u_t + (u^2)_x + (uv)_y + (uw)_z = -p_x + \frac{1}{Re}\Delta u, \tag{2.7b}$$

$$v_t + (uv)_x + (v^2)_y + (vw)_z = -p_y + \frac{1}{Re}\Delta v, \tag{2.7c}$$

$$w_t + (uw)_x + (vw)_y + (w^2)_z = -p_z + \frac{1}{Re}\Delta w - \frac{1}{Fr^2}. \tag{2.7d}$$

where, x, y, and z subscripts denote partial differentiation with respect to spatial variables, and the t subscript denotes partial differentiation with respect to the time variable. Re is the Reynolds number,

$$Re = \frac{UL}{\nu},$$

with U, L and ν denoting the appropriate velocity and length scales, and kinematic viscosity, respectively (see McDonough [110]). The body force term $\rho\boldsymbol{g}$ only remains in the vertical direction, and the corresponding dimensionless Fr appears in Eq. (2.7d); this is the Froude number,

$$Fr = \frac{U}{\sqrt{gL}}. \tag{2.8}$$

where $\boldsymbol{g}$ is the magnitude.

For Eq. (2.7d), we can further simplify the momentum equation by setting the bulk fluid velocity to 0 ($u = 0$), then we obtain

$$\frac{\partial p}{\partial z} = \rho_0 g,$$

where ρ_o denotes the bulk fluid density. Further simplification of the momentum equation by substituting the volume expansion coefficient, and density $\rho_o - \rho =$

$\beta\rho(T - T_o)$, into the momentum Eq. (2.7d) leads to

$$w_t + (uw)_x + (vw)_y + (w^2)_z = \frac{1}{Re}\Delta w + \frac{Gr}{Re^2}T, \tag{2.9}$$

where β is volume expansion coefficient with Grashof number defined as

$$Gr = \frac{g\beta(T_s - T_o)L^3}{\nu^2}.$$

Here, T_s denotes surface temperature, T_o is bulk fluid temperature, and L_c represents a characteristic length.

We then employ a Leray projection method [111] to map $\boldsymbol{u}$ to the divergence-free subspace of solutions, and in the case of solid-wall boundaries, where the usual no-slip/no-flux boundary conditions imposed, this leads to elimination of the pressure gradient terms:

$$\begin{aligned}\langle \nabla p, \boldsymbol{v} \rangle &= \int_\Omega \nabla p \cdot \boldsymbol{v} dV \\ &= \int_{\partial\Omega} p\boldsymbol{v} \cdot \boldsymbol{n} dA - \int_\Omega p \nabla \cdot \boldsymbol{v} dV \\ &= 0.\end{aligned}$$

As a result, the pressure gradient term in Eq. (2.7b) and Eq. (2.7c) is removed.

2.2.1 Galerkin approximation to the governing equations

The pressure gradient teams have been eliminated by the Leray projection, and the 3-D dimensionless form of N.–S. equations will be treated in the absence of body forces here (the detailed 2-D derivation is given in [105]). The effect of body force only remains in the vertical direction for problems considered in this thesis, and the

set of dimensionless governing equations takes the form,

$$u_x + v_y + w_z = 0, \tag{2.10a}$$

$$u_t + (u^2)_x + (uv)_y + (uw)_z = \frac{1}{Re}\Delta u, \tag{2.10b}$$

$$v_t + (uv)_x + (v^2)_y + (vw)_z = \frac{1}{Re}\Delta v, \tag{2.10c}$$

$$w_t + (uw)_x + (vw)_y + (w^2)_z = \frac{1}{Re}\Delta w + \frac{Gr}{Re^2}T, \tag{2.10d}$$

$$\rho c_p\Big(T_t + (uT)_x + (vT)_y + (wT)_z\Big) = \lambda\Delta T + \sum_{i=1}^{N_s}\rho c_{p_i} D_i \nabla Y_i \cdot \nabla T - \sum_{i=1}^{N_s} h_i \dot{\omega}_i, \tag{2.10e}$$

$$(\rho Y_i)_t + (uY_i)_x + (vY_i)_y + (wY_i)_z = \rho D_i \Delta Y_i + \dot{\omega}_i. \tag{2.10f}$$

Then, the Galerkin procedure is applied to the dimensionless governing equations. The purpose of the Galerkin procedure is to convert the continuous nonlinear governing equations to a discrete system. The Galerkin procedure is totally different from finite differencing. All approximations in finite-difference methods are local, extending over only a few grid points; in contrast, construction of a Galerkin procedure is based on a global functional representation McDonough [112].

The global representation of dependent variables is present in the form of Fourier series:

$$q_i(\boldsymbol{x}, t) = \sum_{\boldsymbol{k}=1}^{\infty} a_{\boldsymbol{k},i}(t)\varphi_{\boldsymbol{k}}(\boldsymbol{x}), \qquad x \in \Omega, \quad \in [t_0, t_f], \tag{2.11}$$

where functions $\{\varphi_{\boldsymbol{k}}\}_{\boldsymbol{k}=1}^{\infty}$ are basis functions of the Galerkin approximation for the Fourier coefficients, $a_{\boldsymbol{k},i}$, of the i^{th} dependent variable. There are several requirements associated with these basis functions: 1) $\{\varphi_{\boldsymbol{k}}\}_{\boldsymbol{k}=1}^{\infty}$ is complete in $L^2(\Omega)$; 2) it is orthonormal; 3) it exhibits behavior similar to complex exponentials, $e^{i\boldsymbol{k}\cdot\boldsymbol{x}}$, with respect to differentiation. In this thesis, the SGS model focuses on only one single point, and the peripheral points have no effect on it.

The Fourier series for dependent variables, velocities, species and temperature are:

$$u(x,y,z,t) = \sum_{\boldsymbol{k}}^{\infty} a_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y,z), \tag{2.12a}$$

$$v(x,y,z,t) = \sum_{\boldsymbol{k}}^{\infty} b_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y,z), \tag{2.12b}$$

$$w(x,y,z,t) = \sum_{\boldsymbol{k}}^{\infty} c_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y,z), \tag{2.12c}$$

$$Y_i(x,y,z,t) = \sum_{\boldsymbol{k}}^{\infty} d_{i\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y,z), \tag{2.12d}$$

$$T(x,y,z,t) = \sum_{\boldsymbol{k}}^{\infty} e_{\boldsymbol{k}}(t)\varphi_{\boldsymbol{k}}(x,y,z), \tag{2.12e}$$

with $\boldsymbol{k} = (k_1, k_2, k_3)^T$. The lower bound for components of this wavevector is typically one of $\{-\infty, 0, 1\}$. Also, it is natural to employ complex exponentials as basis functions in the form,

$$\varphi_{\boldsymbol{k}}(x,y,z) = e^{i\boldsymbol{k}\cdot\boldsymbol{x}} = e^{i(k_1x+k_2y+k_3z)} = e^{ik_1x}e^{ik_2y}e^{ik_3z}. \tag{2.13}$$

We substitute the Eqs. (2.12a), (2.12b), (2.12c) into the x-momentum equation (2.10b) to get

$$\frac{\partial}{\partial t}\sum_{\boldsymbol{l}} a_{\boldsymbol{l}}\varphi_{\boldsymbol{l}} + \frac{\partial}{\partial x}\sum_{\boldsymbol{l},\boldsymbol{m}} a_{\boldsymbol{l}}a_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + \frac{\partial}{\partial y}\sum_{\boldsymbol{l},\boldsymbol{m}} a_{\boldsymbol{l}}b_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + \frac{\partial}{\partial z}\sum_{\boldsymbol{l},\boldsymbol{m}} a_{\boldsymbol{l}}c_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} = \frac{1}{Re}\left[\frac{\partial^2}{\partial x^2}\sum_{\boldsymbol{l}} a_l\varphi_{\boldsymbol{l}} + \frac{\partial^2}{\partial y^2}\sum_{\boldsymbol{l}} a_{\boldsymbol{l}}\varphi_l + \frac{\partial^2}{\partial z^2}\sum_{\boldsymbol{l}} a_{\boldsymbol{l}}\varphi_{\boldsymbol{l}}\right] \tag{2.14}$$

After commuting summation and differentiation this leads to,

$$\sum_{\boldsymbol{l}} \dot{a}_{\boldsymbol{l}}\varphi_{\boldsymbol{l}} + i\sum_{\boldsymbol{l},\boldsymbol{m}}(l_1+m_1)a_{\boldsymbol{l}}a_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + i\sum_{\boldsymbol{l},\boldsymbol{m}}(l_2+m_2)a_{\boldsymbol{l}}b_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}$$

$$+ i \sum_{\boldsymbol{l},\boldsymbol{m}} (l_3 + m_3) a_{\boldsymbol{l}} c_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} = -\frac{1}{Re} \sum_{\boldsymbol{l}} (l_1^2 + l_2^2 + l_3^2) a_{\boldsymbol{l}} \varphi_{\boldsymbol{l}}. \quad (2.15)$$

We can use appropriate algebraic methods to remove the imaginary factors i. In order to process formally, we use the orthonormality of $\varphi_{\boldsymbol{k}}$ and form a inner products of each term in Eq. (2.15) to obtain,

$$\dot{a}_{\boldsymbol{k}} + \sum_{\boldsymbol{l},\boldsymbol{m}} A^{(1)}_{\boldsymbol{klm}} a_{\boldsymbol{l}} a_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} B^{(1)}_{\boldsymbol{klm}} a_{\boldsymbol{l}} b_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} C^{(1)}_{\boldsymbol{klm}} a_{\boldsymbol{l}} c_{\boldsymbol{m}} = -\frac{\eta^{(1)} |\boldsymbol{k}|^2}{Re} a_{\boldsymbol{k}},$$
$$\forall -\infty < \boldsymbol{k} < \infty. \quad (2.16)$$

The coefficient $\eta^{(1)}$ denotes a normalization constant arising from the fact that derivatives of basis functions may not possess the same normalization as the functions themselves. The analogous coefficients $\eta^{(2)}$, $\eta^{(3)}$, $\eta^{(4)}$, $\eta^{(5)}$ hold for y-, z-momentum equations, energy equation and species concentration equation, respectively. The Galerkin triple products, $A^{(1)}_{\boldsymbol{klm}}$, $B^{(1)}_{\boldsymbol{klm}}$, $C^{(1)}_{\boldsymbol{klm}}$ are defined as,

$$A^{(1)}_{\boldsymbol{klm}} \equiv (l_1 + m_1) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{x},$$

$$B^{(1)}_{\boldsymbol{klm}} \equiv (l_2 + m_2) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{y},$$

$$C^{(1)}_{\boldsymbol{klm}} \equiv (l_3 + m_3) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{z},$$

where the superscript (1) denotes the x-momentum. Similarity, superscripts (2), (3) represent the y-momentum and z-momentum.

Analogous results from the Galerkin procedure hold for y-momentum and z-momentum

equations:

$$\dot{b}_{\boldsymbol{k}} + \sum_{\boldsymbol{l},\boldsymbol{m}} A^{(2)}_{\boldsymbol{klm}} b_{\boldsymbol{l}} a_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} B^{(2)}_{\boldsymbol{klm}} b_{\boldsymbol{l}} b_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} C^{(2)}_{\boldsymbol{klm}} b_{\boldsymbol{l}} c_{\boldsymbol{m}} = -\frac{\eta^{(2)}|\boldsymbol{k}|^2}{Re} b_{\boldsymbol{k}}, \tag{2.17}$$

$$\dot{c}_{\boldsymbol{k}} + \sum_{\boldsymbol{l},\boldsymbol{m}} A^{(3)}_{\boldsymbol{klm}} c_{\boldsymbol{l}} a_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} B^{(3)}_{\boldsymbol{klm}} c_{\boldsymbol{l}} b_{\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} C^{(3)}_{\boldsymbol{klm}} c_{\boldsymbol{l}} c_{\boldsymbol{m}} = -\frac{\eta^{(3)}|\boldsymbol{k}|^2}{Re} c_{\boldsymbol{k}} - \frac{Gr}{Re^2} e_{\boldsymbol{k}}. \tag{2.18}$$

Similarly, the Galerkin procedure is applied to the energy equation, which includes the Fourier series of independent variables from Eqs. (2.12):

$$\begin{aligned}
\rho c_p \left[\frac{\partial}{\partial t} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} + \frac{\partial}{\partial x} \sum_{\boldsymbol{l},\boldsymbol{m}} a_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} + \frac{\partial}{\partial y} \sum_{\boldsymbol{l},\boldsymbol{m}} b_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} + \frac{\partial}{\partial z} \sum_{\boldsymbol{l},\boldsymbol{m}} c_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} \right] = \\
\lambda \left[\frac{\partial^2}{\partial x^2} \sum_{\boldsymbol{m}} e_m \varphi_{\boldsymbol{m}} + \frac{\partial^2}{\partial y^2} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} + \frac{\partial^2}{\partial z^2} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \right] + \\
\sum_{i=1}^{N_s} \rho c_{p_i} D_i \left[\Big(\frac{\partial}{\partial x} \sum_{\boldsymbol{j}} d_{i\boldsymbol{j}} \varphi_{\boldsymbol{j}} \Big) \Big(\frac{\partial}{\partial x} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \Big) + \Big(\frac{\partial}{\partial y} \sum_{\boldsymbol{j}} d_{i\boldsymbol{j}} \varphi_{\boldsymbol{j}} \Big) \Big(\frac{\partial}{\partial y} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \Big) + \right. \\
\left. \Big(\frac{\partial}{\partial z} \sum_{\boldsymbol{j}} d_{i\boldsymbol{j}} \varphi_{\boldsymbol{j}} \Big) \Big(\frac{\partial}{\partial z} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \Big) \right] - \sum_{i=1}^{N_s} h_i \dot{\omega}_i
\end{aligned} \tag{2.19}$$

We can rearrange the product series on the right side in a simple way,

$$\Big(\frac{\partial}{\partial x} \sum_{\boldsymbol{j}} d_{i\boldsymbol{j}} \varphi_{\boldsymbol{j}} \Big) \Big(\frac{\partial}{\partial x} \sum_{\boldsymbol{m}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \Big) = \frac{\partial}{\partial x} \sum_{\boldsymbol{j}} \sum_{\boldsymbol{m}} d_{i\boldsymbol{j}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{j}} \varphi_{\boldsymbol{m}} = \frac{\partial}{\partial x} \sum_{\boldsymbol{j},\boldsymbol{m}} d_{i\boldsymbol{j}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{j}} \varphi_{\boldsymbol{m}}.$$

With commuting summation and differentiation, and combination of like terms on right side of Eq. (2.19) we obtain

$$\begin{aligned}
\rho c_p \Bigg[\sum_{\boldsymbol{m}} \dot{e}_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} + i \sum_{\boldsymbol{l},\boldsymbol{m}} (l_1 + m_1) a_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} + i \sum_{\boldsymbol{l},\boldsymbol{m}} (l_2 + m_2) b_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} \\
+ i \sum_{\boldsymbol{l},\boldsymbol{m}} (l_3 + m_3) c_{\boldsymbol{l}} e_{\boldsymbol{m}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} \Bigg] = -\lambda \Big[\sum_{\boldsymbol{m}} (m_1^2 + m_2^2 + m_3^2) e_{\boldsymbol{m}} \varphi_{\boldsymbol{m}} \Big]
\end{aligned}$$

$$+\sum_{i=1}^{N_s}\rho c_{p_i}D_i\Bigg[i\sum_{\boldsymbol{j},\boldsymbol{m}}(j_1+m_1)d_{i\boldsymbol{j}}e_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}+i\sum_{\boldsymbol{j},\boldsymbol{m}}(j_2+m_2)d_{i\boldsymbol{j}}e_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}$$
$$+i\sum_{\boldsymbol{j},\boldsymbol{m}}(j_3+m_3)d_{i\boldsymbol{j}}e_{\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}\Bigg]-\sum_{i=1}^{N_s}h_i\dot{\omega}_i. \quad (2.20)$$

In the same way, we use the orthonormality of $\varphi_{\boldsymbol{k}}$ to process formally, and form an inner product with each basis function in Eq. (2.20) to obtain,

$$\dot{e}_{\boldsymbol{k}}+\sum_{\boldsymbol{l},\boldsymbol{m}}A^{(4)}_{\boldsymbol{klm}}a_{\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}B^{(4)}_{\boldsymbol{klm}}b_{\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}C^{(4)}_{\boldsymbol{klm}}c_{\boldsymbol{l}}e_{\boldsymbol{m}}=\Bigg[-\lambda\eta^{(4)}|\boldsymbol{k}|^2e_{\boldsymbol{k}}+$$
$$\sum_{i=1}^{N_s}\rho c_{p_i}D_i\Big[\sum_{\boldsymbol{l},\boldsymbol{m}}D^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}E^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}F^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}\Big]$$
$$-\sum_{i=1}^{N_s}h_i\dot{\omega}_i\Bigg]/\rho c_p,\quad\forall-\infty<\boldsymbol{k}<\infty. \quad (2.21)$$

We can combine the Galerkin triple products on the right side of Eq. (2.21) as

$$\sum_{\boldsymbol{l},\boldsymbol{m}}D^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}E^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}+\sum_{\boldsymbol{l},\boldsymbol{m}}F^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}=\sum_{\boldsymbol{l},\boldsymbol{m}}G^{(4)}_{\boldsymbol{klm}}d_{i\boldsymbol{l}}e_{\boldsymbol{m}}.$$

Define the Galerkin triple products, $A^{(4)}_{\boldsymbol{klm}}$, $B^{(4)}_{\boldsymbol{klm}}$, $C^{(4)}_{\boldsymbol{klm}}$, $D^{(4)}_{\boldsymbol{klm}}$, $E^{(4)}_{\boldsymbol{klm}}$, $F^{(4)}_{\boldsymbol{klm}}$ and $G^{(4)}_{\boldsymbol{klm}}$ here as,

$$A^{(4)}_{\boldsymbol{klm}}\equiv(l_1+m_1)\int_\Omega\varphi_{\boldsymbol{k}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}d\boldsymbol{x},$$

$$B^{(4)}_{\boldsymbol{klm}}\equiv(l_2+m_2)\int_\Omega\varphi_{\boldsymbol{k}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}d\boldsymbol{y},$$

$$C^{(4)}_{\boldsymbol{klm}}\equiv(l_3+m_3)\int_\Omega\varphi_{\boldsymbol{k}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}d\boldsymbol{z},$$

$$D^{(4)}_{\boldsymbol{klm}} \equiv (l_4 + m_4) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{x},$$

$$E^{(4)}_{\boldsymbol{klm}} \equiv (l_5 + l_5) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{y},$$

$$F^{(4)}_{\boldsymbol{klm}} \equiv (l_6 + m_6) \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{z},$$

$$G^{(4)}_{\boldsymbol{klm}} = D^{(4)}_{\boldsymbol{klm}} + E^{(4)}_{\boldsymbol{klm}} + F^{(4)}_{\boldsymbol{klm}},$$

where the superscript (4) indicates these parameters belong to the energy equation.

We now deal with the species concentration equation Eq. (2.10f) again, with Fourier series of dependent variables given in Eqs. (2.12) resulting in

$$\rho\Big[\frac{\partial}{\partial t}\sum_{\boldsymbol{m}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{m}}\Big] + \frac{\partial}{\partial x}\sum_{\boldsymbol{l},\boldsymbol{m}} a_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + \frac{\partial}{\partial y}\sum_{\boldsymbol{l},\boldsymbol{m}} b_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + \frac{\partial}{\partial z}\sum_{\boldsymbol{l},\boldsymbol{m}} c_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} =$$
$$\rho D_i\left[\frac{\partial^2}{\partial x^2}\sum_{\boldsymbol{m}} d_{im}\varphi_{\boldsymbol{m}} + \frac{\partial^2}{\partial y^2}\sum_{\boldsymbol{m}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{m}} + \frac{\partial^2}{\partial z^2}\sum_{\boldsymbol{m}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{m}}\right] + \dot{\omega}_i. \quad (2.22)$$

After commuting summation and differentiation leads to,

$$\rho\Big[\sum_{\boldsymbol{m}} \dot{d}_{i\boldsymbol{m}}\varphi_{\boldsymbol{m}}\Big] + i\sum_{\boldsymbol{l},\boldsymbol{m}}(l_1 + m_1)a_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} + i\sum_{\boldsymbol{l},\boldsymbol{m}}(l_2 + m_2)b_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}}$$
$$+ i\sum_{\boldsymbol{l},\boldsymbol{m}}(l_3 + m_3)c_{\boldsymbol{l}} d_{i\boldsymbol{m}}\varphi_{\boldsymbol{l}}\varphi_{\boldsymbol{m}} = -\rho D_i\Big[\sum_{\boldsymbol{m}}(m_1^2 + m_2^2 + m_3^2)d_{i\boldsymbol{m}}\varphi_{\boldsymbol{m}}\Big] + \dot{\omega}_i. \quad (2.23)$$

We use the orthonormality of $\varphi_{\boldsymbol{k}}$ to process formally, and form an inner product with each basis function in Eq. (2.23) to obtain

$$\dot{d}_{i\boldsymbol{k}} + \sum_{\boldsymbol{l},\boldsymbol{m}} A^{(5)}_{\boldsymbol{klm}} a_{\boldsymbol{l}} d_{i\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} B^{(5)}_{\boldsymbol{klm}} b_{\boldsymbol{l}} d_{i\boldsymbol{m}} + \sum_{\boldsymbol{l},\boldsymbol{m}} C^{(5)}_{\boldsymbol{klm}} c_{\boldsymbol{l}} d_{i\boldsymbol{m}} = -\eta^{(5)} |\boldsymbol{k}|^2 D_i d_{i\boldsymbol{k}} + \frac{\dot{\omega}_i}{\rho},$$

$$\forall \quad -\infty < \boldsymbol{k} < \infty, \quad (2.24)$$

where i denotes the number of species concentration, and the Galerkin triple products, $A^{(5)}_{\boldsymbol{klm}}$, $B^{(5)}_{\boldsymbol{klm}}$, $C^{(5)}_{\boldsymbol{klm}}$ are defined as,

$$A^{(5)}_{\boldsymbol{klm}} \equiv \frac{(l_1 + m_1)}{\rho} \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{x},$$

$$B^{(5)}_{\boldsymbol{klm}} \equiv \frac{(l_2 + m_2)}{\rho} \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{y},$$

$$C^{(5)}_{\boldsymbol{klm}} \equiv \frac{(l_3 + m_3)}{\rho} \int_\Omega \varphi_{\boldsymbol{k}} \varphi_{\boldsymbol{l}} \varphi_{\boldsymbol{m}} d\boldsymbol{z},$$

where the superscript (5) denotes triple products used for species concentration equations.

2.2.2 Euler integration and discrete dynamical systems

We will apply a simple forward Euler single-step, explicit time integration procedure to numerically solve this dynamical system. In the momentum equations, energy equation and species concentration equation, the initial condition (a_0, b_0, c_0, e_0, $d_{i,0}$) should be taken into consideration. Then we obtain

$$a^{n+1} = a^n - \tau \left[\frac{\eta^{(1)} |\boldsymbol{k}|^2}{Re} a^n + A^{(1)} (a^n)^2 + B^{(1)} a^n b^n + C^{(1)} a^n c^n \right], \quad (2.25a)$$

$$b^{n+1} = b^n - \tau \left[\frac{\eta^{(2)} |\boldsymbol{k}|^2}{Re} b^n + A^{(2)} a^n b^n + B^{(2)} (b^n)^2 + C^{(2)} b^n c^n \right], \quad (2.25b)$$

$$c^{n+1} = c^n - \tau\left[\frac{\eta^{(3)}|\boldsymbol{k}|^2}{Re}c^n + \frac{Gr}{Re^2}e^n + A^{(3)}a^n c^n + B^{(3)}b^n c^n + C^{(3)}(c^n)^2\right], \tag{2.25c}$$

$$d_i{}^{n+1} = d_i{}^n - \tau\left[A^{(5)}a^{n+1}d_i{}^n + B^{(5)}b^{n+1}d_i{}^n + C^{(5)}c^{n+1}d_i{}^n + \eta^{(5)}|\boldsymbol{k}|^2 D_i d_i{}^n - \frac{\dot{\omega}_i}{\rho}\right] + d_{i,0}. \tag{2.25d}$$

$$e^{n+1} = e^n + \tau\Bigg[- \lambda\eta^{(4)}|\boldsymbol{k}|^2 e^n + \sum_{i=1}^{N_s}\rho c_{p_i} D_i G^{(4)} d_i^{n+1} e^n - A^{(4)}a^{n+1}e^n - B^{(4)}b^{n+1}e^n - C^{(4)}c^{n+1}e^n - \sum_{i=1}^{N_s} h_i\dot{\omega}_i\Bigg]/\rho c_p + e_0, \tag{2.25e}$$

where τ denotes an arbitrary discrete time step parameter. Rearrange the x-momentum equation leads to

$$a^{n+1} = \tau A^{(1)}a^n\left(\frac{1-\eta^{(1)}\tau|\boldsymbol{k}|^2/Re}{\tau A^{(1)}} - a^n\right) - \tau B^{(1)}a^n b^n - \tau C^{(1)}a^n c^n, \tag{2.26}$$

and to include the logistical map Eq. (1.9), we need to require

$$\frac{1-\eta^{(1)}\tau|\boldsymbol{k}|^2/Re}{\tau A^{(1)}} = 1, \tag{2.27}$$

which means

$$\tau A^{(1)} = 1 - \frac{\eta^{(1)}\tau|\boldsymbol{k}|^2}{Re}. \tag{2.28}$$

Substituting Eqs. (2.27) and (2.28) into Eq. (2.26) results in the form of the x-momentum equation used in this study. In the same way, we can obtain equations in the other two directions. The discrete dynamical system momentum equations are,

$$a^{(n+1)} = \left(1 - \frac{\eta^{(1)}\tau|\boldsymbol{k}|^2}{Re}\right)a^{(n)}\left(1-a^{(n)}\right) - \tau B^{(1)}a^{(n)}b^{(n)} - \tau C^{(1)}a^{(n)}c^{(n)}, \tag{2.29a}$$

$$b^{(n+1)} = \left(1 - \frac{\eta^{(1)}\tau|\boldsymbol{k}|^2}{Re}\right)b^{(n)}\left(1-b^{(n)}\right) - \tau B^{(2)}a^{(n)}b^{(n)} - \tau C^{(2)}b^{(n)}c^{(n)}, \tag{2.29b}$$

$$c^{(n+1)} = \left(1 - \frac{\eta^{(1)}\tau|\boldsymbol{k}|^2}{Re}\right)c^{(n)}\left(1 - c^{(n)}\right) - \tau B^{(3)}a^{(n)}c^{(n)} - \tau C^{(3)}b^{(n)}c^{(n)} - \tau\frac{Gr}{Re^2}e_{\boldsymbol{k}}. \tag{2.29c}$$

Then, we define the bifurcation parameters related to Reynolds number as follows:

$$\begin{aligned}
\beta_u &= 1 - \frac{\eta^{(1)}\tau|\boldsymbol{k}|^2}{Re}, \\
\beta_v &= 1 - \frac{\eta^{(2)}\tau|\boldsymbol{k}|^2}{Re}, \\
\beta_w &= 1 - \frac{\eta^{(3)}\tau|\boldsymbol{k}|^2}{Re}.
\end{aligned}$$

The range of β values usually belongs to (0, 4) due to the logistic map, see Bible [113].

Let α_T and γ_{ij} as,

$$\begin{aligned}
\alpha_T &= -\tau\frac{Gr}{Re^2}e_{\boldsymbol{k}}, \\
\gamma_{12} = \tau B^{(1)}, &\quad \gamma_{13} = \tau C^{(1)}, \\
\gamma_{21} = \tau B^{(2)}, &\quad \gamma_{23} = \tau C^{(2)}, \\
\gamma_{31} = \tau B^{(3)}, &\quad \gamma_{32} = \tau C^{(3)}.
\end{aligned}$$

Rearrange Eq. (2.25e) as

$$\begin{aligned}
e^{(n+1)} = \Bigg[\Big(\rho c_p - \tau\lambda\eta^{(4)}|\boldsymbol{k}|^2 + \sum_{i=1}^{N_s}\tau\rho c_{p_i}D_iG^{(4)}d_i^{(n+1)}\Big)e^n - \tau A^{(4)}a^{(n+1)}e^{(n)} \\
- \tau B^{(4)}b^{(n+1)}e^{(n)} - \tau C^{(4)}c^{(n+1)}e^{(n)} - \sum_{i=1}^{N_s}\tau h_i\dot{\omega}_i\Bigg]/\rho c_p + e_0, \quad (2.30)
\end{aligned}$$

and let $\rho c_p = \tau\lambda\eta^{(4)}|\boldsymbol{k}|^2$, $H_i = \tau h_i$, and $\alpha T d_i = \tau\rho c_{p_i}D_iG^{(4)}$; then define the remaining bifurcation parameters in Eq. (2.30) as

$$\gamma_{uT} = \ \tau A^{(4)}$$

$$\gamma_{vT} = \tau B^{(4)},$$
$$\gamma_{wT} = \tau C^{(4)},$$
$$\beta_T = \rho c_p - 1.$$

The equation of heat capacity ratio is $\gamma = C_P/C_V$, and $C_P - C_V = VT\alpha^2/\beta_T$, where α is the coefficient of thermal expansion, and β_T is the isothermal compressibility. Rearrangement of Eq. (2.25d) leads to,

$$d_i{}^{n+1} = -\Big[(\tau\eta^{(5)}|\boldsymbol{k}|^2 D_i - 1) + \tau A^{(5)} a^{n+1} + \tau B^{(5)} b^{n+1} + \tau C^{(5)} c^{n+1}\Big] d_i{}^n + \tau\frac{\dot{\omega}_i}{\rho} + d_{i,0}, \quad (2.31)$$

and define the bifurcation parameters for Eq. (2.31) as follows and in order to simplify the parameters. Let

$$\beta_{Y_i} = \tau\eta^{(5)}|\boldsymbol{k}|^2 D_i - 1,$$

and define

$$\gamma_{uY_i} = \tau A^{(5)},$$
$$\gamma_{vY_i} = \tau B^{(5)},$$
$$\gamma_{wY_i} = \tau C^{(5)}.$$

Finally, for convenience of notation, use $\dot{\omega}_i$ to replace the original $\tau\dot{\omega}_i/\rho$.

The complete DDS takes the form,

$$a^{(n+1)} = \beta_u a^{(n)}(1 - a^{(n)}) - \gamma_{uv} a^{(n)} b^{(n)} - \gamma_{uw} a^{(n)} c^{(n)}, \quad (2.32a)$$
$$b^{(n+1)} = \beta_v b^{(n)}(1 - b^{(n)}) - \gamma_{vu} a^{(n)} b^{(n)} - \gamma_{vw} b^{(n)} c^{(n)}, \quad (2.32b)$$
$$c^{(n+1)} = \beta_w c^{(n)}(1 - c^{(n)}) - \gamma_{wu} a^{(n)} c^{(n)} - \gamma_{wu} b^{(n)} c^{(n)} + \alpha_T e^{(n)}, \quad (2.32c)$$

$$d_i^{(n+1)} = -\left(\beta_{Y_i} + \gamma_{uY_i} a^{(n+1)} + \gamma_{vY_i} b^{(n+1)} + \gamma_{wY_i} c^{(n+1)}\right) d_i^{(n)} + \dot{\omega}_i + d_{i,0} \qquad i = 1, 2, \ldots, N_s, \tag{2.32d}$$

$$e^{(n+1)} = \left[\left(\sum_{i=1}^{N_s} \alpha_{Td_i} d_i^{(n+1)} - \gamma_{uT} a^{(n+1)} - \gamma_{vT} b^{(n+1)} - \gamma_{wT} c^{(n+1)} \right) e^{(n)} - \sum_{i=1}^{N_s} H_i \, \dot{\omega}_i \right] / (1 + \beta_T) + e_0 \,, \tag{2.32e}$$

with

$$\dot{\omega}_i = \sum_{j=1}^{N_r} \left[C_{f,ij} \prod_{l=1}^{N_s} d_l^{\nu'_{j,l}} - C_{b,ij} \prod_{l=1}^{N_s} d_l^{\nu''_{j,l}} \right]. \tag{2.33}$$

Here, superscripts (n) are time-step indices; a b, c, d_is, and e denote Fourier coefficients of the velocity vector in three directions, species concentrations, and temperature, respectively; the subscripted α_s, β_s, γ_s are DDS bifurcation parameters, all of which are associated with the various physical bifurcation parameters. For example, β_u, β_v, and β_w are functions of Reynolds number; α_T is related to Grashof number; the α_{Td_i} are related to Schmidt and Lewis numbers; and $H_i s$ are associated with specific enthalpies for each species i; the $C_{f,ij}$, $C_{b,ij}$ can be related to Kolmogorov-scale Damköhler numbers. The various γs correspond to velocity, temperature and species concentration gradients. The $d_{i,0}$s and e_0 are high-pass filtered species concentrations and temperature, respectively, for subgrid-scale behavior as described in [103]. We mention that for simplicity, we will set α_T identically equal to zero in the present work, as buoyancy effects are negligible.

2.2.3 Homogeneous and isotropic assumptions

For simplicity, we employ a homogeneous and isotropic assumption for bifurcation parameters as an initial study, such as,

$$\beta_u = \beta_v = \beta_w = \beta,$$

$$\gamma_{uv} = \gamma_{uw} = \gamma_{vu} = \gamma,$$

$$\gamma_{vw} = \gamma_{wu} = \gamma_{wv} = \gamma,$$

$$\gamma_{uT} = \gamma_{vT} = \gamma_{wT} = \gamma_T,$$

$$\gamma_{uY_i} = \gamma_{vY_i} = \gamma_{uwY_i} = \gamma_{Y_i}.$$

Where, β_u,β_v and β_w are bifurcation parameters related to Reynolds number in three different dimension, individually; γ_{uv}, γ_{uw}, γ_{vu}, γ_{vw}, γ_{wu}, and γ_{wv} are bifurcation parameters associated to velocities; γ_{uT}, γ_{vT}, and γ_{wT} are bifurcation parameters related to temperature; and γ_{uY_i}, γ_{vY_i}, and γ_{wY_i} are bifurcation parameters associated to species i. We remark that neither the fluid flow nor the chemistry can be expected to be either homogenous or isotropic in a combusting flow, but this assumption provides a tractable starting point by greatly decreasing the number of different bifurcation parameter values needed. Moreover, we emphasize that no such assumption is needed in a complete LES because all bifurcation parameters can be calculated from high-pass filtered resolved-scale results. The values of these parameters associated with species i, e.g., β_{Y_i}, α_{Td_i} etc., should be analyzed individually since every species has its own characteristics. With this simplification, we obtain the DDS with homogeneous and isotropic assumption,

$$a^{(n+1)} = \beta a^{(n)}(1 - a^{(n)}) - \gamma a^{(n)} b^{(n)} - \gamma a^{(n)} c^{(n)}, \tag{2.34a}$$

$$b^{(n+1)} = \beta b^{(n)}(1 - b^{(n)}) - \gamma a^{(n)} b^{(n)} - \gamma b^{(n)} c^{(n)}, \tag{2.34b}$$

$$c^{(n+1)} = \beta c^{(n)}(1 - c^{(n)}) - \gamma a^{(n)} c^{(n)} - \gamma b^{(n)} c^{(n)} + \alpha_T e^{(n)}, \tag{2.34c}$$

$$d_i^{(n+1)} = -(\beta_{Y_i} + \gamma_{Y_i} a^{(n+1)} + \gamma_{Y_i} b^{(n+1)} + \gamma_{Y_i} c^{(n+1)}) d_i^{(n)} + \dot{\omega}_i + d_{i,0} \qquad i = 1, 2, \ldots, N_s, \tag{2.34d}$$

$$e^{(n+1)} = \left[\left(\sum_{i=1}^{N_s} \alpha_{Td_i} d_i^{(n+1)} - \gamma_T a^{(n+1)} - \gamma_T b^{(n+1)} - \gamma_T c^{(n+1)} \right) e^{(n)} - \sum_{i=1}^{N_s} H_i \, \dot{\omega}_i \right] / (1 + \beta_T) + e_0 \,, \tag{2.34e}$$

2.2.4 Inhomogeneity and anisotropy

After computing and analyzing results with the original homogeneous and isotropic assumption, we then deal with more general prescription of these bifurcation parameters. That is, inhomogeneity and anisotropy conditions are employed for the DDS model, and results are compared with the original solutions. Both sets of calculations will be compared with experimental data to assess effects of the simplifying the homogeneous and isotropic assumption.

2.3 Finite-Rate Chemistry Reduced Mechanism

Overall (global) reactions are a consequence of collections of elementary reactions, and resolution of these elementary reactions is a difficult and time-consuming task, as noted by Warnatz *et al.* [23]. Many elementary reactions produce a negligible contribution to the reaction process, and therefore can be ignored, leading to reduced mechanisms.

Here, we first study the case of H_2-air reactions with N_2 dilution mechanism, and then investigate the case of H_2-air reactions with reacting N_2 mechanism. The skeletal mechanism of H_2-O_2 for this work is listed below in Eqs. (2.35) which is one part of detailed H_2-O_2 reaction mechanism of Li *et al.* [114], and they are also investigated in a reduced mechanism for H_2-air combustion by Boivin *et al.* [32]. The corresponding reaction rate data (assuming Arrhenius form) are listed in Table 2.1, wherein f means forward reaction, and b denotes backward reaction. The coefficient k_0 is a low-pressure rate coefficient, and k_∞ is a high pressure rate coefficient.

Table 2.1: Rate Coefficients for Reduced Mechanism of H_2-Air Reaction

Number	F	A[a]	n	E[a]	B	A[a]	n	E[a]
a	k_f	3.52×10^{16}	-0.7	71.42	k_b	7.04×10^{13}	-0.26	0.60
b	k_f	5.06×10^{4}	2.67	26.32	k_b	3.03×10^{4}	2.63	20.23
c	k_f	1.17×10^{9}	1.3	15.21	k_b	1.28×10^{10}	1.19	78.25
d	k_0	5.75×10^{19}	-1.4	0.0	k_∞	4.65×10^{12}	0.44	0.0
e		7.08×10^{13}	0.0	1.23				
f	k_f	1.66×10^{13}	0.0	3.44	k_b	2.69×10^{12}	0.36	231.86
g		2.89×10^{13}	0.0	-2.08				
h	k_f	4.00×10^{22}	-2.0	0.0	k_b	1.03×10^{23}	-1.75	496.14
i	k_f	1.30×10^{18}	-1.0	0.0	k_b	3.04×10^{17}	-0.65	433.09

$$H + O_2 \rightleftharpoons OH + O \tag{2.35a}$$

$$H_2 + O \rightleftharpoons OH + H \tag{2.35b}$$

$$H_2 + OH \rightleftharpoons H_2O + H \tag{2.35c}$$

$$H + O_2 + M \rightharpoonup HO_2 + M \tag{2.35d}$$

$$HO_2 + H \rightharpoonup 2OH \tag{2.35e}$$

$$HO_2 + H \rightleftharpoons H_2 + O_2 \tag{2.35f}$$

$$HO_2 + OH \rightharpoonup H_2O + O_2 \tag{2.35g}$$

$$H + OH + M \rightleftharpoons H_2O + M \tag{2.35h}$$

$$2H + M \rightleftharpoons H_2 + M \tag{2.35i}$$

This reduced mechanism has been shown to be sufficient to describe premixed and nonpremixed flames, autoignition, and detonations under conditions of practical interest. It consists of 15 reversible elementary reactions in the reduced mechanism

that is described by Boivin *et al.* [32], and we collapse them to a nine-step mechanism, involving eight reacting species H_2, O_2, H_2O, OH, H, O, HO_2, N_2. M is the third body which involves reaction of two species, such as A and B to yield one single product species AB, where M is used to add energy to species and stabilize the excited products species AB* by collision. The third body M is usually regard as any species that can remove the heat from the excited products and finally dissipate it to heat, see Warnatz *et al.* [23]. In this thesis, the third body M includes all reacting species except the reactants/products in a particular reaction. For example, in reaction 2.35d, M includes species H_2, H_2O, OH, O, HO_2, N_2,

$$M = (a)[H_2] + (b)[H_2O] + (c)[OH] + (d)[O] + (e)[HO_2] + (f)[N_2]. \tag{2.36}$$

Where, the chaperon efficiency is a multiplier on the corresponding specific species. The efficiencies are equal to 1 for all of reacting species in M, which is different from the so-called San Diego mechanism, see Saxena and Williams [24]. Detailed numerical process of species reaction in the DDS model is displayed in Section 2.4 and Appendix A.

2.3.1 Investigation on N_2 Chemistry

It is known that understanding of pollutant formation mechanisms is important to protecting the environment, and in the last decade many researchers made efforts toward understanding this mechanism, especially with respect to NO_x and soot formation, as mentioned by Carbonell *et al.* [34]. In this section, the effects of species N_2 will be analyzed in two aspects. One is N_2 acting as only one part of third body in the N_2 dilution mechanism. Another is that N_2 involved in the chemical reaction at the reacting N_2 mechanism. In the second mechanism, it is necessary to consider NO_x effects, especially when temperature exceeds 2000 K, since NO_x reaction is sensitive

to high temperature.

Generally, formation of NO_x includes three primary types: thermal NO_x, fuel NO_x, and prompt NO_x. Beychok [115] made an effort to investigate different formation mechanisms of NO_x. Thermal NO_x is formed through high temperature oxidation of the diatomic nitrogen. This formation rate is determined by temperature and the residence time of nitrogen at that temperature. Fuel NO_x is produced from nitrogen-bearing fuels with excess oxygen in the air, and it is the major emission in combustion of oil and coal. Fuel NO_x can make up as much as of 50% of total emission in oil combustion processes, and as much as 80% in coal combustion processes. Prompt NO_x occur in the earliest stage of combustion; it results from the reaction of atmospheric nitrogen (N_2) with radicals (viz., C, CH, and CH_2) in the air. The levels of prompt NO_x are usually very low, and it is only of interest in the precise emission investigations. Pre-combustion and post-combustion technology are two primary methodologies in reducing NO_x for industrial combustors: pre-combustion prevents NO_x from forming, and it can be accomplished by either using flue gases recirculation (FGR) technology in the combustion process or staging the combustion process; post-combustion allows NO_x to form, then breaks it down in the exhaust gases (for details see, the formation of NO_x URL: http://www.alentecinc.com/papers/NOx/).

We will check the effects of NO_x, and modify the DDS model appropriately. The NO_x reaction, the so called "thermal NO_x mechanism," is associated with high-temperature oxidation of the diatomic nitrogen, as already noted. The three principal reactions (the extended Zeldovich mechanism, see Dixon-Lewis *et al.* [14] and Janicka and Kollmann [116]) producing thermal NO_x are given in [23]:

$$O + N_2 \rightleftharpoons NO + N, \tag{2.37a}$$

$$N + O_2 \rightleftharpoons NO + O, \tag{2.37b}$$

$$N + OH \rightleftharpoons NO + H. \tag{2.37c}$$

The corresponding reaction coefficients are listed in Table 2.2, wherein f means forward reaction, and b denotes backward reaction.

Table 2.2: Rate Coefficients for Thermal NO_x Mechanism

Number	F	A^a	n	E^a	B	A^a	n	E^a
a	k_f	1.8×10^8	0.0	319.00	k_b	3.8×10^7	0.0	3.53
b	k_f	1.8×10^4	1.0	38.90	k_b	3.8×10^3	1.0	173.10
c	k_f	7.1×10^7	0.0	3.74	k_b	1.7×10^8	0.0	204.19

2.4 Discrete Dynamical System Model for Specific Species Reaction

In the section, a specific application of species reaction is carried out in the DDS model in reacting N_2 mechanism. In order to construct the DDS corresponding to the reduced mechanism, an iterated map for each product appearing in every elementary reaction is derived, as did by McDonough in [102]. Each iterated map is one of the specific form of Eq. 2.32d, and the formula is shown in Eq. 2.33. The species notations in the Fortran 77 code are,

$$d_1 \sim H_2, d_2 \sim O_2, d_3 \sim H_2O, d_4 \sim OH, d_5 \sim H,$$
$$d_6 \sim O, d_7 \sim HO_2, d_8 \sim N_2, d_9 \sim N, d_{10} \sim NO.$$

With 10 species are involved in the N_2 reaction mechanism, and the analysis of atomic hydrogen reaction processes in the DDS model is based on this mechanism. The corresponding DDS for atomic hydrogen in reaction is

$$d_5^{(n+1)} = -(\beta_{Y_5} + \gamma_{uY_5} a^{(n+1)} + \gamma_{vY_5} b^{(n+1)} + \gamma_{wY_5} c^{(n+1)}) d_5^{(n)} + \dot{\omega}_5 + d_{5,0} \tag{2.38}$$

with

$$\dot{\omega}_5 = \sum_{j=1}^{21} \left[C_{f,(5,j)} \prod_{l=1}^{10} d_l^{\nu'_{j,l}} - C_{b,(5,j)} \prod_{l=1}^{10} d_l^{\nu''_{j,l}} \right]. \tag{2.39}$$

According to the N_2 reaction mechanism, the formation rate of species H, $\dot{\omega}_5$, is calculated as follow,

$$\begin{aligned}\dot{\omega}_5 = & k_{f,1}\frac{C_{f(5,1)}}{W_2W_5}d_2d_5 + k_{b,1}\frac{C_{b(5,1)}}{W_4W_6}d_4d_6 + k_{f,2}\frac{C_{f(5,2)}}{W_1W_6}d_1d_6 + k_{b,2}\frac{C_{b(5,2)}}{W_4W_5}d_4d_5 \\ & + k_{f,3}\frac{C_{f(5,3)}}{W_1W_4}d_1d_4 + k_{b,3}\frac{C_{b(5,3)}}{W_3W_5}d_3d_5 + k_0\frac{C_{f(5,4)}}{W_2W_5}M(4)d_2d_5 + k_5\frac{C_{f(5,5)}}{W_7W_5}d_7d_5 \\ & + k_{f,6}\frac{C_{f(5,6)}}{W_7W_5}d_7d_5 + k_{b,6}\frac{C_{b(5,6)}}{W_1W_2}d_1d_2 + k_{f,8}\frac{C_{f(5,8)}}{W_5W_4}M(8)d_5d_4 + k_{b,8}\frac{C_{b(5,8)}}{W_3}M(8)d_3 \\ & + k_{f,9}\frac{C_{f(5,9)}}{W_5W_5}M(9)d_5^2 + k_{b,9}\frac{C_{b(5,9)}}{W_1}M(9)d_1 + k_{f,12}\frac{C_{f(5,12)}}{W_9W_4}d_9d_4 + k_{b,12}\frac{C_{b(5,12)}}{W_{10}W_5}d_{10}d_5.\end{aligned} \tag{2.40}$$

Where,

$$\begin{aligned} C_{f(5,1)} &= -W_5, & C_{b(5,1)} &= W_5, & C_{f(5,2)} &= W_5, & C_{b(5,2)} &= -W_5, \\ C_{f(5,3)} &= W_5, & C_{b(5,3)} &= -W_5, & C_{f(5,4)} &= -W_5, & C_{b(5,5)} &= -W_5, \\ C_{f(5,6)} &= W_5, & C_{b(5,6)} &= W_5, & C_{f(5,8)} &= -W_5, & C_{b(5,8)} &= W_5, \\ C_{f(5,9)} &= -2W_5, & C_{b(5,9)} &= 2W_5, & C_{f(5,12)} &= W_5, & C_{b(5,12)} &= -W_5, \end{aligned}$$

and the third partner M are

$$\begin{aligned} M(4) &= \frac{d_1}{W_1} + \frac{d_3}{W_3} + \frac{d_4}{W_4} + \frac{d_6}{W_6} + \frac{d_8}{W_8} + \frac{d_9}{W_9} + \frac{d_{10}}{W_{10}} \\ M(8) &= \frac{d_1}{W_1} + \frac{d_2}{W_2} + \frac{d_6}{W_6} + \frac{d_7}{W_7} + \frac{d_8}{W_8} + \frac{d_9}{W_9} + \frac{d_{10}}{W_{10}} \\ M(9) &= \frac{d_2}{W_2} + \frac{d_3}{W_3} + \frac{d_4}{W_4} + \frac{d_6}{W_6} + \frac{d_7}{W_7} + \frac{d_8}{W_8} + \frac{d_9}{W_9} + \frac{d_{10}}{W_{10}}. \end{aligned}$$

The collision partner (third body) M(4), M(8), and M(9) represent the third body in Eq. 2.35d, Eq. 2.35h and Eq. 2.35e, individually. The third body M is different in different reactions, and in this work it is treated as

$$M = [H_2] + [H_2O] + [OH] + [O] + [N_2] + [N] + [NO],$$

$$M = [H_2] + [H_2O] + [O] + [HO_2] + [N_2] + [N] + [NO],$$
$$M = [O_2] + [H_2O] + [OH] + [O] + [HO_2] + [N_2] + [N] + [NO],$$

where the chaperon efficiencies are set as unity rather than different multipliers.

For M(4) in Eq. 2.35d, there are two different types of rate coefficients: k_0 is a low-pressure rate coefficient, where the concentration of third body M is very small in the low pressure range; k_∞ is a high pressure rate coefficient, where the concentration of collision partner M has a large concentration in the high pressure range. This obvious pressure dependence of rate coefficients reactions is a sequence of reactions, and the simplest case for the pressure dependence reaction is *Lindemann model* (1922) (for details, see Warnatz *et al.* [23]).

In this thesis, in order to consider the effect of concentration of collision parter M, the low pressure rate coefficient, k_0, is applied. In the future study, the high pressure rate coefficient, k_∞, will be investigated, where the reaction rate is independent on the concentrations of the collision partners.

The other species reaction application in the DDS model is similar to this atomic hydrogen case, and the corresponding numerical simulation is performed in Fortran code (for details, see Appendix A).

2.5 Temperature Model

A scaling method is applied in the DDS model for all of independent variables, except for temperature. It is known that chemical reactions are sensitive to temperature variations; thus a physical temperature must be calculated in the DDS model. Temperature is updated by every time step, and the equation for the temperature model in the computing code is

$$\boldsymbol{T}(\boldsymbol{x}, t_{n+1}) = \boldsymbol{T}(\boldsymbol{x}, t_n) + F * \boldsymbol{T}(\boldsymbol{x}, t_n) * e_{n+1} * sgn. \tag{2.41}$$

Here $\boldsymbol{T}(t_{n+1})$ is temperature at the (n + 1) time step, and $\boldsymbol{T}(t_n)$ is temperature at the n time step; The symbol F is a constant factor which is related to the ratio of large scale temperature to small temperature fluctuations; and e_{n+1} is the scaling or small-part of temperature at the (n + 1) time step, which is calculated directly from Eq. 2.34e. The scaling initial value of temperature, e_0, is set as 0.035 in the Fortran code, which means the small part of temperature is regarded as 3.5% percent of total amount in the initial calculation. The notation *sgn* is a simple sign: it is equal to 1 when $e_{n+1} > e_n$; or it is equal to -1 when $e_{n+1} < e_n$.

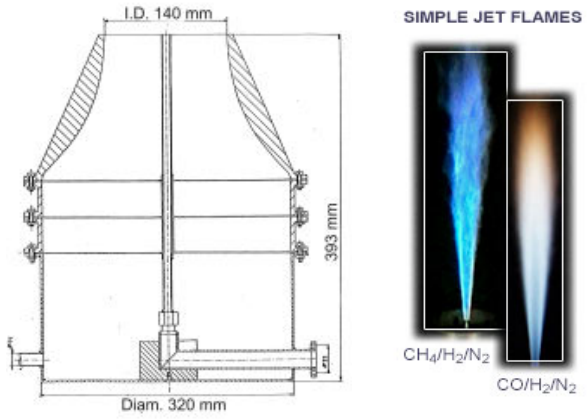
It is hard to obtain an appropriate value for F from a theoretical analysis, and the optimized value must be found by performing numerical testing. The process of this testing is the same as function of regime maps. That is, setting up all of parameters in the DDS model but F, then keep changing this value until the best numerical result is reached. The temporal temperature fluctuation and time averaged temperature in the testing solution are compared with experimental data, and the value of F is determined when the comparison exhibited minimum difference. The testing range of F is from 1 to 4, and the optimized factor is 2.8 in the final Fortran code. Detailed calculation process of temperature model is shown in Appendix A.

Along with the temperature model, a scaling method is applied for momentum to obtain small scales of motion. In the numerical code, a scaling ratio 0.025 is used which means 2.5% percent of the momentum is dissipated from large scale to small scale. However, the scaling ratio is a preset number in this work, and we should put more efforts into investigating this ratio to get a higher accuracy for variable variation prediction in the DDS model.

Chapter 3 Experimental Data

The experimental data are chosen from DLR Institute of Combustion Technology Experimental Data Archives, H_2/N_2 Jet Diffusion Flame [H3] (DLR Stuttgart), which is available at http://www.sandia.gov/TNF/simplejet.html. This flame was selected as a "standard flame" of the *International Workshop on Measurements and Computation of Turbulent Nonpremixed Flames,* Naples, July 1996. It was investigated at the TU Darmstadt, Fachgebiet Energie-und Kraftwerkstechnik, and those data sets are available in the TU Darmstadt–Flame Data Base, as provided by Schneider *et al.* [106]. This is a non-premixed flame with fuel (50% H_2 + 50% N_2, Reynolds number = 10000, nozzle diameter = 8mm, $V_{exit} = 34.8$ m/s) and co-flowing air ($V_{air} = 0.3$ m/s). Meier *et al.* [3] provide a detailed description of the burner geometry and the corresponding diffusion flame structure for various fuels. The burner geometry and diffusion flame structures are shown in Fig. 3.1(a) and Fig 3.1(b), saperately.

Figure 3.2 shows temperature distribution in the global flame field, and there are three different types of temperature lines in this chart. The flame center is plotted along the center of the flame, where highest fuel concentrations are located; the maximum temperature line is made up of locations where the maximum temperature occurs at a specified height in the flame field; the flame boundary line means fuel concentrations are very low (less than 0.1%) at these locations.



(a) Burner (b) Flame Structure

Figure 3.1: Burner Structure and Diffusion Flames Structures

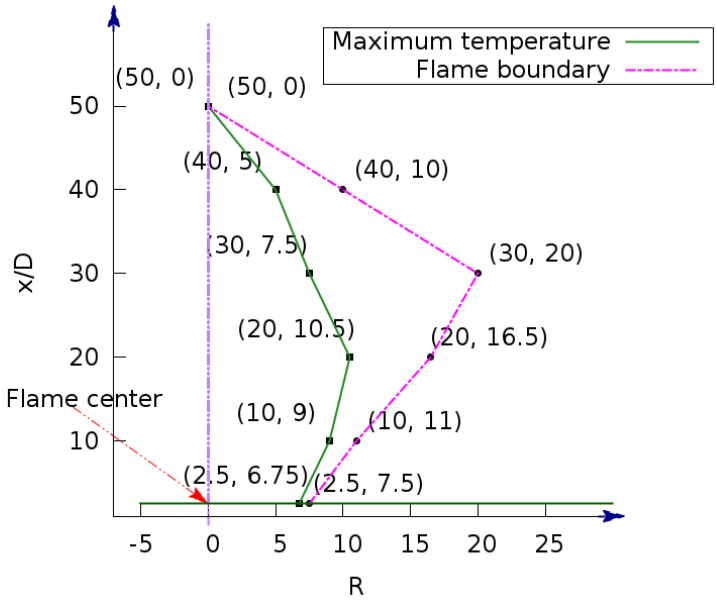
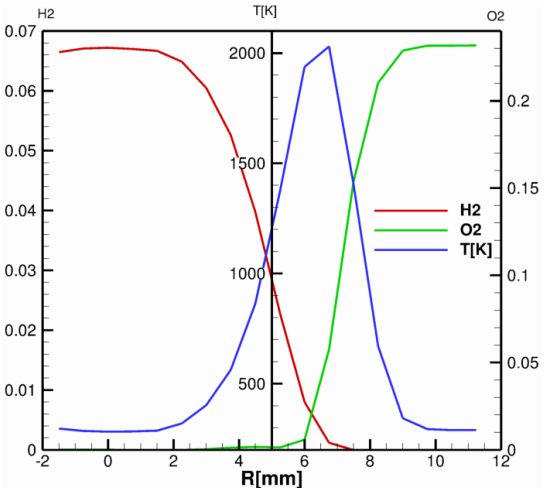


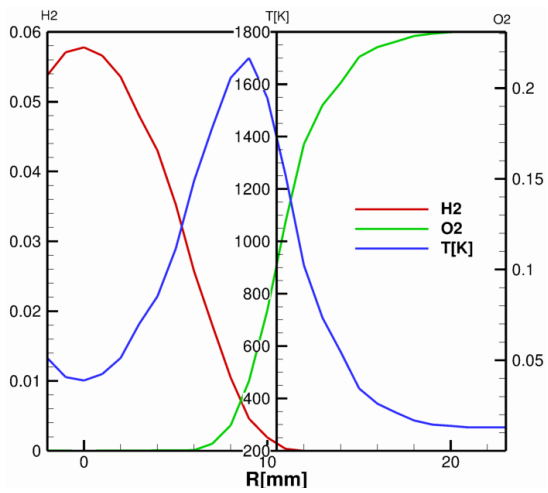
Figure 3.2: Global flame temperature distribution

The variation of temperature and concentration of two major species, H_2 and O_2 are presented at seven vertical locations: these are $x/D = 2.5$, $x/D = 10$, $x/D = 20$, $x/D = 30$, $x/D = 40$, $x/D = 50$ and $x/D = 70$. These figures aid the choice of five specific positions for further investigation. In the following figures, x is the distance from the nozzle along the flame axis, D is the nozzle diameter, and R is the radial

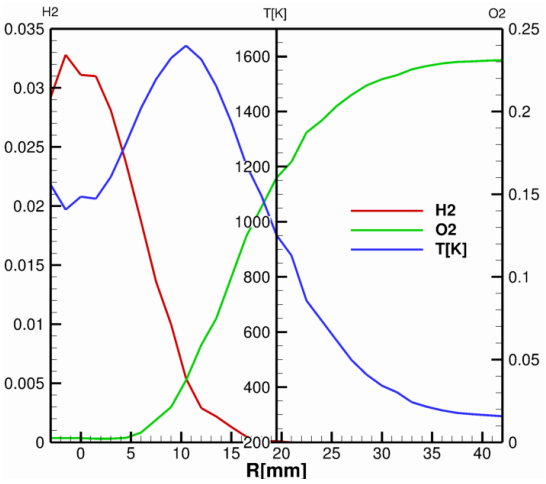
distance from the flame axis. Figure 3.3 exhibits temperature and concentrations of H_2 and O_2 distribution at a specific height, and Fig. 3.3 is consistent with Fig. 3.2 in terms of radial distance of maximum temperature at a specific height.



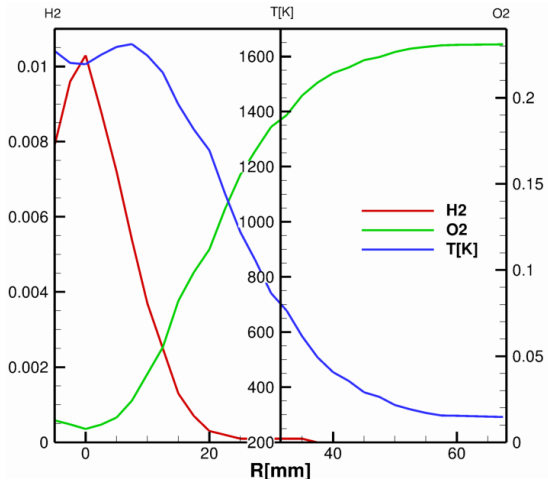
(a) Height: $x/D = 2.5$



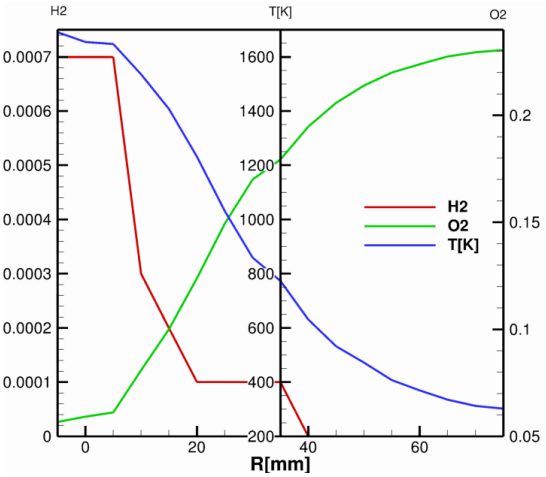
(b) Height: $x/D = 10$



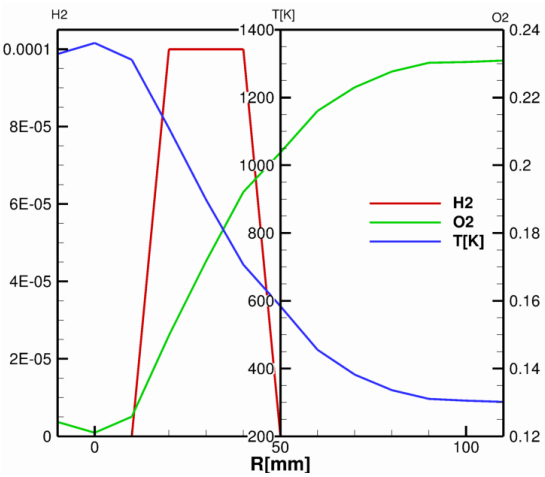
(c) Height: $x/D = 20$



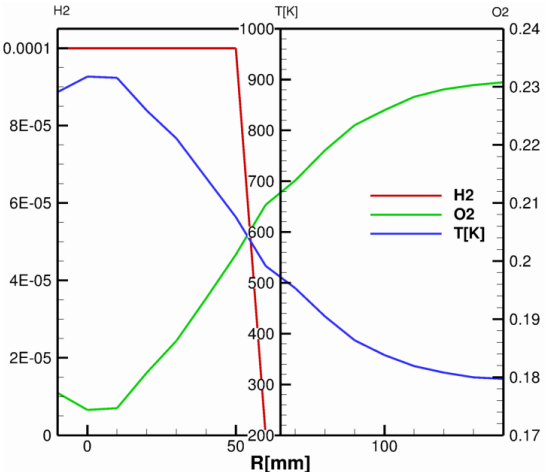
(d) Height: $x/D = 30$



(e) Height: $x/D = 40$



(f) Height: $x/D = 50$



(g) Height: $x/D = 70$

Figure 3.3: Temperature and species concentration distribution on specific heights

Figures 3.3(a) 3.3(b) 3.3(c) and 3.3(d) represent characteristic of non-premixed flame spread which show that temperature varies with the change of fuel/air ratio between rich limit and lean limit. Furthermore, these figures exhibit the maximum temperature at a specific height occurs when fuel/air ratio is closed to stoichiometric mixture. Figures 3.3 shows that the concentrations of H_2 at heights $x/D = 40$, $x/D = 50$, $x/D = 70$ is extremely low; thus we will not choose positions from these heights for our further study. We will list temperature and species concentrations at

seventeen specific points in three tables to explain the reason that we choose the final investigated locations.

The points listed in tables include three types of locations, namely, flame center, maximum temperature (where the temperature is maximum for the chosen x/D), and flame boundary (where the H_2 concentration is close to zero). In the following three tables, data are listed for these types of location. Each case is labeled (i-j): the first number, i, corresponds to the type of position; namely, 1, 2, and 3 represent flame axis, maximum temperature and flame boundary, respectively; the second number, j, increases with the value of x/D.

3.1 Flame Axis

The time averaged temperature, species concentration and the corresponding root mean square (RMS) along the flame axis are listed in Table 3.1. The mean values in the table are the ensemble mean values rather than the Favre mean values; values of RMS are the variables' root mean square fluctuations. The ensemble-average or time averaging values is obtained by integration over a long time interval; the Favre average or density-weighted averaged is calculated from the average of a variable, and most time computed from a conservation equation (for details, see Warnatz *et al.* [23]). RMS in statistics are the square root of the arithmetic mean of the squares of a sample.

Table 3.1 shows that flame temperature along the axis first increases and then decreases with an increase of height and reaches its peak at height around $x/D = 40$. The combustion processes of non-premixed flames depend on diffusion and mixing of fuel and oxidizer; at the nozzle exit of fuel, oxidizer concentration is much lower than fuel concentration, and at the top height of flame, fuel concentration is much lower than oxidizer concentration. The temperature and species concentrations data in Table 3.1 exhibit fuel/oxidizer mixing property which is the key characteristic for

non-premixed flames.

Table 3.1: Experimental mean values for temperature and mass fractions: flame axis

Num #	x/D	r [mm]	Type	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
1-1	2.5	0	Mean	282.7	0.0001	0.9308	0.0672	0.002
			RMS	5.2	0.0002	0.0007	0.0006	0.0009
1-2	10	0	Mean	469.1	0.0001	0.9146	0.0578	0.0275
			RMS	95.9	0.0003	0.0103	0.0039	0.0136
1-3	20	0	Mean	1091.8	0.0026	0.8684	0.0311	0.098
			RMS	230.8	0.0059	0.0174	0.0079	0.0243
1-4	30	0	Mean	1572.2	0.0078	0.8377	0.0103	0.1441
			RMS	234.2	0.0213	0.0184	0.0069	0.0211
1-5	40	0	Mean	1655.1	0.0592	0.8057	0.0007	0.1343
			RMS	279.2	0.0473	0.0154	0.0015	0.0332
1-6	50	0	Mean	1361.3	0.1211	0.786	0	0.0929
			RMS	361.5	0.0367	0.0096	0.0001	0.0309
1-7	70	0	Mean	906.2	0.1744	0.7785	0.0001	0.0471
			RMS	192.7	0.0173	0.004	0	0.0147

3.2 Maximum Temperature

The time averaged temperature, species concentration and the corresponding root mean square (RMS) at the maximum temperature are listed in the Table 3.2. We find that two points at the maximum temperature line are coincident with flame axis, and these two points are located at heights $x/D = 50$ and $x/D = 70$, separately.

Table 3.2: Experimental mean values for temperature and mass fractions: maximum temperature

Num #	x/D	r [mm]	Type	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
2-1	2.5	6.75	Mean	2030.3	0.0576	0.7846	0.0012	0.1565
			RMS	85.4	0.0188	0.0081	0.0012	0.0136
2-2	10	9	Mean	1700.8	0.0386	0.8164	0.0046	0.1403
			RMS	277.3	0.0468	0.0217	0.0057	0.0315
2-3	20	10.5	Mean	1639.5	0.0376	0.8198	0.0054	0.1372
			RMS	288	0.05	0.022	0.0061	0.033
2-4	30	7.5	Mean	1644.7	0.0242	0.8256	0.0054	0.1448
			RMS	220.1	0.0393	0.0195	0.0057	0.0251
2-5	40	5	Mean	1647.8	0.0612	0.8051	0.0007	0.133
			RMS	295.4	0.0486	0.0157	0.0016	0.0345
1-6	50	0	Mean	1361.3	0.1211	0.786	0	0.0929
			RMS	361.5	0.0367	0.0096	0.0001	0.0309
1-7	70	0	Mean	906.2	0.1744	0.7785	0.0001	0.0471
			RMS	192.7	0.0173	0.004	0	0.0147

Table 3.2 shows the maximum temperature at a specific height, and the highest temperature is located at x/D=2.5 and r=6.75. The radius of a position at the maximum temperature line first increases and then decrease with increasing of height. Comparing Table 3.2 with Tables 3.1 and 3.3, we can tell that the maximum temperature line is located between flame axis and flame boundary, which means fuel and oxidizer mixed well in the middle of the non-premixed flame fields .

3.3 Flame Boundary

The time averaged temperature, species concentrations and the corresponding root mean square (RMS) at the flame boundary are listed in Table 3.3.

Table 3.3: Experimental mean values for temperature and mass fractions: flame boundary

Num #	x/D	r [mm]	Type	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
3-1	2.5	7.5	Mean	1403.6	0.1539	0.7645	0	0.0815
			RMS	142.9	0.0153	0.0044	0.0001	0.0133
3-2	10	11	Mean	1252.1	0.1265	0.7903	0.0003	0.0829
			RMS	455.7	0.0625	0.0169	0.0012	0.0469
3-3	20	16.5	Mean	1206.8	0.1246	0.7929	0.0005	0.082
			RMS	445	0.0644	0.0181	0.0016	0.0469
3-4	30	20	Mean	1259.8	0.112	0.7946	0.0003	0.0931
			RMS	380.8	0.0609	0.0185	0.0011	0.0436
3-5	40	10	Mean	1535.7	0.0809	0.7998	0.0003	0.119
			RMS	338.2	0.052	0.0149	0.001	0.038

Table 3.3 shows the radius of flame boundary first increase and then decrease with increasing of height; and the location where fuel (H_2) concentration is no more than 0.05% is regarded as a boundary point.

3.4 Initial Condition

Experimental data in Tables 3.1, 3.2 and 3.3 show the time averaged temperature and species concentration distributions in the global flame fields. This distribution style is consistent with fuel/oxidizer diffusion features, and it represents characteristics of non-premixed flame spread.

In order to reduce repetitive work, we will choose only five positions from the database. These positions include one point from the flame axis, and four points from the maximum temperature line. The fuel (H_2) concentration at the flame boundary is too low to be analyzed, and a small statistical errors may significantly influence comparisons between numerical results and experimental data. Thus, the data from

flame boundary, Table 3.3, will not be selected for further investigation. The oxidizer concentration along the flame axis is also very low, and it is difficult to make a fair comparison with the DDS model with such a low oxidizer concentration. Thus, only one point at the flame axis will be selected as a testing sample. In the maximum temperature line, the fuel/air ratio is close to that of a stoichiometric mixture, and lots of useful information will be obtained if the location at the maximum temperature line is studied in the DDS model. Therefore, four points from the maximum temperature line are chosen.

The initial conditions for the chosen locations are listed in Table 3.4, including temperature and mass fractions of four major species in H_2-air combustion, namely, H_2, O_2, H_2O and N_2. The experimental mean values for temperature and mass fractions are listed in Table 3.5. In the next chapter, we will compare the DDS model results with the database at the same locations.

Table 3.4: Initial conditions for temperature and mass fractions

Case #	x/D	r [mm]	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
1	2.5	0	283.3	0.0000	0.9295	0.0682	0.0023
2	2.5	6.75	2095.6	0.0531	0.7834	0.0011	0.1623
3	10	9	1970.7	0.0206	0.8184	0.0011	0.1598
4	20	10.5	1578.5	0.0111	0.8467	0.0083	0.1340
5	30	7.5	1807.6	0.0068	0.8311	0.0023	0.1598

Table 3.5: Experimental mean values for temperature and mass fractions

Case #	x/D	r [mm]	Type	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
1	2.5	0	Mean	282.7	0.0001	0.9308	0.0672	0.002
			RMS	5.2	0.0002	0.0007	0.0006	0.0009
2	2.5	6.75	Mean	2030.3	0.0576	0.7846	0.0012	0.1565
			RMS	85.4	0.0188	0.0081	0.0012	0.0136
3	10	9	Mean	1700.8	0.0386	0.8164	0.0046	0.1403
			RMS	277.3	0.0468	0.0217	0.0057	0.0315
4	20	10.5	Mean	1639.5	0.0376	0.8198	0.0054	0.1372
			RMS	288	0.05	0.022	0.0061	0.033
5	30	7.5	Mean	1644.7	0.0242	0.8256	0.0054	0.1448
			RMS	220.1	0.0393	0.0195	0.0057	0.0251

Chapter 4 Results and Discussion

4.1 Numerical Solution and Discussion

In this chapter, combustion processes at selected positions are investigated from the database of the co-flow TU Darmstadt-Flame Data Base Schneider *et al.* [106], and detailed information of experimental data are listed in Tables 3.4 and Table 3.5 of the preceding chapter. Time series of velocities, temperature and species concentrations at these positions calculated by the low-order DDS model developed in Chapter 2. Numerical solutions are compared with experimental data; discussion regarding behaviors of the computed results are provided; and the potential factors that may cause discrepancies between computation and experiment are analyzed. In addition, the DDS model in different types of flame locations will be tested to check whether the DDS model can mimic the combustion process in the whole flow field. The bifurcation parameters for two reduced chemical mechanisms for the DDS model are studied, and the optimized parameters for this DDS model will be chosen.

4.2 Regime Maps

For the sake of simplicity, the bifurcation parameters in Eqs. (2.34) are initially studied with the homogeneous and isotropic assumption. Then, regime maps with inhomogeneity and anisotropy conditions are investigated. Recall that in Chapter 2, bifurcation parameters with homogeneous and isotropic assumption were set as

$$\beta_u = \beta_v = \beta_w = \beta,$$

$$\gamma_{uv} = \gamma_{uw} = \gamma_{vu} = \gamma,$$

$$\gamma_{vw} = \gamma_{wu} = \gamma_{wv} = \gamma,$$

$$\gamma_{uT} = \gamma_{vT} = \gamma_{wT} = \gamma_T,$$

$$\gamma_{uY_i} = \gamma_{vY_i} = \gamma_{uwY_i} = \gamma_{Y_i}.$$

We remark that neither the fluid flow nor the chemistry can be expected to be either homogeneous or isotropic in this case, but this assumption provides a tractable starting point. Moreover, we again note that no such assumption is needed in a complete LES, because all bifurcation parameters can be calculated from high-pass filtered resolved-scale results. The values of parameters associated with species i, e.g., β_{Y_i}, α_{Td_i} etc., should be analyzed individually, because every species has its own characteristics. We present regime maps, β vs. γ for two different mechanisms in this section. Once the different regimes are identified by their power spectral density (detailed analysis provided in [105]), we can choose regimes where chaotic behavior is present, and apply the corresponding bifurcation parameters of the chaotic region in Eqs. (2.34) to evolve the DDS.

4.2.1 β vs. γ in Homogeneous and Isotropic Assumption

In this section, regime maps for five locations are shown. The bifurcation parameters used in the regime maps are β vs. γ, and power spectral density is used to check fluid flow status. Power spectral density (PSD) is used to describe how the power of a signal or time series is distributed over the different frequencies, and it commonly expressed in watts per Hertz (W/Hz), but in this case its unit is (dB/Hz)). With a homogeneous and isotropic assumption regime map, values of bifurcation parameters, β and γ, are determined, and they are listed in Table 4.1. Since the regime maps in the five location are the same under the current condition, only one regime map is exhibited in Fig. 4.1.

Table 4.1: Values of parameters with homogeneous and isotropic assumption

Case #	x/D	r [mm]	β	γ
1	2.5	0.00	3.6416	0.2360
2	2.5	6.75	3.6416	0.2360
3	10	9.00	3.6416	0.2360
4	20	10.5	3.6416	0.2360
5	30	7.50	3.6416	0.2360

Figure 4.1(a) indicates the variation of PSD status is a function of the bifurcation parameters, and Fig. 4.1(b) is the corresponding zoom in chart including the useful regime. The color table, Fig. 4.1(c), has 14 colors, which includes the flow behavior states indicated. McDonough [105] observed that the basic sequence corresponding to increasing β with γ fixed is from steady to periodic, to subharmonic, to chaotic states. The regime map, shown in Fig. 4.1(c), is made up of 14 colors, and these colors constitute a series of color islands which represent different flow states. In color table, Fig. 4.1(c), colors 0 and 13 represent steady and divergent states, individually, and they cover much of the regime map; for non-steady behavior, colors 1 and 2 indicate periodic states; colors designated 3 through 6 denote quasiperiodic states; and colors designated 7 through 12 express chaotic states.

In this work, chaotic noisy states are reached when β beyond 3.5 with γ is fixed at 0.3. Recall that the Reynolds number of fuel is 10,000, and we are studying a turbulent diffusion flames. Thus, chaotic regimes are of interest area for this studies. In this work, the region with color 11 is chosen as an useful region for parameters investigation. There are a couple of regions are filled with color 11 in the current regime map, and the region where β beyond 3.5 with γ is around 0.3 is selected as an interesting area. A grey point (β=3.6416, γ=0.2360) is marked in the interesting area in Fig. 4.1(a), and more detailed information of this area is displayed in the

zoom in of Fig. 4.1(b). Then, then values of β and γ that are determined by the regime map are used in the DDS model. As we mentioned, the regime map is used to determine an useful region for bifurcation parameters, and the specific values of bifurcation parameters are selected randomly in this useful region. The SGS DDS model is sensitive to the value of bifurcation parameters, and the regime map provide a good way to approach real physical settings for these parameters.

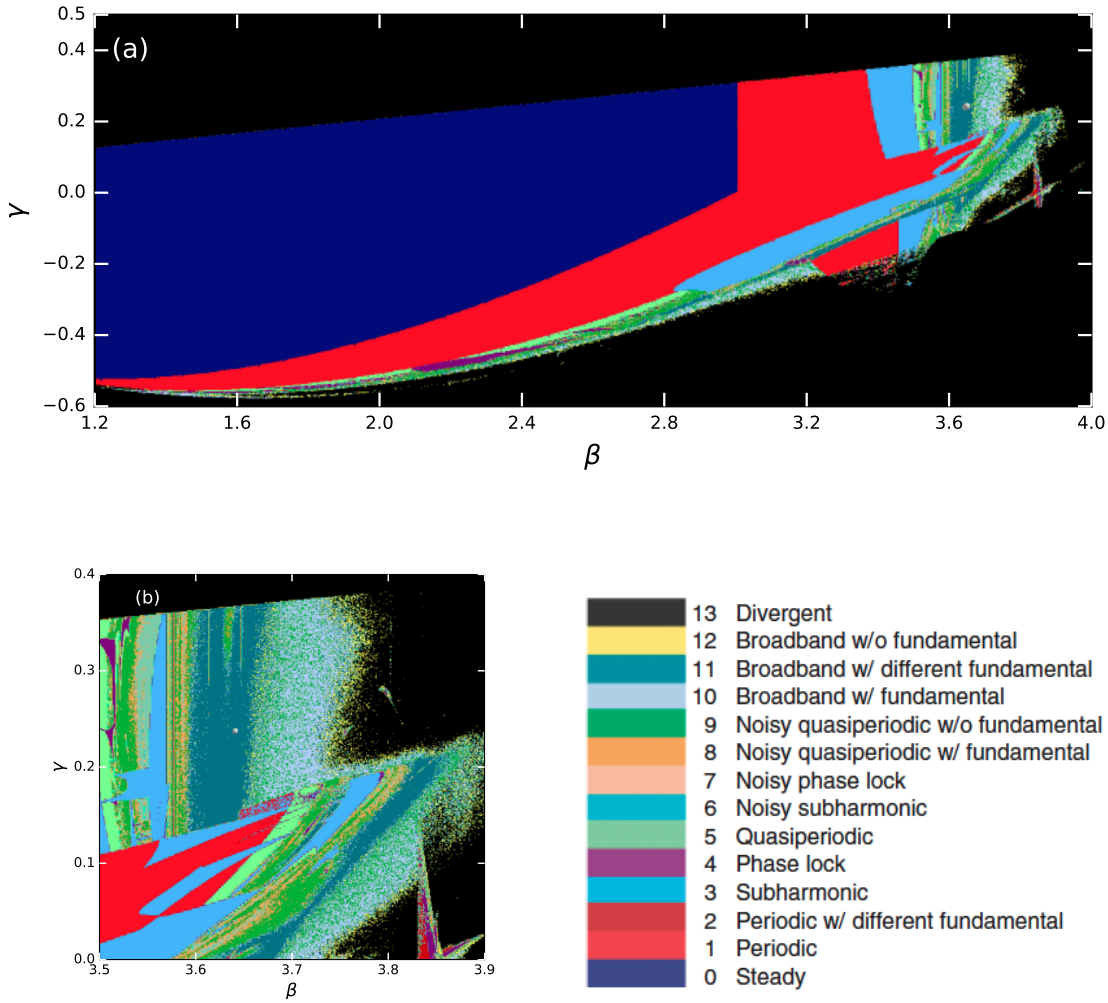


Figure 4.1: Regime map with Homogenous and Isotropic Assumption: (a) β vs. γ at location $x/D = 2.5$, $r = 0$; (b) zoom at interested region; (c) color table for flow states in regime maps

4.2.2 β vs. γ with Reacting N_2 Mechanism

With homogeneous and isotropic assumption, values of bifurcation parameters, β and γ, in reacting N_2 mechanism are the same as the values in N_2 dilution mechanism, shown in Table 4.1. Thus, regime maps and values of parameters with reacting N_2 mechanism will not repeat.

4.2.3 β vs. γ in Inhomogeneous and Anisotropic Conditions

Next, bifurcation parameters β and γ are investigated in inhomogeneous and anisotropic conditions, and they are defined as

$$\beta_u = \beta_1,\ \beta_v = \beta_2,\ \beta_w = \beta_3,$$

$$\gamma_{uv} = \gamma_{uw} = \gamma_u = \gamma_1,$$

$$\gamma_{vu} = \gamma_{vw} = \gamma_v = \gamma_2,$$

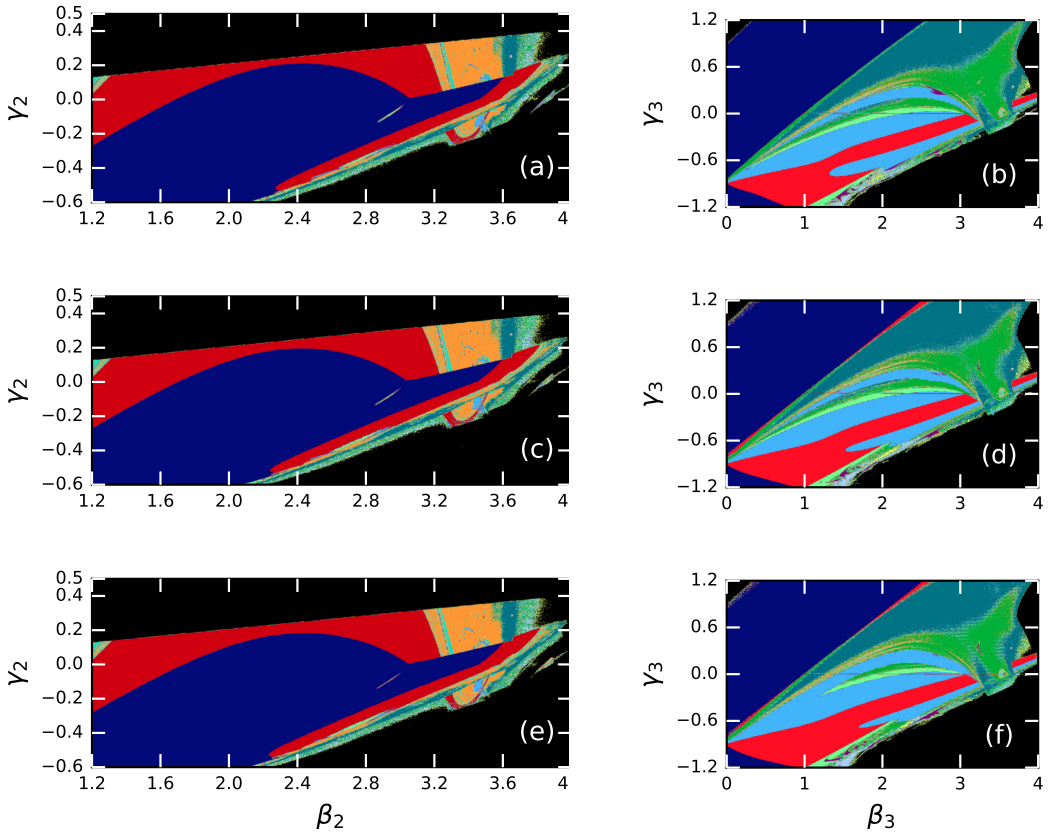
$$\gamma_{wu} = \gamma_{wv} = \gamma_w = \gamma_3.$$

Values of β and γ in five locations are listed in Table 4.2, and the corresponding regime maps are shown in Fig 4.2. Three regime maps are employed to determine useful regimes for bifurcation parameters at every single location. The PSD sequence in the regime maps is a function of β and γ, and a 2D regime map requires and only requires a pair of bifurcation parameters for every calculation. When we construct first regime map for β_u and γ_u, the other bifurcation parameters should be fixed. Since the values of bifurcation parameters under homogeneous and isotropic assumptions are known, and we can use them for β_u and γ_u. In the same way, the values of β_u and γ_u are known and it is not necessary to construct a new regime map for them again. In constructing the regime map for β_v and γ_v, the values of β_u and γ_u are fixed, and set $\beta_v = \beta_w$ and $\gamma_v = \gamma_w$ because only a pair of bifurcation parameters are needed for a regime map. The corresponding figures are displayed in Figs. 4.2(a)(c)(e)(g)(i)

for the five locations of Table 4.2. Similarly, when we construct regime maps for β_w and γ_w, the values of β_u β_v γ_u, and γ_v are fixed. Regime maps for β_w and γ_w are exhibited in Figs. 4.2(b)(d)(f)(h)(j) for the five cases. Again, a grey point is marked at an interesting region in each regime map.

Table 4.2: Value of parameters with inhomogeneous and anisotropic conditions

Case #	x/D	r [mm]	β_1	β_2	β_3	γ_1	γ_2	γ_3
1	2.5	0.00	3.6416	3.6304	3.6528	0.2360	0.2404	0.3372
2	2.5	6.75	3.6416	3.6416	3.6416	0.2360	0.2668	0.3020
3	10	9.00	3.6416	3.6416	3.6304	0.2360	0.2800	0.3284
4	20	10.5	3.6416	3.6304	3.6640	0.2360	0.2492	0.3592
5	30	7.50	3.6416	3.6416	3.6640	0.2360	0.3064	0.3460



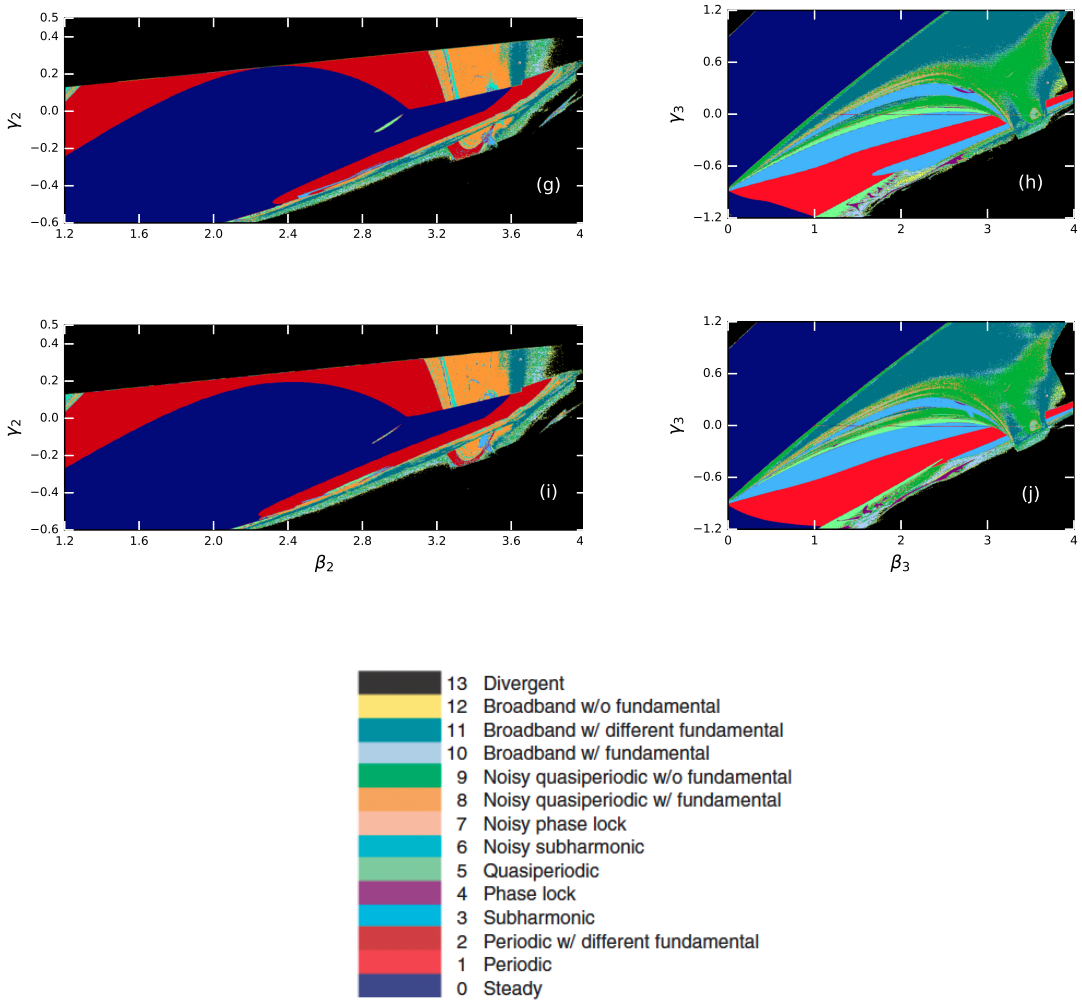
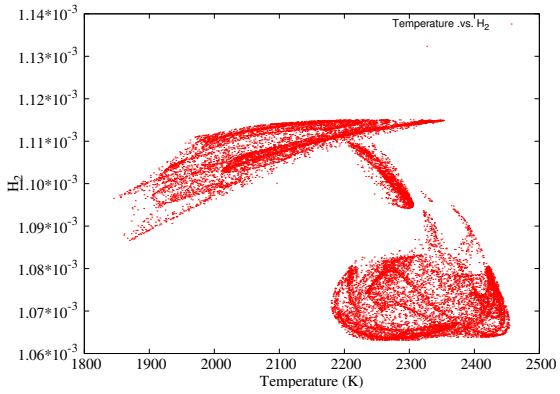


Figure 4.2: Regime map with inhomogeneous and anisotropic condition: (a) β_2 vs. γ_2 at location $x/D = 2.5$, $r = 0$; (b) β_3 vs. γ_3 at location $x/D = 2.5$, $r = 0$; (c) β_2 vs. γ_2 at location $x/D = 2.5$, $r = 6.75$; (d) β_3 vs. γ_3 at location $x/D = 2.5$, $r = 6.75$; (e) β_2 vs. γ_2 at location $x/D = 10$, $r = 9$; (f) β_3 vs. γ_3 at location $x/D = 10$, $r = 9$; (g) β_2 vs. γ_2 at location $x/D = 20$, $r = 10.5$; (h) β_3 vs. γ_3 at location $x/D = 20$, $r = 10.5$; (i) β_2 vs. γ_2 at location $x/D = 30$, $r = 7.5$; (j) β_3 vs. γ_3 at location $x/D = 30$, $r = 7.5$; (k) color table for flow states

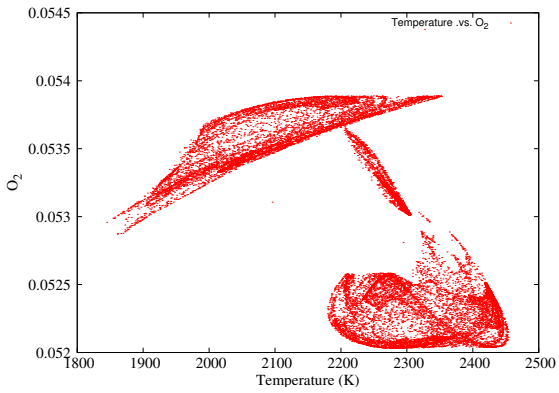
The regime maps, Figs. 4.1 and 4.2, demonstrate that the distribution of PSD is associate with bifurcation parameters settings, and the settings also affect flow states layouts. It's a truth that flow state is a function of physical settings. Figures. 4.2 exhibit that the interesting region in regime map, β_3 vs. γ_3, is much bigger than that in both regime maps, β_2 vs. γ_2 and β_1 vs. γ_1 (see Figs. 4.2) .

4.3 Temperature-Phase Portraits

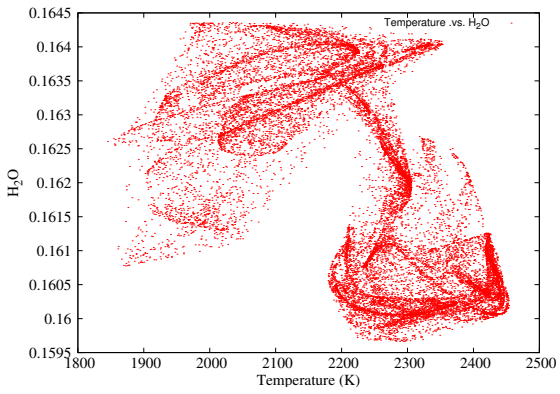
In this subsection, temperature-species portraits at location $x/D = 2.5$, $r = 6.75$ with N_2 reaction mechanism are presented, and simplified discussion and analysis on the phase-portrait phenomena are presented. A phase portrait usually only contains trajectories of solutions.



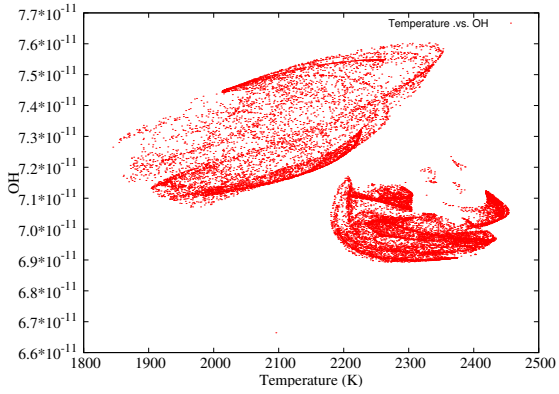
(a) Temperature vs. H_2



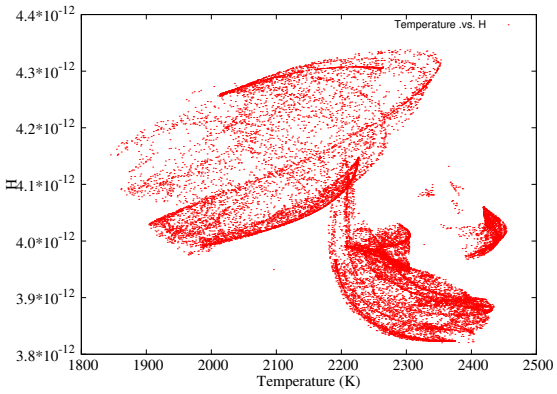
(b) Temperature vs. O_2



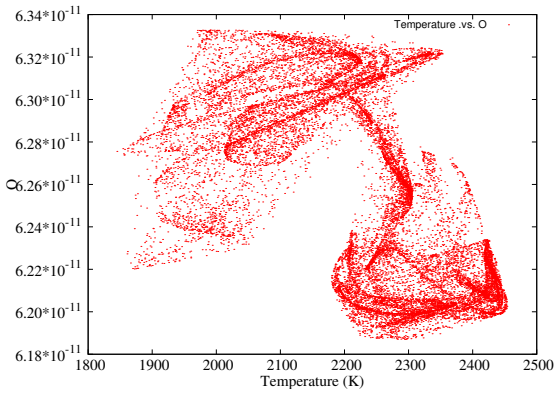
(c) Temperature vs. H_2O



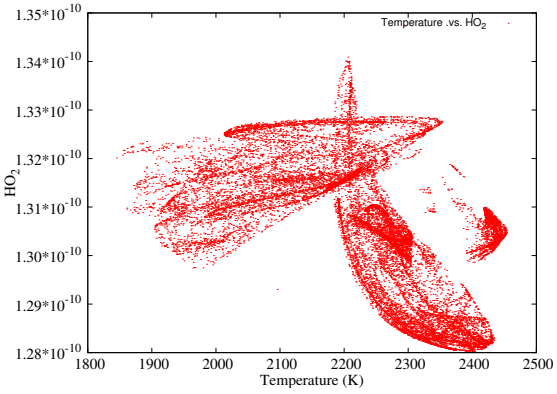
(d) Temperature vs. OH



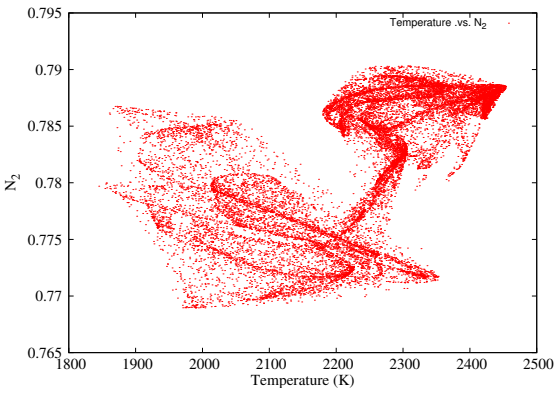
(e) Temperature vs. H



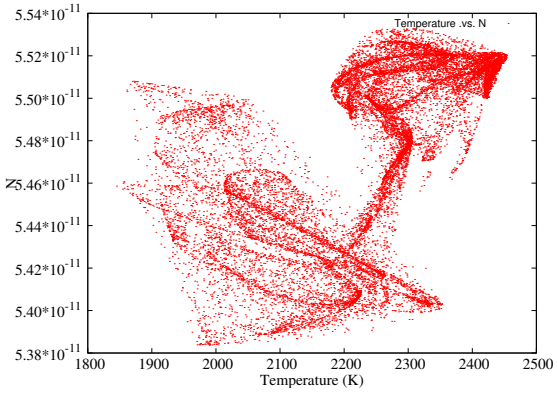
(f) Temperature vs. O



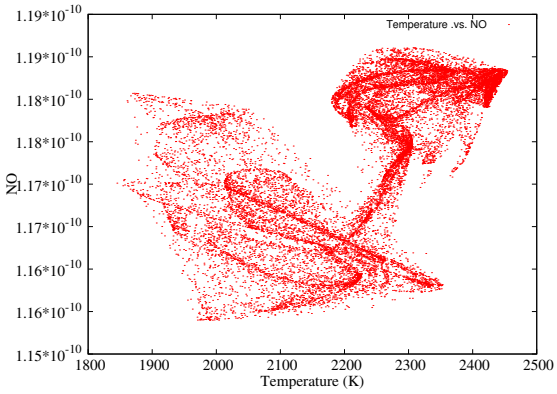
(g) Temperature vs. HO_2



(h) Temperature vs. N_2



(i) Temperature vs. N



(j) Temperature vs. NO

Figure 4.3: Temperature-Species Concentration Phase Portraits

The species concentrations are determined from the reduced mechanism listed in

Chapter 2. This includes both forward and backward reactions, where forward reaction constants and reverse reaction constants vary among different reactions at different temperatures. As it is discussed in Chapter 1, the typical chemical reaction rates are expressed in the form of an Arrhenius law [23]. The temperature phase portraits reveal the relationship between variation of species concentrations and temperature. Combustion reactions usually release a large amount of heat because they are exothermic reactions and have negative enthalpy. This results in increasing the temperature, which changes the equilibrium constant; this causes fluctuation of species concentrations.

The reduced chemical reaction mechanism of this work includes four different types of elementary reactions in radical chain reactions: chain initiation, chain propagation, chain propagation and chain termination. Chain propagation involves intermediate species, including radicals and/or atoms (in this paper they are atoms); they are unstable and have rapid reaction rates, and typically their concentrations are quite low. Figures 4.3 demonstrate that intermediate atoms, such as OH, H, O, HO_2, N, and NO have low concentrations.

Figures 4.3(a), 4.3(b), 4.3(c) show phase-portraits of temperature vs. H_2, O_2 and H_2O concentrations, respectively. In the range 1800–2100 K, species concentrations of H_2, O_2 and H_2O are at a relative high level; then in the range 2200–2400 K, species concentrations decrease. The maximum temperature of the hydrogen-oxygen flame is 3400 K, and that of the hydrogen-air flame is 2400K with an exact stoichiometric mixture. When species H_2 and O_2 react with other intermediate species, their concentrations decrease, and as a result ambient temperature increases. Their concentrations are at the lowest level when temperature is in the range 2200–2400 K, which is the range of maximum hydrogen-air flame temperatures .

Species concentrations N_2, N, and NO shown in Figs. 4.3(h), 4.3(i), 4.3(j) are described by the extended Zeldovich mechanism in the DDS model. The NO_x produc-

tion in this case was dominated by the thermal NO_x mechanism, and this mechanism is very sensitive to temperature. Thermal NO_x is formed when temperature is above 1500 K, and it forms in a significant amount when flame temperature reaches 1800 K; the higher the flame temperature the higher the concentration of thermal NO_x. Figures 4.3(h), 4.3(i), 4.3(j) show that a rapid increase in the rate NO_x formation occurs when temperatures are above 2200K, and this phenomenon meets the characteristic of thermal NO_x formation. Thus, many efforts should be made to make sure the temperature is not overestimated, since over predicted temperature will lead to an unrealistically over predicted levels of NO_x.

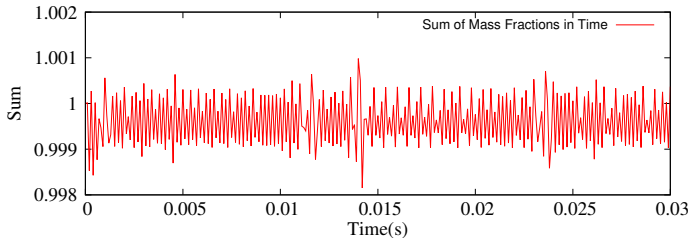
4.4 Numerical Results

In this section, further comparison between numerical solutions and experimental data are performed in all five locations. In detail, time averaged values of temperature and major species concentrations are listed in tables; time series of dependent variables, such as temperature and major species concentrations are presented in various figures that follow. The sum of mass fractions is directly calculated using all species instead of employing it to eliminate one species. Numerical solutions of two mechanisms with two different conditions are investigated: numerical solution of N_2 dilution mechanism with homogeneous and isotropic assumption; numerical solutions of reacting N_2 mechanisms with homogeneous and isotropic assumption; numerical solution of N_2 dilution mechanism with anisotropic and inhomogeneous conditions. The DDS model case number (i) in the tables and figures, corresponds to the type of mechanisms with a specific condition; namely, case number 1, 2 and 3 represent the N_2 dilution mechanism with homogeneous and isotropic assumption, the reacting N_2 mechanisms with homogeneous and isotropic assumption, the N_2 dilution mechanism with anisotropic and inhomogeneous conditions, respectively. The DDS model in the reacting N_2 mechanisms with anisotropic and inhomogeneous conditions is treated as

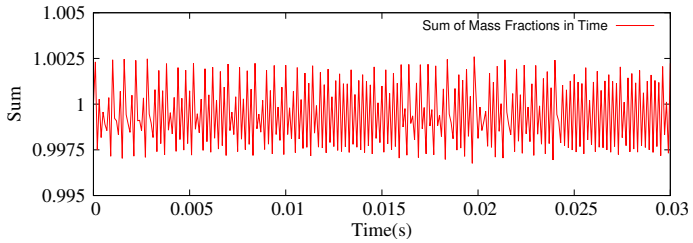
case 4, which is calculated at location $x/D = 2.5$, $r = 0$ only. In order to test the DDS capability in global flow fields, the numerical model is applied to multiple locations, and numerical solutions of the two mechanisms with two different conditions are performed at the five chosen points that are mentioned in Chapter 3.

4.4.1 Validation of model

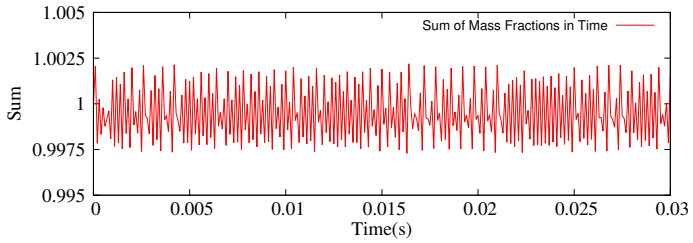
McDonough and Zhang [102] provided a 2-D DDS for combustion processes, but no one has previously constructed a 3-D case. We will use temporal sum of mass fraction fluctuations as a measurement to validate the DDS model. Instead of forcing the sum to unity by calculating all but one species and then setting it to satisfy the required unity value as others have done, we directly calculate all species concentrations and then observe the sum of mass fractions. These results are display in Figs. 4.4.



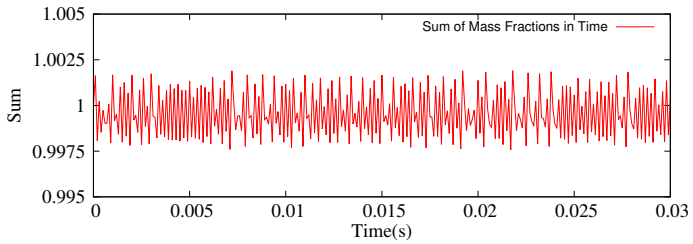
(a) Location: $x/D = 2.5$ with $r = 0$



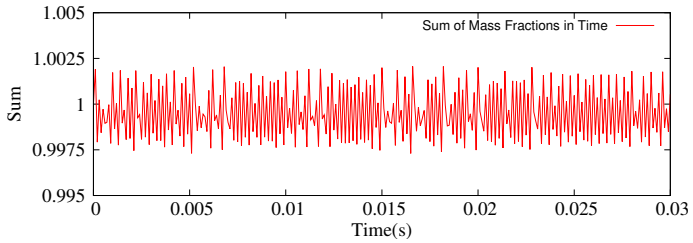
(b) Location: $x/D = 2.5$ with $r = 6.75$



(c) Location: $x/D = 10$ with $r = 9$



(d) Location: $x/D = 20$ with $r = 10.5$



(e) Location: $x/D = 30$ with $r = 7.5$

Figure 4.4: Sum of mass fractions in Time

Figures 4.4 show that the sum of mass fractions is close to unity for all five locations. The time averaged of sum of mass fractions for Fig. 4.4(a) is 0.9998, for Fig. 4.4(b) is 0.9996, for Fig. 4.4(c) is 0.9997, for Fig. 4.4(d) is 0.9997, for Fig. 4.4(e) is 0.9997, with these values showing less than 0.1% discrepancies from the required value. These results show that the DDS model works well in terms of conservation of species mass fractions. In the following subsection, we will present computed results and compare them with experimental data to verify this model.

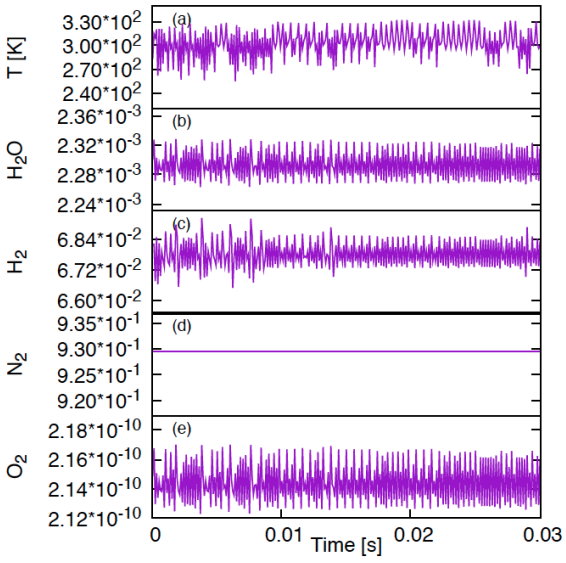
4.4.2 Solution comparisons with experimental data

4.4.2.1 Location: x/D=2.5, r=0

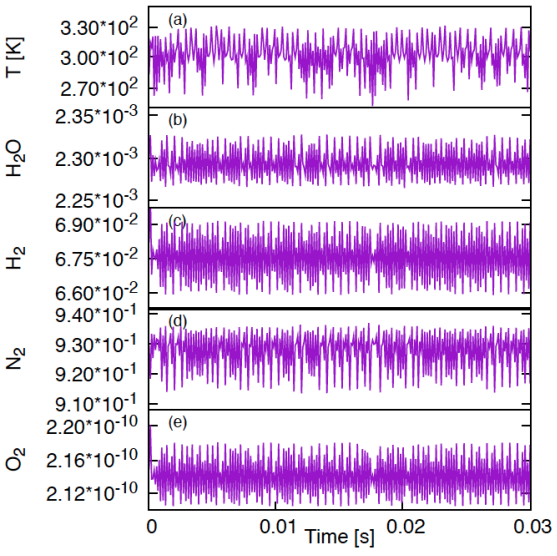
Time-averaged numerical solutions at location $x/D = 2.5$, $r = 0$ are shown in Table 4.3, and time series of dependent variables behaviors are presented in Figs. 4.5. Comparison between numerical solutions and experimental data is included in Table 4.3 and Figs. 4.5.

Table 4.3: Solution and comparison at location: $x/D = 2.5$, $r = 0$

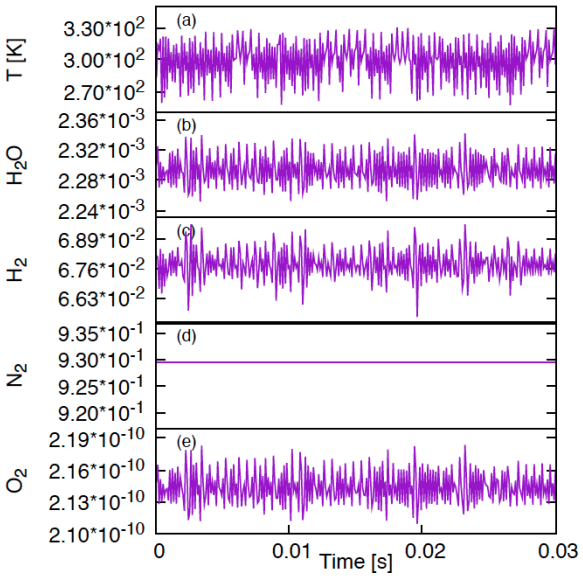
	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
Experiment mean	282.7	0.0001	0.9308	0.0672	0.002
DDS model #1	301.6	2.09E-10	—	0.0623	0.0022
DDS model #2	300.9	2.11E-10	0.938	0.0598	0.00229
DDS model #3	299.3	2.09E-10	—	0.0623	0.0021
Experiment RMS	5.2	0.0002	0.0007	0.0006	0.0009
DDS RMS #1	14.09	2.64E-21	—	2.37E-4	2.97E-07
DDS RMS #2	14.1	2.67E-21	5.14E-02	2.29E-4	3.09E-7
DDS RMS #3	14.09	2.66E-21	—	2.39E-4	2.98E-07
Discrepancy #1	18.86	1.0E-4	—	0.00469	0.0002
Discrepancy #2	18.22	9.99E-5	0.00707	0.00736	0.00029
Discrepancy #3	16.63	1.0E-4	—	0.0047	0.0002
Error #1 (%)	6.67	-99.99	—	-7.279	10.194
Error #2 (%)	6.44	-99.99	0.759	-10.95	14.58
Error #3 (%)	5.88	-99.99	—	-7.279	10.27



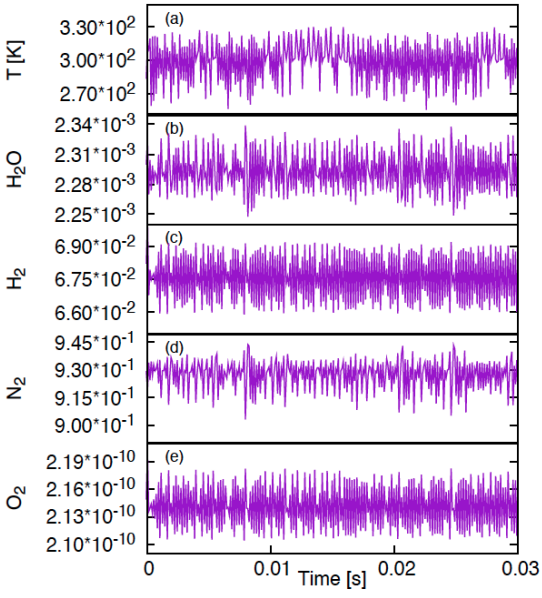
(a) Case 1



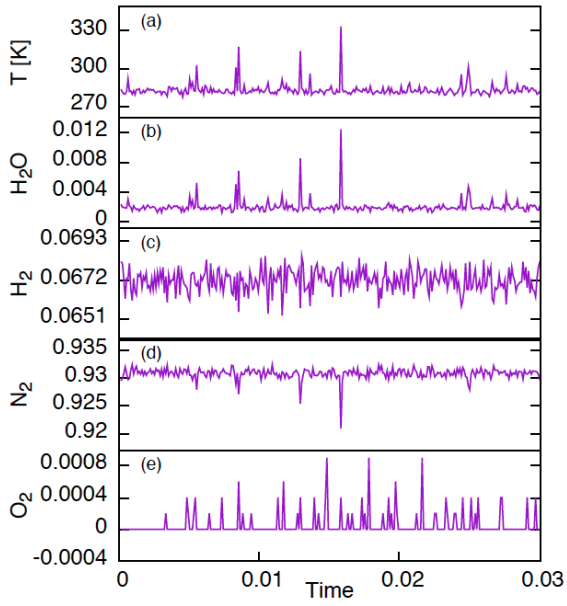
(b) Case 2



(c) Case 3



(d) Case 4



(e) Experiment data

Figure 4.5: Numerical solution and comparison at location: $x/D = 2.5$ with $r = 0$

Table 4.3 shows computed results of temperature, major species concentrations for the DDS model in three different cases. The DDS model tends to over predict temperature in all three cases. The value of temperature in case 3 is the smallest one in the numerical results, which also holds the smallest discrepancy. All of the DDS RMS for temperature term are larger than the corresponding experimental RMS. Concentration of O_2 is very small in the current position, since this point is located at the center of the flame and close to fuel exit. The numerical prediction of O_2 concentration is close to zero, and the experimental record is also very small. This case is chosen from flame axis, where fuel directly exits from the nozzle. Fuel concentration along this flame axis is very high; on the other hand air concentration is very low, and is close to zero. Though the percentage error of O_2 concentration is very large, the absolute discrepancy is relatively small. Furthermore, the corresponding experimental RMS of O_2 concentration is 0.0002, which is much larger than any of the DDS O_2 concentration RMS predictions in three cases.

Figures 4.5 (a)(b)(c)(d) display time series of behaviors of temperature, species concentrations of H_2O, H_2, N_2 and O_2 in four cases. The case 4 includes the N_2 reaction mechanism with inhomogeneous and anisotropic conditions. All variables in the numerical solutions exhibit turbulent fluctuation behaviors as is seen in the experimental data. Time series of fluctuations in case 3 and case 4 are more similar to the experimental data than the other two cases, which proves isotropic and homogeneous assumption is not valid in this non-premixed combustion situation.

Figures 4.5 (a)(b)(c)(d) show that fluctuations of variables do not produce large-amplitude in all four cases, however, large-amplitude occur occasionally in the fluctuations of the experimental data. Recall that the DDS model in this work is derived for SGS, and the calculation is local and only for a single point. Numerical solutions of the DDS model are set by initial condition only, and they will not be affected by adjacent points. In contrast, temperature and species concentrations in the experimental data depend on both initial conditions and surrounding flow behaviors, such as fuel flow and air flow movement. The large-amplitude occur several times in the experimental case, and this phenomenon may be caused species concentrations in the monitored position being disturbed by the adjacent position due to fluid flow. However, with a small time scale (reaction times) in both the DDS model and the experimental data, energy and species concentrations in the adjacent points should not transport to the investigated location. Furthermore, the temperature model (see section 2.5) and the time scale that are used in the DDS model probably do not meet with the experimental setting. Therefore, in order to obtain high-amplitude fluctuations, the temperature model and scaling method in the DDS should be validated by the experimental data from further studies.

Figures 4.5 show fluctuations frequency in numerical solution is higher than the experiment data. A possible reason for this difference is that the fluctuation scale in the DDS model is more sensitive than experiment equipment, and the former can

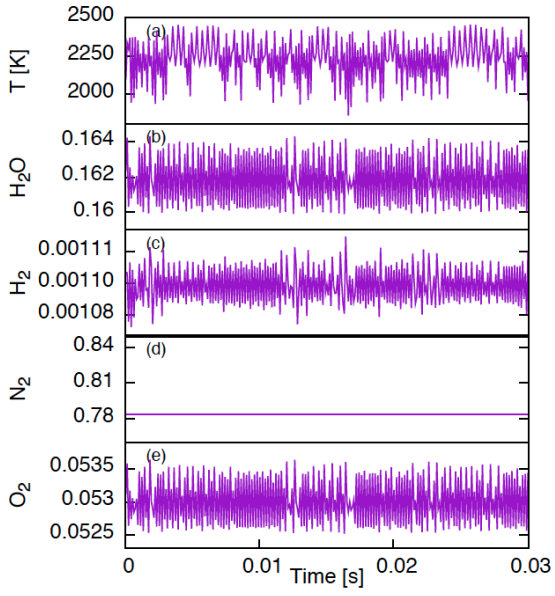
record smaller variation than the latter; another reason is that the computed time scale is incompatible with the experimental data. The absolute O_2 concentration in this case is too small to be analyzed. However, the time average of temperature, species concentrations of N_2, H_2 and H_2O match with the experimental data fairly well, with the largest error being less than 15%. For the largest error percentage in H_2O concentration, the experimental RMS is 0.0009 which is bigger than the corresponding numerical RMSs in all three cases. It is noteworthy that discrepancies of species concentrations of H_2 and H_2O are much larger than the others, and discrepancy of N_2 is even larger than the corresponding experimental RMS (0.0007). Case 2 has the largest discrepancies in species concentrations of H_2 and H_2O among the three cases in Table 4.3, which may imply that the N_2 reaction mechanism is not a suitable scheme at low temperature condition (viz., temperature is less than 300K).

4.4.2.2 Location: x/D=2.5, r=6.75

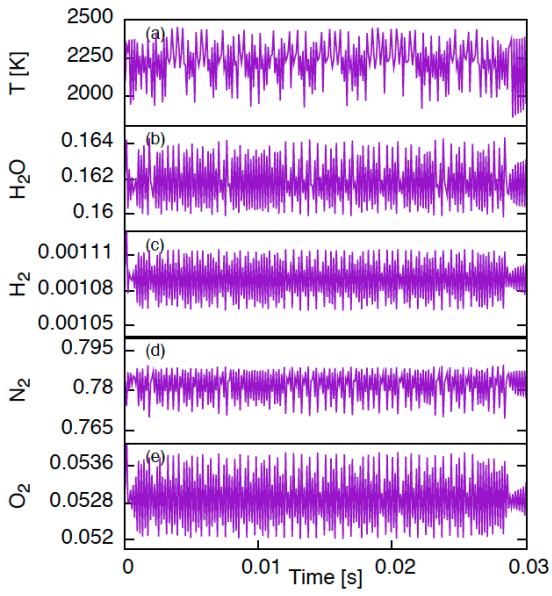
The current investigated location is chosen from the maximum-temperature line where temperature reaches the maximum value for a given height (x/D) as we mentioned in Chapter 3. Time-averaged numerical solutions at location $x/D = 2.5$, $r = 6.75$ are shown in Table 4.4, and time series of dependent variable behaviors are presented in Figs. 4.6. Comparison between numerical solutions and experimental data is included in Table 4.4 and Figs. 4.6.

Table 4.4: Solution and comparison at location: $x/D = 2.5$, $r = 6.75$

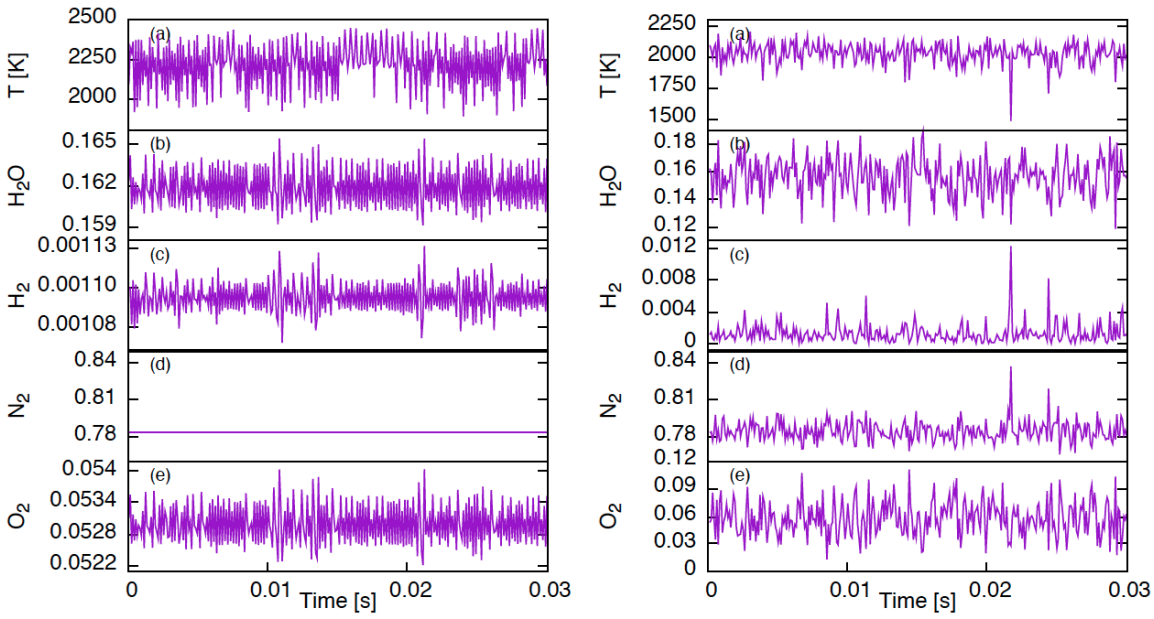
	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
Experiment mean	2030.3	0.0576	0.7846	0.0012	0.1565
DDS model #1	2228.0	0.00519	—	0.00101	0.156
DDS model #2	2228.7	0.00518	0.786	0.00096	0.161
DDS model #3	2216.4	0.00519	—	0.00101	0.156
Experiment RMS	85.4	0.0188	0.0081	0.0012	0.0136
DDS RMS #1	24.1	1.8E-4	—	7.36E-8	1.49E-3
DDS RMS #2	24.1	1.62E-4	3.63E-02	5.93E-8	1.53E-3
DDS RMS #3	24.1	1.8E-4	—	7.37E-8	1.49E-3
Discrepancy #1	197.75	0.00571	—	0.00019	0.0004
Discrepancy #2	198.4	0.0058	0.0017	0.00024	0.0044
Discrepancy #3	186.11	0.00573	—	0.00019	0.00047
Error #1 (%)	9.73	-9.91	—	-15.94	-0.258
Error #2 (%)	9.77	-10.09	0.21	-20.01	-2.83
Error #3 (%)	9.17	-9.95	—	-16.04	-0.298



(a) Case 1



(b) Case 2



(c) Case 3

(d) Experiment data

Figure 4.6: Numerical solution and comparison at location: $x/D = 2.5$, $r = 6.75$

As can be seen from Table 4.4, temperature is over predicted by the DDS model for all three cases; and the temperature term in case 3 is the smallest one in the numerical solutions, which holds the smallest discrepancy from the experimental time averaged value. There is no a huge error (recall the 99.99% discrepancy of O_2 concentration from the DDS model at location 1) in this location. The maximum percentage error of H_2 concentration occurs at case 2. Besides the concentration of H_2, the maximum percentage error of any other variables is less than 11% in all three cases. The discrepancy and the DDS RMS of H_2 concentration is far less than the corresponding experimental RMS, which is viewed as that the numerical results of H_2 concentration is an acceptable solution for a low-order model to be used on the subgrid scale.

The discrepancy in case 2 is larger than others, but the DDS RMSs in case 2 are still much less than the corresponding experimental RMS. Although the discrepancies of temperature are larger than the experimental RMS in all three cases, the percentage error are still less than 10%. Furthermore, the discrepancies of four species concen-

trations in the DDS model for all three cases are far less than the corresponding experimental RMS, and the largest percentage error is around 20%. Table 4.4 shows that numerical results in the three cases considered here nearly match mean values for species concentrations, and their discrepancies are smaller than the experimental RMS.

Figures 4.6 (a)(b)(c) show times series of dependent variables of the DDS model in three cases, and Fig. 4.6(d) exhibits the corresponding behaviors in the experimental data. All variables in the DDS model for all three cases exhibit turbulent fluctuations as in Fig. 4.6(d), which demonstrate the DDS model can mimic turbulent fluctuations in at least a qualitative way. Time series of fluctuations in case 3 show a better performance than the others, thus again demonstrating the DDS model with an non-isotropic condition can reach good turbulent behaviors.

Specifically, temperature and concentrations of H_2O, O_2 in Fig. 4.6(c) present similar fluctuations as Fig. 4.6(d) in both frequency and amplitude. Since the adjacent points in flow field are unable to affect the location that is analyzed in this work, computed solutions in the DDS model cannot repeat intermittent high-amplitude fluctuations in Fig. 4.6(c). In further studies, a temperature model that can reproduce high-amplitude fluctuations should be investigated to approximate the turbulent fluctuation behaviors seen in the experimental data. The DDS model also remains high-pass information, and this character may cause the solutions of the DDS model to exhibit higher frequency, especially in case 1 and case 2, than in the experimental data. In order to remove the higher frequency issue, a correct time scale that is close to experimental data or appropriate low-pass filtering of computed results is required.

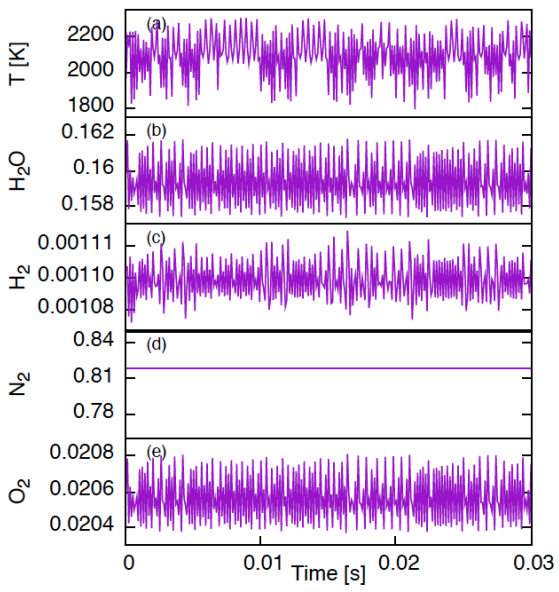
4.4.2.3 Location: x/D=10, r=9

Time-averaged numerical solutions at location $x/D = 10$, $r = 9$ are shown in Table 4.5, and time series of dependent variable behaviors are presented in Figs. 4.7.

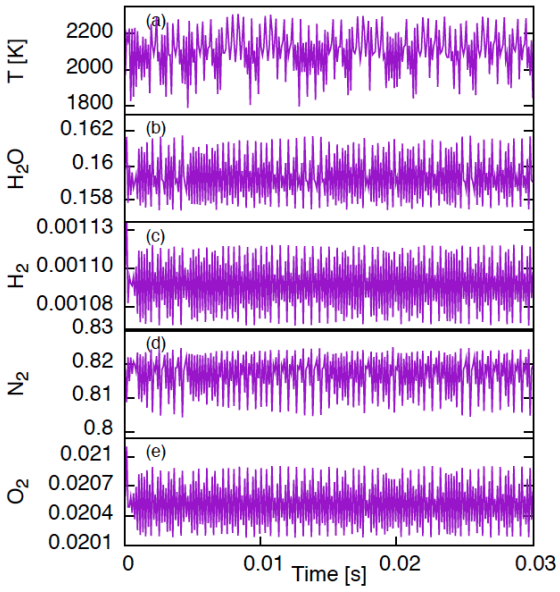
Comparison between numerical solutions and experimental data is included in Table 4.5 and Figs. 4.7.

Table 4.5: Solution and comparison at location: $x/D = 10$, $r = 9$

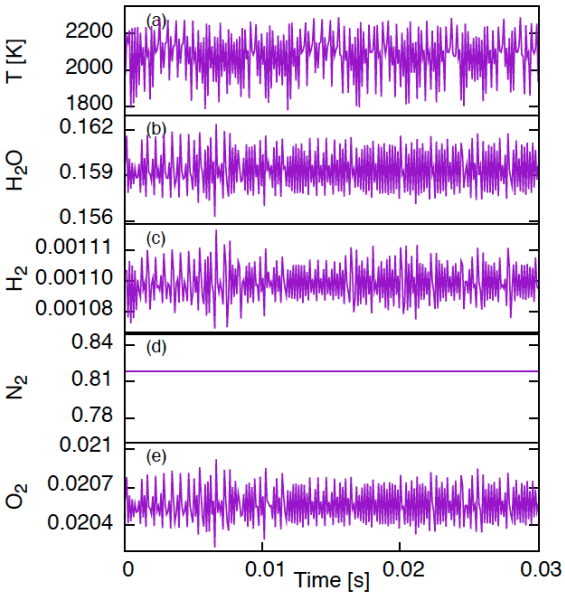
	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
Experiment mean	1700.8	0.0386	0.8164	0.0046	0.1403
DDS model #1	2095.52	0.00201	—	0.00101	0.154
DDS model #2	2096.01	0.00201	0.821	9.6E-4	0.158
DDS model #3	2081.5	0.00201	—	0.00101	0.154
Experiment RMS	277.3	0.0468	0.0217	0.0057	0.0315
DDS RMS #1	24.1	4.68E-5	—	2.82E-07	1.36E-3
DDS RMS #2	24.07	4.69E-5	3.93E-02	2.84E-7	1.31E-3
DDS RMS #3	24.04	4.70E-5	—	2.85E-07	1.32E-3
Discrepancy #1	394.72	0.0185	—	0.00359	0.0132
Discrepancy #2	395.21	0.0185	0.0042	0.0036	0.018
Discrepancy #3	380.69	0.0185	—	0.00359	0.0132
Error #1 (%)	23.2	-47.9	—	-78.1	9.43
Error #2 (%)	23.24	-47.98	0.518	-79.14	12.85
Error #3 (%)	22.38	-47.91	—	-78.11	9.43



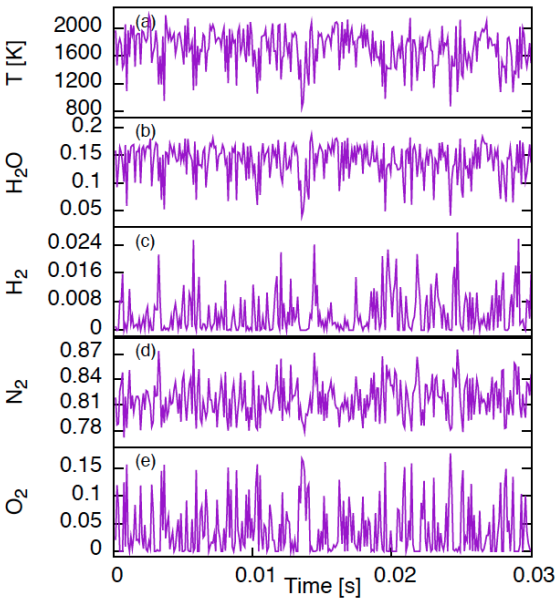
(a) Case 1



(b) Case 2



(c) Case 3



(d) Experimental data

Figure 4.7: Numerical solution and comparison at location: $x/D = 10$, $r = 9$

Table 4.5 provides a quantitative comparison between numerical solutions of the DDS model and the experimental data. Numerical solutions of all three cases of the

DDS model, such as temperature, major species concentrations, and related turbulent flames statistics, are listed in this Table. Figures 4.7 exhibit numerical solutions of the current investigated location at a qualitative sense, and provide comparison of temporal fluctuations between the DDS model in the three cases and the experimental data.

The DDS model over predicts temperature in all three cases, and case 3 has a smallest value for temperature. The least discrepancy in the DDS model for temperature term is 380.69 K, which is larger than the corresponding experimental RMS, 277.3 K; and the least percentage error of temperature of the DDS model in all three cases is larger than 22% (see Table 4.5). The initial condition for temperature is 1970.7 K (see, Table 3.4) which is nearly 270 K more than the experimental time averaged temperature value. The time averaged temperature in the DDS model is closer to its initial condition rather than the corresponding experimental mean value. In addition, the smallest percentage error of O_2 concentration is 47.9%, and for H_2 concentration this value is 78.1%, in all three cases. Both computed time averaged of concentrations of O_2 and H_2 are closer to their initial condition (see, Table 3.4) rather than the experimental averaged values. This kind of result shows the DDS model is probably highly influenced by its initial condition. Though the percentage error of species concentrations, O_2 and H_2 are very big, the discrepancies for all species concentration are still less than the corresponding experimental RMS in all three cases (for details, see Table. 4.5).

Recall that the species concentrations relate to bifurcation parameters are set as preset values in this work. Without appropriate bifurcation parameters that can represent species concentration fluctuations, the DDS model will not exactly predict physical chemical reactions, let alone obtain correct numerical solutions for species concentrations. Therefore, more bifurcation parameters that relate to species concentrations should be investigated to improve the DDS model predictions of species

reactions.

Figures 4.7(a) 4.7(b) 4.7(c) prove that the DDS model can mimic turbulent combustion fluctuation behaviors at location $x/D = 10$, $r = 9$. At this location, the DDS model has demonstrated that it works well in three points in this section. It is seen that temperature in the numerical solutions (Figs. 4.7(a) 4.7(b) 4.7(c)) is similar to that in the experimental data (see Fig. 4.7(d)) in both qualitative and appearance, both them display bilateral oscillation (swing up and down in a centerline). In addition, temperature fluctuation in case 3 (see Fig. 4.7(c)) almost exactly match the experimental data in frequency and amplitude, and this numerical term covers all of turbulent features as shown in Fig. 4.7(d).

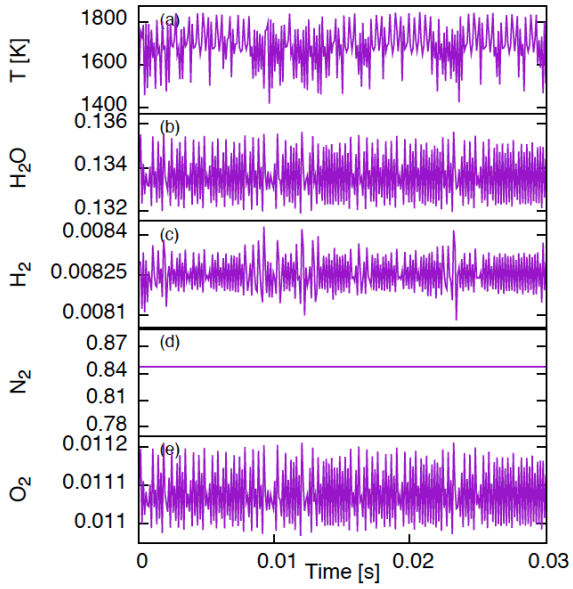
The major difference between the DDS model and the experimental data is that numerical solutions hold a higher frequency oscillation, and higher frequency is a truth for the DDS model results in this work. This discrepancy may be caused by the apparatus for acquiring experiment data lacking sufficient sensitivity and losing some high-pass frequency information. If Fig. 4.7(d) is constructed with a low-pass filtering, the turbulent fluctuation behaviors of numerical solutions may be close to experimental case in both qualitative and quantitive. The second possible reason is that the time scale used in the DDS model does not match that in the experimental data. Another difference between the DDS model and the experimental data is that there are occasional high-amplitude fluctuations in Fig. 4.7(d), however this feature is not repeated in the DDS model. The similar high-amplitude fluctuations issue has been analyzed in the previous work, thus, it is not necessary to restate it again.

4.4.2.4 Location: x/D=20, r=10.5

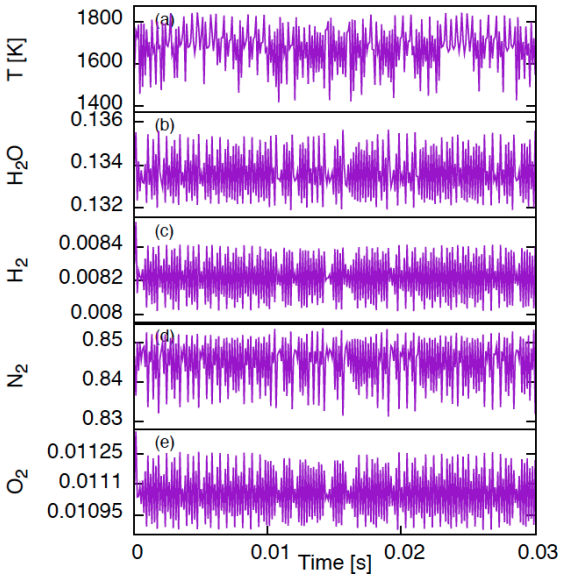
Time-averaged numerical solutions at location $x/D = 20$, $r = 10.5$ are shown in Table 4.6, and time series of dependent variable behaviors are presented in Figs. 4.8. Comparison between numerical solutions and the experimental data is included in Table 4.6 and Figs. 4.8.

Table 4.6: Solution and comparison at location: $x/D = 20$, $r = 10.5$

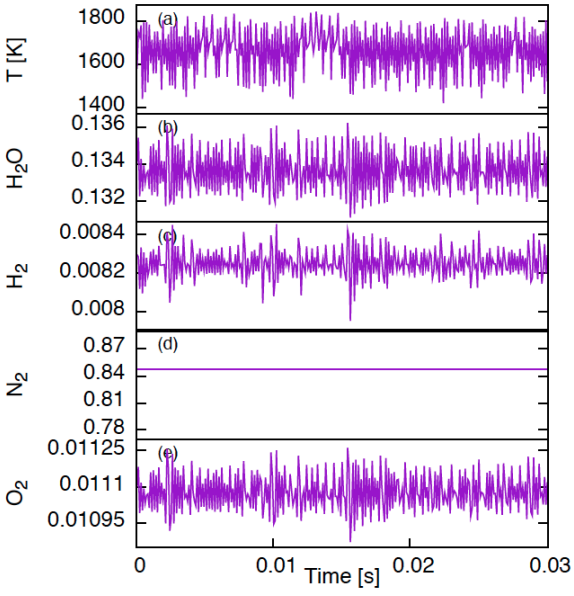
	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
Experiment mean	1639.5	0.0376	0.8198	0.0054	0.1372
DDS model #1	1679.06	0.0108	—	0.0076	0.129
DDS model #2	1680.3	0.0108	0.849	0.00725	0.133
DDS model #3	1673.3	0.0108	—	0.0076	0.129
Experiment RMS	288	0.05	0.022	0.0061	0.033
DDS RMS #1	22.42	2.46E-5	—	2.5E-6	1.08E-3
DDS RMS #2	22.46	2.46E-5	0.0409	2.51E-6	1.08E-3
DDS RMS #3	22.42	2.47E-5	—	2.52E-6	1.08E-3
Discrepancy #1	39.56	0.0267	—	0.0022	0.0085
Discrepancy #2	40.8	0.0268	0.029	0.0018	0.0045
Discrepancy #3	33.85	0.0267	—	0.0022	0.0083
Error #1 (%)	2.41	-71.21	—	40.61	-6.24
Error #2 (%)	2.489	-71.23	3.59	34.19	-3.28
Error #3 (%)	2.06	-71.18	—	40.82	-6.07



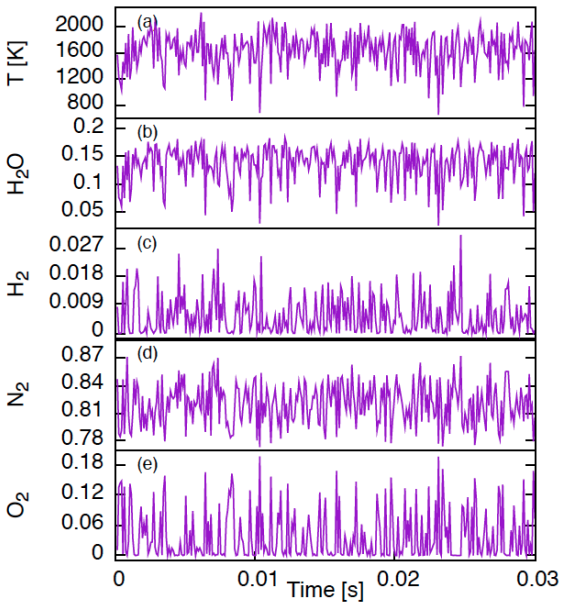
(a) Case 1



(b) Case 2



(c) Case 3



(d) Experiment data

Figure 4.8: Numerical solution and comparison at location: $x/D = 20$, $r = 10.5$

Table 4.6 and Figs. 4.8 show the DDS model in all three case over predict temperature, and case 3 has a smallest value for temperature term. Temperature of numerical

solutions in Table 4.6 show a striking similarity as that in the experimental data, and the corresponding error is less than 2.5% for all three cases. In addition, time series of temperature fluctuation of the numerical solutions in Figs. 4.8 perform very well, and the difference between the numerical cases and the experimental data is very small.

For species concentrations, the absolute percentage errors of O_2 concentration and H_2 concentration are more than 70% and 40%, separately. As a validation testing, such big differences are unacceptable. However, the purpose of this sub-grid scale DDS model is not to exactly repeat the real physics in quantitative, but rather to test capability of the DDS model in simulating turbulent behaviors. In addition, the numerical simulation at a single fixed location is hard to predict behaviors of the same fixed location in flow fields. Thus, it is not wise to expect the numerical solutions can exactly repeat the variables fluctuations as them in the experimental data. The discrepancies of species concentrations O_2, H_2 and H_2O are less than the corresponding experimental RMS in Table 4.6, which means numerical oscillations are within the experimental amplitude. Though, the discrepancy of N_2 concentration in case 2 (0.029) is a bit of big, the percentage error is still very small (less than 4%).

Figures 4.8 show time series of dependent variables of the DDS model, both the numerical results in all three cases and the experimental data exhibit turbulent fluctuations. Time series of temperature, H_2O concentration and H_2 concentration in Fig. 4.8(a) have similar behaviors as these in Fig. 4.8(c), both them match the experimental data well. Time series of O_2 concentration in Fig. 4.8(c) displays more high-amplitude oscillations than that in Fig. 4.8(a), which demonstrates non-isotropic conditions are more suitable than isotropic assumption for the DDS model. Though time series of variables in Fig. 4.8(b), case 2, show turbulent fluctuations, these fluctuations look like periodic oscillations. Fig. 4.8(b) is dissimilar to Fig. 4.8(d), and the possible reason is that the time scale in the case 2 is not comparable to

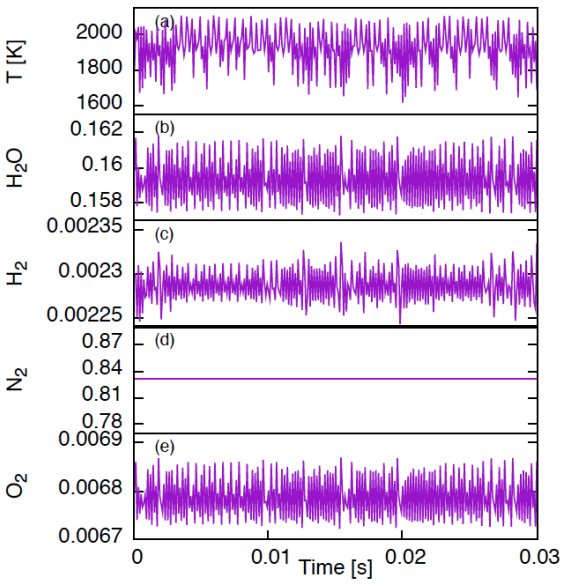
the experimental data, and the inappropriate time scale produces higher frequency fluctuations.

4.4.2.5 Location: x/D=30, r=7.5

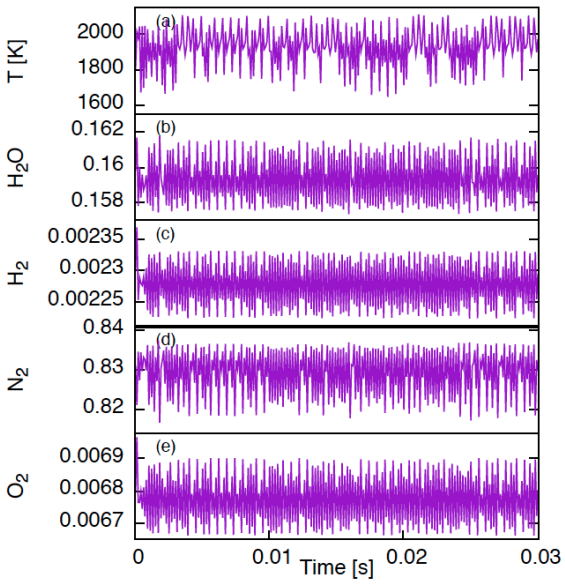
The current investigated point is located at top of flame, where is far away from fuel nozzle exit. This location is chosen from the maximum temperature line at height $x/D = 30$; at this height, the flame boundary radius is $r = 20$. The initial condition in Table 3.4 show species concentrations of O_2 and H_2 is 0.0068 and 0.0023, individually, which illustrate both oxidizer and fuel concentration are relatively low. Time-averaged of the numerical solutions at location $x/D = 3$, $r = 10.5$ are shown in Table 4.7, and time series of dependent variable behaviors are presented in Figs. 4.9. Comparison between the numerical solutions and the experimental data is included in Table 4.7 and Figs. 4.9.

Table 4.7: Solution and comparison at location: $x/D = 30$, $r = 7.5$

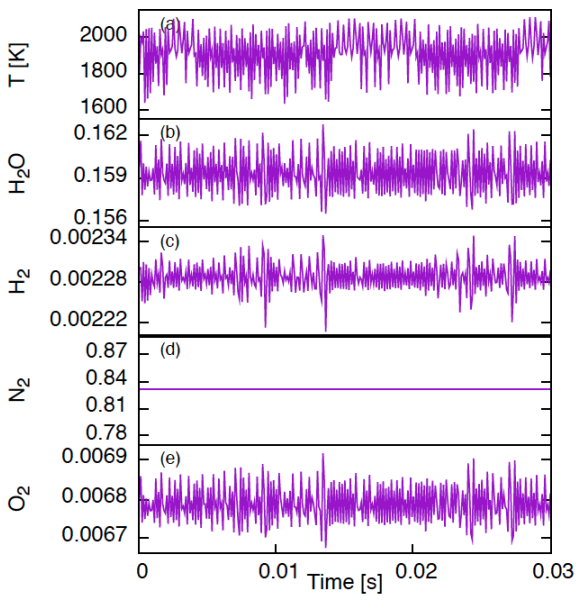
	T [K]	Y_{O_2}	Y_{N_2}	Y_{H_2}	Y_{H_2O}
Experiment mean	1644.7	0.0242	0.8256	0.0054	0.1448
DDS model #1	1922.5	0.0066	—	0.0021	0.153
DDS model #2	1924.88	0.0066	0.833	0.002	0.158
DDS model #3	1915.0	0.0066	—	0.0021	0.154
Experiment RMS	220.1	0.0393	0.0195	0.0057	0.0251
DDS RMS #1	23.67	2.66E-6	—	2.7E-7	1.44E-3
DDS RMS #2	23.71	2.66E-6	4.08E-2	2.59E-7	1.48E-3
DDS RMS #3	23.67	2.67E-6	—	2.72E-7	1.44E-3
Discrepancy #1	277.8	0.0176	—	0.0033	0.0086
Discrepancy #2	280.18	0.0176	0.0075	0.0034	0.01345
Discrepancy #3	270.3	0.0176	—	0.0033	0.0087
Error #1 (%)	16.89	-72.58	—	-61.01	5.97
Error #2 (%)	17.04	-72.63	0.911	-62.88	9.29
Error #3 (%)	16.43	-72.57	—	-61.02	6.04



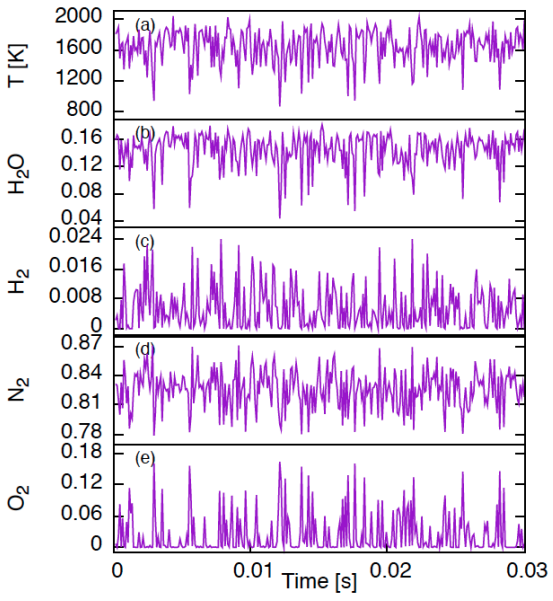
(a) Case 1



(b) Case 2



(c) Case 3



(d) Experiment data

Figure 4.9: Numerical solution and comparison at location: $x/D = 30$, $r = 7.5$

Table 4.7 and Figs. 4.9 show the DDS model obtain a over predicted temperature. The time averaged temperature in the experimental data (seeTable 4.7) is 1644.7 K, which is at least 270 K less than the numerical solutions. Case 3 has the smallest discrepancy for temperature in three cases, and the corresponding temperature is 1915 K .

The initial condition of temperature in the current investigated location is 1807.6 K (see Table 3.4) which is 162.9 K more than the experimental time averaged temperature, and 107.4 K less than the numerical temperature in case 3. These differences demonstrate temperature in the DDS model is hugely influenced by initial condition. In order to improve prediction of temperature, the temperature model in this work should be updated in the further studies. For concentrations of O_2 and H_2, the differences between the experimental time averaged values and the computed time averaged values are too big to be acceptable, and the corresponding percentage error are more than 70% and 61%, separately. In order to get better species concentrations fluctua-

tions, an appropriate temperature model with appropriate species related bifurcation parameters setting should be applied for further investigation.

The computed time averaged values in Table 4.7 are closer to initial conditions in Table 3.4 rather than the corresponding experimental data, which demonstrate species concentrations in the DDS model are also greatly affected by initial conditions. It is difficult for the DDS model to get a numerical time averaged value that is close to the experimental time averaged value if the latter is far away from its initial conditions. The possible reason for this problem is that the DDS model only takes local information into consideration, and it is unable to count global flow fields. But this problem will be solved when the SGS DDS model is applied to LES model.

Figures 4.9 provide a comparison of time series of dependent variables between the DDS model in three cases and the experimental data. Time series of dependent variables exhibit turbulent fluctuations in both the numerical solutions and the experimental data. However, variables fluctuations in Fig. 4.9(b) are close to the time averaged value. Furthermore, solutions of the DDS model in Fig. 4.9(b) do not produce high amplitudes as displayed in other three figures. In terms of time series of variables fluctuations, case 2 does not perform well. It may caused by the N_2 reaction with homogenous and isotropic assumption is not a suitable combination for the SGS DDS model, or the time scales in the DDS model is not in an appropriate region. The temperature turbulent fluctuations in the DDS model (see Figs. 4.9(a) 4.9(b) 4.9(c)) show similar behaviors as them in experimental data (see Fig. 4.9(d)), all of them have two sides oscillations and occasional high amplitude. It looks like that the concentrations of H_2 and O_2 only produce one side oscillation (see Fig. 4.9(d)). The reason for this phenomenon is that the times series of species concentrations should be above zero, and these two species concentrations are just close to zero. If the time averaged lines of these two species concentrations are plotted in Fig. 4.9(d), it will be easy to recognize that these two species concentrations also have two sides oscilla-

tions along the lines. Figure 4.9(c) produces occasional high-amplitude fluctuations in temperature, H_2, O_2 and H_2O concentrations, and case 3 produces very distinct turbulent fluctuations even in low species concentrations (viz., concentrations of H_2 and O_2). The performance of the DDS model in case 1 (see Fig. 4.9(a)), case 2 (see Fig. 4.9(b)) and case 3 (see Fig. 4.9(c)) demonstrates that inhomogeneous and anisotropic conditions are suitable settings (case 3) for the turbulent diffusion flames chemical reactions processes.

Tables and Figures in this section demonstrate that the DDS model can match the time averaged values of temperature and species concentrations with the experimental data in a certain degree, and most of the computed RMS in tables are lower than the corresponding experimental RMS. We note that the physical condition in the database is not the same as them in the current model, and the purpose of the SGS DDS model is to simulate the unresolved small part. Thus, we could not expect the computed results in the DDS model can exactly agree with the experimental data. Overall, the computed results can mimic the diffusion flames combustion processes, and the 3-D DDS model works relatively well. Nevertheless, we will also discuss potential factors that cause discrepancies in the DDS model at section 4.5.

4.5 Discrepancy Analysis

Zeng *et al.* [100] made a simple testing for the DDS model at a single position, and in that paper, the buoyancy term, α_T (see Eq. 2.32c) was set to zero in all three dimensions. For a low flow velocity case, buoyancy effects cannot be ignored; however, the velocity of fuel is 34.8 m/s and the Reynolds number is 10,000, which means buoyancy effect only accounts for a small part in flame velocity fluctuations. Moreover, the buoyancy term was calculated at all three dimensions in two chemical mechanisms with different bifurcation parameter assumptions during programming processes, and there is almost no difference in the numerical solutions whether buoy-

ancy effect was applied or not. Thus, for simplification, the final programming only considers buoyancy effect in the vertical direction, that is z direction in Eq. 2.32c.

In terms of reduced mechanisms for chemical reactions, Zeng *et al.* [100] used a nine step reduced mechanism (including forward and backward reactions), and this mechanism does not contain reactions for NO_x. As we know NO_x does exist in combustion processes, especially at high temperature (viz., temperature is over 2000K), and N_2 is one part of the fuel in H_2/N_2 jet diffusion flame. In this work, a twelve step reduced mechanism with N_2 reactions and isotropic and homogenous assumption is applied in the DDS model. For time-averaged values of variables, there is no big difference between the numerical solution of the reacting N_2 mechanism and the N_2 dilutions mechanism. Thus, NO_x will not an investigated object in further discrepancy analysis. However, both the nine step and twelve step mechanisms are reduced mechanisms, and some important information inevitably lost. In order to consider more species and obtain a more accurate prediction of chemical reactions, a detailed chemical mechanisms may be a good option for the DDS model in further studies.

Recall that in Chapter 2, the DDS model is derived for use as part of a SGS model for LES, and in this context the model would be evaluated only at a single point rather than in global flow fields. The computed solutions of the DDS model are set by initial conditions in Table 3.4 and selected bifurcation parameter values, and they do not consider non-local part information, let alone details in global flow fields. In contrast, although the experimental data are measured at fixed locations, temperature and species concentrations are affected by adjacent positions and global air/fuel flows. Since the adjacent points in the flow fields are unable to affect the numerical solutions in the DDS model, these is no occasional high amplitude in the numerical figures. Furthermore, if the occasional high amplitude in the experimental figures are removed, the experimental time series of variables fluctuations would fairly close to the numerical solutions.

The major difference in comparison is that the numerical solutions holds higher frequency oscillation than that in the experimental data, and the higher frequency is a truth for DDS model results in this work. This discrepancy is probably caused by the time scale of the DDS model is not the same as that of the experimental data. In this work, a model for physical temperature is studied for use in calculating species reaction rates, and a scaling method is employed for the corresponding momentum analysis. The temperature model is made up two parts (see Eq. 2.41), that is, the resolved temperature from the previous time step, and a unresolved part (temperature fluctuation) which is calculated with a scaling initial value for temperature. The initial scaling factor, e_0, is a preset number, and this number plays a vital role on temperature fluctuations. It is probably that the initial scaling factor, e_0, that is used in this work does not reveal real physical settings as them in the experimental conditions, then cause the DDS model tends to over predict temperatures in the diffusion flame. The same scaling issue exists in the momentum scaling implementation, and the scaling initial values for velocities are preset numbers too. Both scaling values for temperature and velocities are not verified by a numerical testing. Therefore, for further investigation, a numerical testing for the scaling factors will be utilized.

The treatment of collision partners is listed in section 2.4, which discussed the way for chaperon efficiencies and pressure dependence of rate coefficients selection. For simplicity, the chaperon efficiencies are set as unity in the numerical computing, however, these efficiencies have their own values that are validated by experiment. For example, chaperon efficiencies of the third body M in Eq. 2.32d are 2.5 for H_2, 12.0 for H_2O, and 1.0 for all other species (see Boivin *et al.* [32]), that is

$$M = (2.5)[H_2] + (16)[H_2O] + (1.0)[OH] + (1.0)[O] + (1.0)[N_2] + (1.0)[N] + (1.0)[NO].$$

In addition, a low-pressure rate coefficient is selected in the pressure dependence of rate coefficients case. Recall that a low-pressure rate coefficient is chosen only when

the concentration of the collision partner is very small. However, in the current physical settings, the collision parter has a large concentration since it includes concentration of N_2. Therefore, a high-pressure rate coefficient should be selected for the pressure dependence of rate coefficients situation.

Zeng *et al.* [100] applied a homogeneous and isotropic assumption in an initial investigation, and the numerical solutions with this condition do not show a good performance in predicting time series of variables fluctuations. In the references, McDonough [102] and [103], this assumption was not employed, and results were qualitatively better, even though the model was only 2D. In the current work, inhomogeneous and anisotropic conditions are used for the DDS model, and the corresponding solutions display very good turbulent fluctuations as did in the experimental data, which demonstrates inhomogeneity and anisotropy is a precondition for turbulent diffusion flames.

Furthermore, there are numerous bifurcation parameters in the DDS model, but only a pair of bifurcation parameters $\beta_{\mathbf{u}}$ and $\gamma_{\mathbf{u}}$ that are related to momentum are studied in this work. Other bifurcation parameters that associated with species i, (e.g, β_{Y_i}, α_{Td_i} etc.,) and temperature (such as β_T) are not analyzed. The bifurcation parameters that are related to species should be analyzed thoroughly rather than setting them with preset values, since every species has its own characteristics and most of them are sensitive to values of bifurcation parameters. To repeat a real physical process, a regime map should be applied for every bifurcation parameter in the DDS model to obtain optimal value, however, this attempt is limited by the current computing capability.

Chapter 5 Conclusions and Future Work

In this thesis, a 3-D discrete dynamical system for finite-rate combustion is derived from governing PDEs, and a nine-step reduced mechanism with N_2 dilution and a twelve-step reduced mechanism with N_2 reaction are utilized to mimic H_2-air turbulent combustion. These two reduced mechanisms are used to check effects of NO_x in the DDS model. The N_2 reaction mechanisms show bad fluctuation behaviors in the numerical solution figures, which means this mechanisms does not improve calculation accuracy for chemical reactions. Thus, though NO_x exists in high temperature (over 2000K) flames, the NO_x concentration fluctuations does not play a major role in the DDS model for the current physical condition. In addition, the discrepancy of reacting N_2 mechanism in the location $x/D = 2.5$, $r = 0$ shows this N_2 reaction mechanism is not suitable in a low temperature condition. However, there are only 10 species involved in the reacting N_2 mechanism, and some potential minor species are ignored in this reduced mechanism.

The sum of species mass fractions in all five cases are close to unity, which demonstrates the DDS model works well in multiple locations for mass conservation in 3D. The numerical solutions of the DDS model in all five locations indicate that the DDS model is capable of mimicking turbulent combustion processes in multiple locations and physical environments. Time series of dependent variables exhibit turbulent fluctuations similar to those of experimental data in a qualitative sense; some variables even display behaviors in accord with the experimental data in a quantitative sense. Temperature phase portraits are employed to analyze variation of ten species con-

centrations with temperature, and it is found that features in these phase portraits exhibit strong similarity to actual chemical reaction characteristics.

The bifurcation parameters in the DDS model are studied with two conditions, that is, an isotropic and homogenous assumption and the anisotropic and inhomogeneous conditions, in two reduced mechanisms for turbulent combustion processes. The numerical solutions illustrate that the isotropic and homogenous assumption is not valid in this non-premixed flame, and the DDS model with the anisotropic and inhomogeneous conditions displays better turbulent behaviors. Actually, the coefficients in the DDS model should be computed dynamically as the calculation progresses instead of setting it a priors.

Turbulence statistics and fluctuation amplitude from the DDS model are qualitatively different from those of the experimental data; however, the computed time averaged temperature and species concentrations from the DDS model are essentially within experimental error at most locations for all three cases considered. The DDS model is evaluated at fixed locations, and it is not influenced by adjacent points in the whole flame field. This may contribute to the DDS model being unable to reproduce intermittent high amplitudes seen in experimental data. In addition, the DDS model is significantly affected by initial conditions, and it did not produce numerical time-averaged temperature and species concentrations that are approximate to the experimental averaged values when the time averaged values in experiment are far away from their initial conditions. The essential differences between the numerical setting and experimental surroundings may in result in significant discrepancies in the comparisons.

The SGS DDS model is built with a scaling method, which is used to fulfill representative interaction with small scales. The fluctuations of variables in local locations are regarded as small-scale values of the whole flame fields in this work. The scaling ratios in the DDS model make the SGS model provide adequate dissipation from the

large grid scales to the small grid scales, when the SGS is implemented in a LES. Calculation of variables in the SGS model depend on high-pass filtering results, and numerical solutions from the DDS model will result from local (in space and time) construction of bifurcation parameters. In terms of energy conserving codes, the physical temperature obtained from the resolved-scale solution rather than a scaled temperature would be used in a SGS model.

In this thesis, a model for physical temperature is studied, and a scaling method is employed for the corresponding momentum analysis for the DDS model. The numerical solutions tends to over predict temperatures in the diffusion flames. The purpose of constructing this DDS model is to provide a SGS simulation for a single location only, and there is no reason to expect the numerical solution of DDS exactly repeat the physical behavior of same location in the whole flow field. In general, we believe the results of this study are sufficiently promising to suggest continued investigation of discrete dynamical systems for finite-rate chemistry as low-order models on subgrid-scale for a large-eddy simulation.

In terms of future work, in order to improve accuracy of numerical simulation for species reactions, more reduced mechanism should be investigated even a detailed mechanisms should be an optional subject for study. In this thesis, only a pair of bifurcation parameters $\beta_{\mathbf{u}}$ and $\gamma_{\mathbf{u}}$ that are related to momentum are investigated. In order to reveal real physical processes, more bifurcation parameters that associated with species i, (e.g., β_{Y_i}, α_{Td_i} etc.,) and temperature (such as β_T) should be studied by plotting their regime maps and considering their effects on, e.g., intermittencies in species concentrations in future studies. A temperature model that can reproduce high-amplitude fluctuations should be investigated to approximate the turbulent fluctuation behaviors seen in the experimental data. In order to remove the higher frequency issue, a correct time scale that is close to experimental data or appropriate low-pass filtering of computed results is required. In further investiga-

tion on locating the optimal scaling of initial conditions for temperature model and momentum analysis, a new bifurcation parameters searching scheme for these scaling initial values will be constructed. In order to make up the shortcomings of the current treatment of collision partners, different multipliers that are valid from experiment for the chaperon efficiencies of the collision parter will be applied, and a high-pressure rate coefficient for the pressure dependence of rate coefficients case will be selected. For the whole flow field, we need to add this SGS model to the resolved part of a LES and thus construct an approximation to the complete solution.

Appendices

A Reduced Chemical Mechanism Fortran Code

```
      PROGRAM  TRBCHM3
      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (imx=50001,ivr=3)
      PARAMETER (MXSPCS=10,MXRCTN=20)
      PARAMETER (nu0=1001,nv0=1001,nw0=2,nvar=3)
      PARAMETER (R0=8.314D-3)
c     ! Universal gas constant [kJ/(g mol K)]

      DIMENSION u(0:imx),v(0:imx),w1(0:imx),q(ivr,0:imx)

      DIMENSION D(MXSPCS),D0(MXSPCS),D00(MXSPCS),DFM(MXRCTN),
     1     DBM(MXRCTN),W(MXSPCS),ALFATY(MXSPCS),BETAY(MXSPCS),
     1     GAMUY(MXSPCS),GAMVY(MXSPCS),GAMWY(MXSPCS),G(MXSPCS),
     2     H(MXSPCS)    !Add 2 terms change

      DIMENSION CF(MXSPCS,MXRCTN),CB(MXSPCS,MXRCTN),
     1     yrslvd(MXSPCS), drs(mxspcs),fvdata(nu0,nv0,nw0,nvar)

      DIMENSION CYIYJ(MXSPCS,MXSPCS),YBAR(MXSPCS),CUY(MXSPCS),
     1        CVY(MXSPCS),CWY(MXSPCS),CTY(MXSPCS),YSQ(MXSPCS),
     2        y(MXSPCS)                    !Add 1 terms change

      DIMENSION RMLNBR(MXSPCS,MXRCTN),PMLNBR(MXSPCS,MXRCTN),
     1    RMLWT(MXSPCS),PREXPF(MXRCTN),ALFAF(MXRCTN),
     2    AEF(MXRCTN),PREXPB(MXRCTN),ALFAB(MXRCTN),AEB(MXRCTN),
     3    FWDRTE(MXRCTN),BWDRTE(MXRCTN),SRMLN(MXRCTN),
     4    SPMLN(MXRCTN),DA(MXRCTN)

      DIMENSION MPITRS(MXRCTN),MPMOD(MXRCTN)

      DATA TL0,PL,RHOL /2095.6D0,1.01D5,0.263D0/
      DATA RMLWT /2.01588D0,31.9988D0,18.01528D0,17.0074D0,
     1         1.00794D0, 15.9994D0,33.d0,28.0134D0,
```

```
     2          14.0067d0,30.0061d0/
      DATA G /0.120D0,1.D0,0.743D0,0.462D0,0.044D0,
     1           0.276D0,0.d0,3*0.D0/
      DATA H /0.114D0,0.128D0,-1.D0,0.706D0,0.440D0,
     1            0.443D0,0.443d0, 0.121D0,2*0.D0/

c      DATA H /58.93D0,65.47D0,-153.38D0,0.706D0,0.440D0,
      1           0.443D0,0.443d0, 62.27D0,2*0.D0/  !checking

      DATA PREXPF /3.52D16,5.06D4,1.17D9,5.75D19,7.08D13,
      1       1.66D13, 2.89D13,4.00D22,1.30D18,1.8D8,1.8D4,
      2       7.1D7,8*0.D0/ !2.0D14/3.52D16,
        !1.5D14/1.7D14,1.00D8/1.17D9
      DATA PREXPB /7.04D13,3.03D4,1.28D10,0.D0,0.D0,2.69D12,
      1      0.D0, 1.03D23,3.04D17,3.8D7,3.8D3,1.7D8,8*0.D0/
      DATA ALFAF /-0.7D0,2.67D0,1.3D0,-1.4D0,0.D0,0.D0,0.D0,
     &               -2.0D0,-1.0,0.D0,1.D0,0.D0,8*0.D0/
      DATA ALFAB /-0.26D0,2.63D0,1.19D0,0.D0,0.D0,0.36D0,0.D0,
     &            -1.75D0,-0.65D0,0.D0,1.D0,0.D0,8*0.D0/
      DATA AEF /7.142D1,2.632D1,1.521D1,0.D0,1.23D0,3.44D0,
      &    -2.08D0,  0.D0,0.D0,3.19D2,3.89D1,3.74D0,8*0.D0/
      DATA AEB /0.60D0,2.023D1,7.825D1,0.D0,0.D0,2.3186D2,0.D0,
     &     4.9614D2,4.3309D2,3.53D0,1.731D2,2.0419D2,8*0.D0/

  200 FORMAT(1X,18(1PE13.6,1X))

      NSTPS = 20000
      DT = 0.0001
      TRBTSCL = 1.D-5
      tl = tl0

      NSPCS = 10 !8
      NFRCTN = 12 !9
      NBRCTN = 12 !9
      NRCTNS = NFRCTN + NBRCTN
      NSTATS = NSTPS/2

c      write(*,*) MXSPCS,NSTATS
      fctr = 0.06d0
      fctr1 = 1.d0 - fctr
      FCTRT = 2.8D0     !2./2.8
      cinit = 0.035d0


ccc Testing Enthalpy
     Do i=1, MXSPCS
         print*,'H(',i,')=', H(i)
```

```
         print*,'G(',i,')=', G(i)
      End Do
      Stop


*  initial values for species (C1100550)
*******************2nd pt:(5,25)************
      drs(1) =0.0139274224D0         !0.02732832d0 H2
      drs(2) =0.0423559498D0         !0.00147168d0 O2
      drs(3) =0.2299430254D0         !0.11232d0    H2O
      drs(4) =1.D-10                 !2.49459d-3   OH
      drs(5) =1.D-10                 !3.75054d-2   H
      drs(6) =1.D-10                 !9.36488d-10  O
      drs(7) =1.D-10                 !3.64748d-13  HO2
      drs(8) =0.7137736025D0         !0.3586d0     N2
      drs(9) =1.D-10                 !9.36488d-10  N
      drs(10)=1.D-10                 !3.64748d-13  NO

      avgmlwt = 0.d0
      do i=1,nspcs
       avgmlwt = avgmlwt + drs(i)*rmlwt(i)
      end do

      sum=0.d0
      do i=1,nspcs
         sum=sum+drs(i)
      end do
ccccc!!!!!      write(*,*) sum     !test

      summfrc = 0.d0
      do i=1,nspcs
        yrslvd(i) = drs(i)*rmlwt(i)/avgmlwt
        summfrc = summfrc + yrslvd(i)
      end do

      write(*,*) avgmlwt,summfrc,yrslvd(8)
      h2init = drs(1)*fctr*rmlwt(1)/avgmlwt    !mass fractions
      o2init = drs(2)*fctr*rmlwt(2)/avgmlwt
      h2oinit = drs(3)*fctr*rmlwt(3)/avgmlwt
      ohinit = drs(4)*fctr*rmlwt(4)/avgmlwt
      hinit = drs(5)*fctr*rmlwt(5)/avgmlwt
      oinit = drs(6)*fctr*rmlwt(6)/avgmlwt
      ho2init = drs(7)*fctr*rmlwt(7)/avgmlwt
      HN2init = drs(8)*fctr*rmlwt(8)/avgmlwt   !add one term
      HNinit = drs(9)*fctr*rmlwt(9)/avgmlwt    !add one term
      HNoinit = drs(10)*fctr*rmlwt(10)/avgmlwt  !add one term
```

```
c      summ=h2init+o2init+h2oinit+ohinit+hinit+oinit+ho2init
c      write(*,*)  'n2=', HN2init ! 'testh2o',h2oinit  !test

       cffctr = 1.d0

       DO J=1,NFRCTN
        DO I=1,NSPCS
         CF(I,J) = 0.D0
         RMLNBR(I,J) = 0.D0
         PMLNBR(I,J) = 0.D0
        END DO
       END DO

       DO J=1,NBRCTN
        DO I=1,NSPCS
         CB(I,J) = 0.D0
        END DO
       END DO

       BETAU = 3.6416d0! 6416d0/7536     !3.6d0
       BETAV = BETAU    !3.6d0
       BETAW = BETAU


       DO I=1,NSPCS
        BETAY(I) = 0.125D0/sqrt(rmlwt(i))
c        !.09  -1   ! .085 < betay < .1275
       END DO

       GAM12=0.2316d0 !0.2756!0.264d0  2688D0
       GAM13=GAM12
       GAM21=GAM12
       GAM23=GAM12
       GAM31=GAM12
       GAM32=GAM12

****Bifurcation parameters for Energy equations******
         BETAT = 1.5d0      !change from 1.5
         GAMUT = -4.2D0     !3.0/4.2
         GAMVT = -3.0D0
         GAMWT = -3.2D0    ! Add one term change

****Bifurcation parameters for chemical reaction equations***
       gamuy(1) =1.2d0     !1.2/0.62
       gamvy(1) = -1.2d0     !1.2/-0.62
       gamwy(1) = 0.62d0    !1.2/0.62  ! Add one term change
       gamuy(2) = 0.62d0    !-1.3/0.62
```

```
       gamvy(2) = -0.62d0    !-1.3/-3.35/-0.62
       gamwy(2) = 0.62d0    ! Add one term change -1.3/0.62
       gamuy(3) = 0.83d0     !0.83/0.65
       gamvy(3) = -0.65d0    !-1.595/-0.65
       gamwy(3) = 0.65d0     !-1.2/0.65 ! Add one term change
       gamuy(4) = 0.65!1.d0       !1.0
       gamvy(4) = -0.65!1.02d0
       gamwy(4) = 0.65! 1.0d0    ! Add one term change
       gamuy(5) = 0.65d0
       gamvy(5) = -.65d0   !-0.65
       gamwy(5) = 0.65d0    ! Add one term change
       gamuy(6) = .65d0
       gamvy(6) = -.65d0    ! -0.35/-0.65
       gamwy(6) = .65d0    ! Add one term change
       gamuy(7) = .65d0
       gamvy(7) = -.65d0    !-0.65
       gamwy(7) = .65d0    ! Add one term change
       gamuy(8) =-0.65d0     !-0.35
       gamvy(8) =0.65d0     !-0.65/0.35
       gamwy(8) =-0.65d0    !Add three terms -0.35
       gamuy(9) =-0.65d0     !-0.35
       gamvy(9) =0.65d0     !-0.65/0.35
       gamwy(9) =-0.65d0    !Add three terms -0.35
       gamuy(10) =-0.65d0     !-0.35
       gamvy(10) =0.65d0     !-0.65/0.35
       gamwy(10) =-0.65d0    !Add three terms -0.35

       ALFAT = 0.1D0     ! ==> No buoyancy effect if set to 0

*** Add terms to 3D
      ALFATY(1) = 1.D0*g(1)*(GAMUT*gamuy(1)+GAMVT*gamvy(1)+
     1             GAMWT*gamwy(1))         ! H2
      ALFATY(2) = 1.D0*g(2)*(GAMUT*gamuy(2)+GAMVT*gamvy(2)+
     1             GAMWT*gamwy(2))         ! O2
      ALFATY(3) = 1.D0*g(3)*(GAMUT*gamuy(3)+GAMVT*gamvy(3)+
     1             GAMWT*gamwy(3))         ! H2O
      ALFATY(4) = 1.D0*g(4)*(GAMUT*gamuy(4)+GAMVT*gamvy(4)+
     1             GAMWT*gamwy(4))         ! OH
      ALFATY(5) = 1.D0*g(5)*(GAMUT*gamuy(5)+GAMVT*gamvy(5)+
     1             GAMWT*gamwy(5))         ! H
      ALFATY(6) = 1.D0*g(6)*(GAMUT*gamuy(6)+GAMVT*gamvy(6)+
     1             GAMWT*gamwy(6))         ! O
      ALFATY(7) = 1.D0*g(7)*(GAMUT*gamuy(7)+GAMVT*gamvy(7)+
     1             GAMWT*gamwy(7))         ! HO2
      ALFATY(8) = 1.D0*g(8)*(GAMUT*gamuy(8)+GAMVT*gamvy(8)+
     1             GAMWT*gamwy(8))         ! hn2--N2
      ALFATY(9) = 1.D0*g(9)*(GAMUT*gamuy(9)+GAMVT*gamvy(9)+
```

```
1             GAMWT*gamwy(9))          ! hn2--N
 ALFATY(10) = 1.D0*g(10)*(GAMUT*gamuy(10)+GAMVT*gamvy(10)+
1             GAMWT*gamwy(10))         ! hn2--NO



 DO J=1,NFRCTN
  IF(J.EQ.1)THEN
    RMLNBR(5,J) = 1.D0
    RMLNBR(2,J) = 1.D0
    PMLNBR(4,J) = 1.D0
     PMLNBR(6,J) = 1.D0
    CF(5,J) = -RMLWT(5)
     CF(2,J) = -RMLWT(2)
     CF(4,J) = RMLWT(4)
     CF(6,J) = RMLWT(6)
    CB(5,J) = -CF(5,J)
     CB(2,J) = -CF(2,J)
     CB(4,J) = -CF(4,J)
     CB(6,J) = -CF(6,J)
  END IF
  IF(J.EQ.2)THEN
    RMLNBR(1,J) = 1.D0
    RMLNBR(6,J) = 1.D0
    PMLNBR(4,J) = 1.D0
    PMLNBR(5,J) = 1.D0
    CF(1,J) = -RMLWT(1)
     CF(6,J) = -RMLWT(6)
    CF(4,J) = RMLWT(4)
    CF(5,J) = RMLWT(5)
    CB(1,J) = -CF(1,J)
     CB(6,J) = -CF(6,J)
     CB(4,J) = -CF(4,J)
     CB(5,J) = -CF(5,J)
  END IF
  IF(J.EQ.3)THEN
     RMLNBR(1,J) = 1.D0
    RMLNBR(4,J) = 1.D0
    PMLNBR(3,J) = 1.D0
    PMLNBR(5,J) = 1.D0
    CF(1,J) = -RMLWT(1)
    CF(4,J) = -RMLWT(4)
    CF(3,J) = RMLWT(3)
    CF(5,J) = RMLWT(5)
    CB(1,J) = -CF(1,J)
     CB(4,J) = -CF(4,J)
     CB(3,J) = -CF(3,J)
```

```
    CB(5,J) = -CF(5,J)
END IF
IF(J.EQ.4)THEN
  RMLNBR(5,J) = 1.D0
  RMLNBR(2,J) = 1.D0
  PMLNBR(7,J) = 1.D0
  CF(5,J) = -RMLWT(5)
  CF(2,J) = -RMLWT(2)
  CF(7,J) = RMLWT(7)
END IF
IF(J.EQ.5)THEN
   RMLNBR(7,J) = 1.D0
  RMLNBR(5,J) = 1.D0
  PMLNBR(4,J) = 2.D0
  CF(7,J) = -RMLWT(7)
  CF(5,J) = -RMLWT(5)
   CF(4,J) = 2.d0*RMLWT(4)
END IF
IF(J.EQ.6)THEN
   RMLNBR(7,J) = 1.D0
  RMLNBR(5,J) = 1.D0
  PMLNBR(1,J) = 1.D0
   PMLNBR(2,J) = 1.D0
  CF(7,J) = -RMLWT(7)
  CF(5,J) = -RMLWT(5)
   CF(1,J) = RMLWT(1)
   CF(2,J) = RMLWT(2)
  CB(7,J) = -CF(7,J)
   CB(5,J) = -CF(5,J)
   CB(1,J) = -CF(1,J)
   CB(2,J) = -CF(2,J)
END IF
IF(J.EQ.7)THEN
   RMLNBR(7,J) = 1.D0
  RMLNBR(4,J) = 1.D0
  PMLNBR(3,J) = 1.D0
  PMLNBR(2,J) = 1.D0
  CF(7,J) = -RMLWT(7)
  CF(4,J) = -RMLWT(4)
   CF(3,J) = RMLWT(3)
  CF(2,J) = RMLWT(2)
END IF
IF(J.EQ.8)THEN
   RMLNBR(5,J) = 1.D0
  RMLNBR(4,J) = 1.D0
  PMLNBR(3,J) = 1.D0
  CF(5,J) = -RMLWT(5)
```

```
      CF(4,J) = -RMLWT(4)
       CF(3,J) = RMLWT(3)
      CB(5,J) = -CF(5,J)
       CB(4,J) = -CF(4,J)
       CB(3,J) = -CF(3,J)
    END IF
    IF(J.EQ.9)THEN
       RMLNBR(5,J) = 2.D0
      PMLNBR(1,J) = 1.D0
      CF(5,J) = -2.d0*RMLWT(5)
      CF(1,J) = RMLWT(1)
      CB(5,J) = -CF(5,J)
       CB(1,J) = -CF(1,J)
    END IF
    IF(J.EQ.10)THEN
       RMLNBR(6,J) = 1.D0
      RMLNBR(8,J) = 1.D0
       RMLNBR(10,J) = 1.D0
      PMLNBR(9,J) = 1.D0
      CF(6,J) = -RMLWT(6)
      CF(8,J) = -RMLWT(8)
      CF(10,J) = RMLWT(10)
      CF(9,J) = RMLWT(9)
      CB(6,J) = -CF(6,J)
      CB(8,J) = -CF(8,J)
      CB(10,J) = -CF(10,J)
      CB(9,J) = -CF(9,J)
    END IF
    IF(J.EQ.11)THEN
       RMLNBR(9,J) = 1.D0
      RMLNBR(2,J) = 1.D0
       PMLNBR(10,J) = 1.D0
      PMLNBR(6,J) = 1.D0
      CF(9,J) = -RMLWT(9)
      CF(2,J) = -RMLWT(2)
      CF(10,J) = RMLWT(10)
      CF(6,J) = RMLWT(6)
      CB(9,J) = -CF(9,J)
      CB(2,J) = -CF(2,J)
      CB(10,J) = -CF(10,J)
      CB(6,J) = -CF(6,J)
    END IF
    IF(J.EQ.12)THEN
       RMLNBR(9,J) = 1.D0
      RMLNBR(4,J) = 1.D0
       PMLNBR(10,J) = 1.D0
      PMLNBR(5,J) = 1.D0
```

```
          CF(9,J) = -RMLWT(9)
          CF(4,J) = -RMLWT(4)
          CF(10,J) = RMLWT(10)
          CF(5,J) = RMLWT(5)
          CB(9,J) = -CF(9,J)
          CB(4,J) = -CF(4,J)
          CB(10,J) = -CF(10,J)
          CB(5,J) = -CF(5,J)
        END IF

        SRMLN(J) = 0.D0
        SPMLN(J) = 0.D0
        DO I=1,NSPCS
         SRMLN(J) = SRMLN(J) + RMLNBR(I,J)
         SPMLN(J) = SPMLN(J) + PMLNBR(I,J)
         cf(i,j) = cf(i,j)*cffctr
        END DO
       END DO

*** i-loop for backward reaction
C       Do i=1,

        a0 = 0.90D0      ! volivity of fuel
        B0 = 0.05D0      ! velocity of air
        e0 = 0.05D0      ! Add one term change
        C0 = 0.05D0       ! c0 = 0.05/0.1

c            a0 = a
c            b0 = b
c            e0 = e
c            c0 = c

       D0(1) = h2init
       D0(2) = o2init
       D0(3) = h2oinit
       D0(4) = ohinit
       D0(5) = hinit
       D0(6) = oinit
       D0(7) = ho2init
       D0(8) = hn2init    !add one term
       D0(9) = hNinit    !add one term
       D0(10) = hNOinit   !add one term

c      Do i=1,8
c         print*,'d0(',i,')=', d0(i)
c      End do
```

```
      TIME = 0.D0
      summwt = 0.d0

      avgmlwt = 0.d0
      do i=1,nspcs
       avgmlwt = avgmlwt + (d0(i)+yrslvd(i))/rmlwt(i)
      end do
      avgmlwt = 1.d0/avgmlwt

      summfrc = 0.d0
      do i=1,nspcs
       y(i) = d0(i) + yrslvd(i)*fctr1
       summfrc = summfrc + y(i)
      end do

      z = (8.d0*y(1)-y(2)+1.d0)/9.d0

ccccccc!!!      write(*,*) avgmlwt,summfrc,z

      WRITE(11,200)TIME,A0,B0,E0,c0,(D0(I),I=1,NSPCS)
      WRITE(8,200)TIME,A0,B0,E0,tl,(y(I),I=1,NSPCS),
     1                        z,avgmlwt,summfrc

      TAUF = TRBTSCL

cccc      print*, ALFAT
*  iterate equations for a, b, c for npts time steps
      write(*,*)'  '
      write(*,*)'Beginning time evolution loop',k
          DO N=1,nstps
             t = (n-1)*dt
             TIME = N*DT
           a0 = a
           b0 = b
           e0 = e
           c0 = c


         DO N=1,NSTPS
            TIME = N*DT

       IF(N.EQ.NSTATS)THEN
         UBAR = A0
         VBAR = B0
         WBAR = E0         !Add one term
         TBAR = C0
         USQ = A0**2
```

```
          VSQ = B0**2
          WSQ = E0**2      !Add one term
          TSQ = C0**2
          CTU = A0*C0
          CTV = B0*C0
          CTW = E0*C0      !Add one term
          CUV = A0*B0
          CUW = A0*E0      !Add one term
          CVW = B0*E0      !Add one term

          DO J=1,NSPCS
           YBAR(J) = D0(J) !!D0(j)/D0(j)
           YSQ(J) = D0(J)**2
           CUY(J) = A0*D0(J)
           CVY(J) = B0*D0(J)
           CWY(J) = E0*D0(J)  !Add one term
           CTY(J) = C0*D0(J)
           DO I=1,NSPCS
            CYIYJ(I,J) = D0(I)*D0(J)
           END DO
          END DO
        END IF

        FRTMX = 0.D0
        BCTMX = 0.D0
        DO J=1,NRCTNS
          IF(J.LE.NFRCTN)THEN
           IF(ALFAF(J).LT.0.D0)THEN
             TALFA = 1.D0/TL**(-ALFAF(J))
           ELSE IF(ALFAF(J).GT.0.D0)THEN
             TALFA = TL**ALFAF(J)
           ELSE
             TALFA = 1.D0
           END IF
           FWDRTE(J) = PREXPF(J)*TALFA*EXP(-AEF(J)/(R0*TL))
c       if (j==5) then
c        co=7.2071d-5
c        ch=7.2071d-5
c        coh=9.9986d-1
c        fwdrte(j)=fwdrte(5)*co*ch/coh   ! the third body M
c       end if
           IF(FWDRTE(J).GT.FRTMX) FRTMX = FWDRTE(J)
           SRMLN1 = 1.D0-SRMLN(J)
           IF(ABS(SRMLN1).LT.1.D-12)THEN
             RHOI = 1.D0
           ELSE
             RHOI = 1.D0/RHOL**ABS(SRMLN1)
```

```
          END IF
          TAUC = RHOI/FWDRTE(J)
          write(*,*)j,tauc
          DA(J) = TAUF/TAUC
          IF(DA(J).GE.1.D0)THEN
            MPMOD(J) = 0.d0
            MPITRS(J) = DA(J) + 1    ! the meaning of MPITRS
          ELSE
            MPMOD(J) = 1.D0/DA(J) + 1
            MPITRS(J) = 1.d0
          END IF
           print*, j,FWDRTE(j),FRTMX
          print*, J,DA(J),MPMOD(J),MPITRS(J)
        ELSE
            NDX = J - NFRCTN
          IF(ALFAB(NDX).LT.0.D0)THEN
            TALFB = 1.D0/TL**(-ALFAB(NDX))
          ELSE IF(ALFAB(NDX).GT.0.D0)THEN
            TALFB = TL**ALFAB(NDX)
          ELSE
            TALFB = 1.D0
          END IF
          BWDRTE(NDX) =PREXPB(NDX)*TALFB*EXP(-AEB(NDX)/(R0*TL))
          IF(BWDRTE(NDX).GT.BCTMX)BCTMX = BWDRTE(NDX)
          SPMLN1 = 1.D0-SPMLN(NDX)
          IF(ABS(SPMLN1).LT.1.D-12)THEN
            RHOI = 1.D0
          ELSE
            RHOI = 1.D0/RHOL**ABS(SPMLN1)
          END IF

*****Attention DA(j) not exist when BWDRTE(NDX)=0.d0
          TAUC = RHOL**(1.D0-SPMLN(NDX))/BWDRTE(NDX)
          DA(J) = TAUF/TAUC
          IF(DA(J).GE.1.D0)THEN
            MPMOD(J) = 0.d0
            MPITRS(J) = DA(J) + 1    ! the meaning of MPITRS
          ELSE
            MPMOD(J) = 1.D0/DA(J) + 1
            MPITRS(J) = 1.D0
          END IF
c          print*, NDX,DA(J),MPMOD(J),MPITRS(J)
        END IF
       END DO
c          stop

       DO J=1,NRCTNS
```

```
        IF(J.LE.NFRCTN)THEN
           FWDRTE(J) = FWDRTE(J)/FRTMX
           MPITRS(J) = FWDRTE(J)*10. + 1
c           print*, J,FWDRTE(J)
        ELSE
           NDX = J - NFRCTN
           BWDRTE(NDX)=BWDRTE(NDX)/BCTMX
c           print*, NDX,BWDRTE(NDX)
        END IF
ccc         FWDRTE(J) = 0.d0
ccc         if(n.eq.1)write(*,*)j,MPITRS(J)
ccc         MPITRS(J) = 1
C        print*, J FWDRTE(J)
        END DO
c            stop

       if(n.eq.nstats)then
         do j=1,nrctns
         IF(J.LE.NFRCTN)THEN
            write(10,201)j,mpmod(j),MPITRS(J),DA(J),FWDRTE(J)
         ELSE
            NDX = J - NFRCTN
         write(10,201)NDX,mpmod(j),MPITRS(J),DA(J),BWDRTE(NDX)
         End if
         end do
  201 format(1x,i1,2(1x,i3),3x,2(1pe10.3,1x))
       end if

*  Fluid flow calculations

        if(n.eq.1)then
          suma = 0.d0
          sumb = 0.d0
          sume = 0.d0        ! Add one term
          do m=1,nstps
          A = A0*BETAU*(1.D0-A0) - GAM12*A0*B0 - GAM13*A0*E0
     1        + ALFAT*C0
          B = B0*BETAV*(1.D0-B0) - GAM21*A0*B0 - GAM23*B0*E0
     1        + ALFAT*C0*0.d0
          E = E0*BETAW*(1.D0-E0) - GAM31*A0*E0 - GAM32*B0*E0
     1        + ALFAT*C0*0.d0        !Add terms
           suma = suma + a
           sumb = sumb + b
           sume = sume + e
           if(m.eq.1)then
             asve = a
             bsve = b
```

```
            esve = e
           end if
           a0 = a
           b0 = b
           e0 = e
          end do

          abr = suma/(nstps-1)
          bbr = sumb/(nstps-1)
          ebr = sume/(nstps-1)
**deal with velocity a0,b0,c0
          a = asve
          b = bsve
          e = esve
c hpc      write(*,*) abr,bbr,ebr

        else
          A = A0*BETAU*(1.D0-A0) - GAM12*A0*B0 - GAM13*A0*E0
     1        + ALFAT*C0
          B = B0*BETAV*(1.D0-B0) - GAM21*A0*B0 - GAM23*B0*E0
     1        + ALFAT*C0*0.d0
          E = E0*BETAW*(1.D0-E0) - GAM31*A0*E0 - GAM32*B0*E0
     1        + ALFAT*C0*0.d0         !Add terms
        end if

        A0 = A - abr
        B0 = B - bbr
        E0 = E - ebr  ! Add one term

*  Chemical kinetics
        w=0.d0
          Do J=1,NSPCS
          IF(J.EQ.1)THEN
         DFM(9) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
     1          + D0(4)/RMLWT(4)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
         DBM(9) = D0(2)/RMLWT(2)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
     1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
         w(1)=CF(1,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
     1       + CB(1,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
     2       + CF(1,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
     2       + CB(1,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
     3       + CF(1,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
     3       + CB(1,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
     4       + CF(1,9)/RMLWT(5)**2*FWDRTE(9)*DFM(9)*D0(5)**2
     4       + CB(1,9)/RMLWT(1)*BWDRTE(9)*DBM(9)*D0(1)
```

```
      D00(1)= -(BETAY(1)+GAMUY(1)*A0+GAMVY(1)*B0+GAMWY(1)*E0)
     1                *D0(1)+ w(1)
      D(1) =  D00(1) + h2init

       END IF
       IF(J.EQ.2)THEN
      DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
     1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
     2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
      w(2) =CF(2,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
     1      + CB(2,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
     2      + CF(2,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
     2      *DFM(4)*D0(2)*D0(5)
     3      + CF(2,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
     3      + CB(2,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
     4      + CF(2,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
     5   + CF(2,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
     5  + CB(2,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
      D00(2)=-(BETAY(2)+GAMUY(2)*A0+GAMVY(2)*B0+GAMWY(2)*E0)
     1              *D0(2)+w(2)
      d(2) = D00(2) + o2init
       END IF
       IF(J.EQ.3)THEN
      DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
     1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
     2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
      DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
     1         + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2         + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
      w(3) =CF(3,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
     1     + CB(3,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
     2     + CF(3,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
     3     + CF(3,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
     3     *DFM(8)*D0(4)*D0(5)
     3     + CB(3,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
      D00(3)= -(BETAY(3)+GAMUY(3)*A0+GAMVY(3)*B0+GAMWY(3)*E0)
     1              *D0(3)+w(3)
      D(3) = D00(3) + h2oinit
       END IF
       IF(J.EQ.4)THEN
      DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
     1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
     2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
      DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
     1         + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2         + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
      w(4) =CF(4,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
```

```
1      + CB(4,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2      + CF(4,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2      + CB(4,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3      + CF(4,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
3      + CB(4,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
4      + CF(4,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
5      + CF(4,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
6      + CF(4,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
6      *DFM(8)*D0(4)*D0(5)
6      + CB(4,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
7   + CF(4,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
7   + CB(4,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
    D00(4)= -(BETAY(4)+GAMUY(4)*A0+GAMVY(4)*B0+GAMWY(4)*E0)
1              *D0(4)+w(4)
    D(4) =  D00(4) + ohinit
      END IF
      IF(J.EQ.5)THEN
     DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
     DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
     DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
     DFM(9) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
1          + D0(4)/RMLWT(4)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
     DBM(9) = D0(2)/RMLWT(2)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
     w(5) =CF(5,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
1       + CB(5,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2       + CF(5,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2       + CB(5,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3       + CF(5,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
3       + CB(5,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
4       + CF(5,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
4       *DFM(4)*D0(2)*D0(5)
5       + CF(5,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
6       + CF(5,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
6       + CB(5,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
7       + CF(5,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
7       *DFM(8)*D0(4)*D0(5)
7       + CB(5,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
8       + CF(5,9)/RMLWT(5)**2*FWDRTE(9)*DFM(9)*D0(5)**2
```

```
8       + CB(5,9)/RMLWT(1)*BWDRTE(9)*DBM(9)*D0(1)
9   + CF(5,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
9   + CB(5,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
   D00(5)= -(BETAY(5)+GAMUY(5)*A0+GAMVY(5)*B0+GAMWY(5)*E0)
1              *D0(5) +w(5)
   D(5) =  D00(5) + hinit
     END IF
     IF(J.EQ.6)THEN
    w(6) =CF(6,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
1      + CB(6,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2      + CF(6,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2      + CB(6,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3    + CF(6,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
3   + CB(6,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
4    + CF(6,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
4   + CB(6,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
   D00(6)= -(BETAY(6)+GAMUY(6)*A0+GAMVY(6)*B0+GAMWY(6)*E0)
1               *D0(6)+w(6)
   D(6) =  D00(6) + oinit
     END IF
     IF(J.EQ.7)THEN
    DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    w(7) = CF(7,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
1        *DFM(4)*D0(2)*D0(5)
2      + CF(7,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
3      + CF(7,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
3      + CB(7,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
4      + CF(7,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
   D00(7)= -(BETAY(7)+GAMUY(7)*A0+GAMVY(7)*B0+GAMWY(7)*E0)
 1              *D0(7) +w(7)
   D(7) =  D00(7) + ho2init
     END IF
     IF(J.EQ.8)THEN
   w(8)=CF(8,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
1   + CB(8,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
   D00(8)= -(BETAY(8)+GAMUY(8)*A0+GAMVY(8)*B0+GAMWY(8)*E0)
 1               *D0(8)+w(8)
   D(8) = D00(8)+ hn2init
     END IF
     IF(J.EQ.9)THEN
w(9)=CF(9,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
1   + CB(9,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
2   + CF(9,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
2   + CB(9,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
3    + CF(9,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
```

```
     3  + CB(9,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
       D00(9)= -(BETAY(9)+GAMUY(9)*A0+GAMVY(9)*B0+GAMWY(9)*E0)
     1              *D0(9)+w(9)
       D(9) = D00(9)+ hninit
         END IF
         IF(J.EQ.10)THEN
     w(10)=CF(10,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
     1 + CB(10,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
     2 + CF(10,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
     2 + CB(10,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
     3  + CF(10,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
     3 + CB(10,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
       D00(10)= -(BETAY(10)+GAMUY(10)*A0+GAMVY(10)*B0+GAMWY(10)
     1                *E0)*D0(10)+w(10)
       D(10) = D00(10)+ hn0init
          END IF
       END DO    ! end J-loop for reactions

       do i=1,nspcs
        d0(i) = d(i)
       end do

*  Calculate average molecular weight

       avgmlwt = 0.d0
       do i=1,nspcs
        avgmlwt = avgmlwt + (d(i)+yrslvd(i))/rmlwt(i)
       end do
       avgmlwt = 1.d0/avgmlwt

*  Thermal energy

       SUMC = 0.d0  ! C0     !maybe the key
       DO I=1,NSPCS
        SUMC = SUMC + C0*ALFATY(I)*D(I)  !*G(I)*D(I)
c        SUMFJ = 0.D0
c        SUMBJ = 0.D0
c        DO J=1,NFRCTN
c         PRODFJ = 1.D0
c         IF(CF(I,J).NE.0.D0)THEN
c           DO L=1,NSPCS
c            PRODFJ = PRODFJ*D(L)**RMLNBR(L,J)
c           END DO
c           SUMFJ = SUMFJ + CF(I,J)*PRODFJ*FWDRTE(J)
c         END IF
c        END DO     ! end j-loop over forward reactions
c        IF(NBRCTN.GT.0)THEN
```

```
c           DO J=1,NBRCTNS
c            PRODBJ = 1.D0
c            IF(CB(I,J).NE.0.D0)THEN
c              DO L=1,NSPCS
c               PRODBJ = PRODBJ*D(L)**PMLNBR(L,J)
c              END DO
c              SUMBJ = SUMBJ + CB(I,J)*PRODBJ*BWDRTE(J)
c            END IF
c           END DO   ! end j-loop over backward reactions
c         END IF
          SUMC = SUMC -H(I)*W(I) ! (SUMFJ-SUMBJ)
        END DO         ! end i-loop over species equations

        C = (SUMC-GAMUT*A0*C0-GAMVT*B0*C0-GAMWT*E0*C0)
     1      /(1.d0+BETAT) + cinit
         if (c.ge.c0) sgnc=1.d0
         if (c.lt.c0) sgnc=-1.d0

!        C0=C !!!!!!!
c        TL = TL0 + fctr*TL0*C
         TL = TL0 + fctrt*TL0*C*sgnc
c     TL = TL0 + fctr*TL0*0.5d0*(C+C0)*FCTRT

        summfrc = 0.d0
        do i=1,nspcs
         if(i.eq.4)then
           fctr2 = 1.06*fctr1
         else
           fctr2 = fctr1
         end if
         y(i) = d(i) + yrslvd(i)*fctr2
         summfrc = summfrc + y(i)
        end do
c          WRITE(*,*) summfrc

        z = (8.d0*y(1)-y(2)+1.d0)/9.d0
ccc        y(3) = 1.d0 - (y(1)+y(2)+y(4)+y(5)+y(6))
        WRITE(11,200)TIME,A0,B0,E0,c,SUMC,(d0(I),I=1,NSPCS)
        WRITE(8,200)TIME,A0,B0,E0,tl,(y(I),I=1,NSPCS),
       1                          z,avgmlwt,summfrc !tl,

        IF(N.GE.NSTATS)THEN
          SUMMWT = SUMMWT + AVGMLWT
          AVGMFRC = AVGMFRC + SUMMFRC
          UBAR = UBAR + A0
          VBAR = VBAR + B0
          WBAR = WBAR + E0   !Add one term
```

```
      TBAR = TBAR + TL
      USQ = USQ + A0**2
      VSQ = VSQ + B0**2
      WSQ = WSQ + E0**2  !Add one term
      TSQ = TSQ + C**2
      CTU = CTU + A0*C
      CTV = CTV + B0*C
      CTW = CTW + E0*C   !Add one term
      CUV = CUV + A0*B0
      CUW = CUW + A0*E0  !Add one term
      CVW = CVW + B0*E0  !Add one term
      DO J=1,NSPCS
       YBAR(J) = YBAR(J) + d(J)
       YSQ(J) = YSQ(J) + d(J)*d(J)
       CUY(J) = CUY(J) + A0*d(J)
       CVY(J) = CVY(J) + B0*d(J)
       CWY(J) = CWY(J) + E0*d(J)  !Add one term
       CTY(J) = CTY(J) + C*d(J)
       DO I=1,NSPCS
        CYIYJ(I,J) = CYIYJ(I,J) + d(I)*d(J)
       END DO
      END DO
     END IF


     A0 = A
     B0 = B
     E0 = E !Add one term
     C0 = C
    End do    !end n-loop

    write(*,*)'Time evolution ended.'
    write(*,*)'  '

    NAVG = NSTPS - NSTATS - 1

    UBAR = UBAR/NAVG
    VBAR = VBAR/NAVG
    WBAR = WBAR/NAVG !Add one term
    TBAR = TBAR/NAVG
    SUMMWT = SUMMWT/NAVG
    AVGMFRC = AVGMFRC/NAVG
    USQ = USQ/NAVG
    VSQ = VSQ/NAVG
    WSQ = WSQ/NAVG   !Add one term
    TSQ = TSQ/NAVG
    CUV = CUV/sqrt(USQ*VSQ)/NAVG
```

```
CUW = CUW/sqrt(USQ*WSQ)/NAVG    !Add one term
CVW = CVW/sqrt(VSQ*WSQ)/NAVG    !Add one term
CTU = CTU/sqrt(USQ*TSQ)/NAVG
CTV = CTV/sqrt(VSQ*TSQ)/NAVG
CTW = CTW/sqrt(WSQ*TSQ)/NAVG    !Add one term
DO J=1,NSPCS
 YBAR(J) = YBAR(J)/NAVG
 YSQ(J) = YSQ(J)/NAVG
 CUY(J) = CUY(J)/sqrt(USQ*YSQ(J))/NAVG
 CVY(J) = CVY(J)/sqrt(VSQ*YSQ(J))/NAVG
 CWY(J) = CWY(J)/sqrt(WSQ*YSQ(J))/NAVG  !Add one term
 CTY(J) = CTY(J)/sqrt(TSQ*YSQ(J))/NAVG
END DO
DO J=1,NSPCS
 DO I=1,NSPCS
  CYIYJ(I,J) = CYIYJ(I,J)/sqrt(YSQ(I)*YSQ(J))/NAVG
 END DO
END DO

WRITE(9,*)'  '
WRITE(9,*)'  '
WRITE(9,*)'  Time Averages'
WRITE(9,*)'  '
WRITE(9,*)' UBAR = ',UBAR
WRITE(9,*)' VBAR = ',VBAR
WRITE(9,*)' WBAR = ',WBAR    !Add one term
WRITE(9,*)' TBAR = ',TBAR
WRITE(9,*)' MLWT = ',SUMMWT
WRITE(9,*)' MFRC = ',AVGMFRC
DO I=1,NSPCS
 WRITE(9,*)' YBAR(',I,') = ',YBAR(I)
END DO
WRITE(9,*)'  '
WRITE(9,*)'  Variances'
WRITE(9,*)' USQ = ',USQ
WRITE(9,*)' VSQ = ',VSQ
WRITE(9,*)' WSQ = ',WSQ    !Add one term
WRITE(9,*)' TSQ = ',TSQ
DO I=1,NSPCS
 WRITE(9,*)' YSQ(',I,') = ',YSQ(I)
END DO
WRITE(9,*)'  '
WRITE(9,*)'  Correlations'
WRITE(9,*)' CUV = ',CUV
WRITE(9,*)' CUW = ',CUW    !Add one term
WRITE(9,*)' CVW = ',CVW    !Add one term
WRITE(9,*)' CTU = ',CTU
```

```
      WRITE(9,*)' CTV = ',CTV
      WRITE(9,*)' CTW = ',CTW     !Add one term
      WRITE(9,*)'  '
      write(9,*)' J ','    ','    CUY(J)    ','    ','     CVY(J)
','
     1           '    ','  CWY(J)    ','    ','  CTY(J)    '
      do j=1,NSPCS
      write(9,'(1x,I2,3(3x,1pe13.6))')j,
     1         CUY(J),CVY(J),CWY(J),CTY(J)
      end do
      WRITE(9,*)'  '
      DO I=1,NSPCS
       DO J=I+1,NSPCS
        WRITE(9,*)' CY(',I,',',J,') = ',CYIYJ(I,J)
       END DO
      END DO

      Stop
      END
```

B Regime Maps Parallel Computing Fortran Code

```
      program pmnspc
      implicit real*8 (a-h,p-z)
      real*4 RTSEC,RTHRS,RTMNS,SECNDS,SIXTY,ZERO
      include 'omp_lib.h'   !define parallel environment
      integer i,j,k,m,n


c Variables nu0, nv0, and nw0 are used for data output in
c FieldView (FV) format. To make a regime map for viewing
c in FV, the regime map will be a 2-D surface. Even though
c the 3-D PMNS equations are calculated, the surface will be of
c dimension nu0 by nv0 (i.e. a cut through the nu0 x nv0 x nw0
c volume through the nw0 plane.

      parameter(imx=50001,ivr=3,namx=1000,npdfmx=1000,nlmx=100)
      parameter(nfftmx=16384,nu0=1001,nv0=1001,nw0=2,nT0=146,
     1                 nvar=3) ! added nw0
      parameter(zero=0.d0,mxspcs=10,ncore=2) !Add new term

      character*7 qcc(ivr),qmx(ivr),qmn(ivr),qbar(ivr),q2(ivr),
     1            qts(ivr),qacl1(ivr),qflt(5*ivr),qskw(5*ivr),
     2            qsf2(ivr),qsf3(ivr),qsf4(ivr),qsf6(ivr)

      dimension u(0:imx),v(0:imx),w(0:imx),d(mxspcs),
     1     gamvy(mxspcs),gamwy(mxspcs),drs(mxspcs),uf(0:imx),
     2     vf(0:imx),wf(0:imx),fn(nfftmx),frq(nfftmx),
     3      pwr(nfftmx),dq(ivr),gamuy(mxspcs)

*  Basic variables and derivatives
      dimension q(ivr,0:imx),dqdt(ivr,imx),dqdx(ivr,imx),
     1          dqdz(ivr,imx),dqdy(ivr,imx)

*  Classical turbulence statistics:  correlations,
*  autocorrelations, PDFs, means, flatness and skewness
      dimension corfns(ivr,0:namx),pdfs(ivr,npdfmx),
     1    qpsq(ivr),qmax(ivr),qmin(ivr),qavg(ivr),tintscl(ivr),
     2    acl1(ivr),f(5*ivr),s(5*ivr) ,qpcor(ivr,ivr)

*  Statistical quantities arising in the Kolmogorov theories:
structure
*  functions of various orders
      dimension s2(5*ivr,nlmx),s3(5*ivr,nlmx),s4(5*ivr,nlmx),
     1          s6(5*ivr,nlmx),sm(5*ivr),rlscl(nlmx)

*  Data storage for FieldView-plottable output.
```

```
      dimension fvdata(nu0,nv0,nw0,nvar),var1(nu0,nv0,nw0),
     1          var2(nu0,nv0,nw0),var3(nu0,nv0,nw0)

  100 format(i5)
c i5 = integer value right justified in 5 columns.
  101 format(d13.6)
c d13.6 = double precision value with 13 columns and
c 2 decimal places.
  200 format(1x,15(a7,1x))
c 1x = Single spacing; 15 repetitions of seven character
c string name
c followed by a line of spacing.
  201 format(10(1pe13.6,1x))
c Ten repetitions of 13 columns and 6+1 decimal places with a
c line of spacing.
  202 format(3i5)

      data qmn / '  umin ','  vmin ',' wmin'/
      data qmx / '  umax ','  vmax ',' wmax'/
      data qbar / '  ubar ','  vbar ',' wbar'/
      data q2 / ' <u^2> ',' <v^2> ',' <w^2>'/
      data qcc / ' <uv> ',' <uw> ',' <vw> '/
      data qts / '  t_u0 ','  t_v0 ',' u_w0'/
      data qacl1 / 'L1<u,u>','L1<v,v>','L1<w,w>'/
      data qflt / '  F(u) ','  F(v) ', ' F(w) ',
     1            'F(dudt)','F(dvdt)', 'F(dwdt)',
     2            'F(dudx)','F(dvdx)', 'F(dwdx)',
     3            'F(dudy)','F(dvdy)', 'F(dwdy)',
     4            'F(dudz)','F(dvdz)', 'F(dwdz)'/
      data qskw / '  S(u) ','  S(v) ', ' S(w) ',
     1            'S(dudt)','S(dvdt)', 'S(dwdt)',
     2            'S(dudx)','S(dvdx)', 'S(dwdx)',
     3            'S(dudy)','S(dvdy)', 'S(dwdy)',
     4            'S(dudz)','S(dvdz)', 'S(dwdz)'/
      data qsf2 / ' s2(u) ',' s2(v) ',' s2(w) '/
      data qsf3 / ' s3(u) ',' s3(v) ',' s3(w) '/
      data qsf4 / ' s4(u) ',' s4(v) ',' s4(w) '/
      data qsf6 / ' s6(u) ',' s6(v) ',' s6(w) '/


      npts = 10000
      nstat = 5000
      nfft2 = 4096
      iu0max = 11!1
      jv0max = 11!1
      kw0max = 1
      ivars = 3
```

```
      u0min = 1.2d0
      u0max = 4.d0
      v0min = -0.6d0
      v0max = 0.5d0
      w0min = 0.01d0
      w0max = 0.8d0

      write(*,*)'  '
      write(*,*)'  '
      write(*,*)'       *** Input data have been loaded ***'
      write(26,*)'input data have been read'
      write(26,*)'npts =',npts
      write(26,*)'nstat =',nstat
      write(26,*)'nfft2 = ',nfft2
      write(26,*)'iu0max = ',iu0max
      write(26,*)'jv0max = ',jv0max
      write(26,*)'kw0max = ',kw0max
      write(26,*)'ivars = ',ivars
      write(26,*)'u0min =',u0min
      write(26,*)'u0max =',u0max
      write(26,*)'v0min =',v0min
      write(26,*)'v0max =',v0max
      write(26,*)'w0min =',w0min
      write(26,*)'w0max =',w0max


      nptavg = npts - nstat
      nptavg1 = nptavg - 1
      nauto = 100
      npdf = 500
      itype = 1     ! itype=0 => velocity statistics, only
                    ! itype>0 => statistics for velocity plus
                    !            first itype scalars


      itype2 = itype + 2
      pi = dacos(-1.d0)
c      print*, pi
      dt = 5.d0*pi/(npts-1)
c      print*, 'dt=', dt
c      stop

      if(iu0max.gt.1)then
        da = (u0max-u0min)/(iu0max-1)
      else
```

```
        u0max = u0min
        da = 0.d0
      end if
      if(jv0max.gt.1)then
        db = (v0max-v0min)/(jv0max-1)
      else
        v0max = v0min
        db = 0.d0
      end if
      if(kw0max.gt.1)then
        dc = (w0max-w0min)/(kw0max-1)
      else
        w0max = w0min
        dc = 0.d0
      end if

      fvdata = 0.d0

      RTSEC = SECNDS(ZERO)


ccc Initial conditios
****Bifurcation parameters for Momentum equations*****
         alpha = 0.0d0

         als = 0.d0
      bls = 0.d0
      cls = 0.d0  !-0.1d0

****Bifurcation parameters for Energy equations******
         BETAT = 1.5d0      !change from 1.5
         GAMUT = -4.2D0     !3.0/4.2
         GAMVT = -3.0D0
         GAMWT = -3.2D0    ! Add one term change

****Bifurcation parameters for chemical reaction equations****
       gamuy=0.d0
       gamvy=0.d0
       gamwy=0.d0
       gamuy(1) =1.2d0     !1.2/0.62
       gamvy(1) = -1.2d0     !1.2/-0.62
       gamwy(1) = 0.62d0    !1.2/0.62  ! Add one term change
       gamuy(2) = 0.62d0    !-1.3/0.62
       gamvy(2) = -0.62d0    !-1.3/-3.35/-0.62
       gamwy(2) = 0.62d0    ! Add one term change -1.3/0.62
       gamuy(3) = 0.83d0     !0.83/0.65
       gamvy(3) = -0.65d0    !-1.595/-0.65
```

```
        gamwy(3) = 0.65d0      !-1.2/0.65 ! change 0.83
        gamuy(4) = 0.65!1.d0        !1.0
        gamvy(4) = -0.65!1.02d0
        gamwy(4) = 0.65! 1.0d0     ! Add one term change
        gamuy(5) = 0.65d0
        gamvy(5) = -.65d0    !-0.65
        gamwy(5) = 0.65d0     ! Add one term change
        gamuy(6) = .65d0
        gamvy(6) = -.65d0     ! -0.35/-0.65
        gamwy(6) = .65d0     ! Add one term change
        gamuy(7) = .65d0
        gamvy(7) = -.65d0     !-0.65
        gamwy(7) = .65d0     ! Add one term change
        gamuy(8) =-0.35d0      !-0.35
        gamvy(8) =0.35d0      !-0.65/0.35
        gamwy(8) =-0.35d0     !Add three terms -0.35


        ALFAT = 0.D0  ! ==> No buoyancy effects if set to zero

*  initial values for species (C1100550)
*******************2nd pt:(5,25)************
      drs=0.d0
      drs(1) =0.0139274224D0         !0.02732832d0
      drs(2) =0.0423559498D0         !0.00147168d0   !0.00147168
      drs(3) =0.2299430254D0         !0.11232d0
      drs(4) =1.D-10                 !2.49459d-3
      drs(5) =1.D-10                 !3.75054d-2
      drs(6) =1.D-10                 !9.36488d-10
      drs(7) =1.D-10                 !3.64748d-13
      drs(8) =0.7137736025D0         !0.3586d0   !Add one term


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc!$OMP PARALLEL DO DEFAULT(PRIVATE), SHARED(fvdata,var1,var2,
ccc!$OMP+ iu0max,u0min,kw0max,w0min,db,da,dc,jv0max,nfft2,itype
ccc!$OMP+ ,npts,nstat,ivars,nptavg,nptavg1,nauto,npdf ,v0min
ccc!$OMP+ itype2,pi,dt)

      if(iu0max*jv0max*kw0max.gt.1)then
        write(*,*)'  '
        write(*,*)'beginning bifurcation parameter value loop'
      end if

*  Begin to parallel computing
c     call system_clock (count1, count_rate, count_max)
      begin= omp_get_wtime ()
```

```
c     define how many cores used in the structure
      call OMP_SET_NUM_THREADS(ncore)


ccc!$OMP PARALLEL Private(k,iu0,jv0,kw0)

!$OMP PARALLEL PRIVATE(k,iu0,jv0,kw0,betau,betav,bataw,
!$OMP+ gamma11,gamma12,
!$OMP+ gamma13,gamma21,gamma23,gamma31,gamma32,itypsln,solntyp)
!$OMP+ SHARED(fvdata,var1,var2,var3,u0min,iu0max,v0min,jv0max,
!$OMP+ kw0max,da,db,dc,nfft2,itype,itype2,pi,dt,npts,nstat,
!$OMP+ nptavg,nptavg1,nauto,npdf,alpha,als,bls,cls,BETAT,GAMUT,
!$OMP+ GAMWT,gamuy,gamvy,gamwy,ALFAT,drs,w0min,GAMVT, ivars)
!$OMP DO
      do kw0=1,kw0max
       do jv0=1,jv0max
        do iu0=1,iu0max


*  Input DDS bifurcation parameters.

*  The below four lines will allow the bifurcation parameters
*  to change values in the case where more than one bifurcation
*  parameter is being investigated (i.e. creating regime maps)

         betau = u0min + (iu0-1)*da
         gamma12 = v0min + (jv0-1)*db
         gamma11 = w0min + (kw0-1)*dc
c         print*, betau,gamma12
c         stop

         var1(iu0,jv0,kw0) = betau
         var2(iu0,jv0,kw0) = gamma12
      var3(iu0,jv0,kw0) = gamma11

c  For studying several different t6ypes of flow regimes, the
c  following lines will need to be commented out. These lines
c  set specific values of the bifurcation parameters for
c  investigating only one set of  bifurcation parameters
c  (i.e generating time series  data for one type of flow).

****Bifurcation parameters for Momentum equations******
c          BETAU = 3.76d0     !3.6d0
          BETAV = BETAU !3.74d0     !3.6d0
          BETAW = BETAU !3.74D0     !Add 1 term change
c         beta2 = 3.87d0
```

```
c          beta1 = 3.90d0      !2.4d0
c          beta3 = 3.10d0      !3.0d0

c          alpha = 0.0d0

c          gamma12 = 0.3d0
ccc        gamma12 = gamma11
          gamma13 = gamma12      !-0.3d0
          gamma21 = gamma12      !-0.3d0
           gamma23 = gamma12      !-0.3d0
          gamma31 = gamma12 !0.3d0! -0.5d0      !-0.3d0
          gamma32 = gamma12 !0.3d0! -0.02d0      !-0.3d0

*  initial values for a, b, c
          do k=1,1
           if(k.eq.1)then
             a = 0.3d0
             b = 0.95d0
             c = 0.2d0
             e = 0.05d0
           else
          a = 0.3d0
          b = 0.95d0
          c = 0.20001d0
             e = 0.05d0
        end if

           itypsln = 0

           q(1,0) = a
           q(2,0) = b
           q(3,0) = c


cccc*  iterate equations for a, b, c for npts time steps
       write(*,*)'  '
       write(*,*)'Beginning time evolution trajectory',k
c           do i=1,npts
c            t = (i-1)*dt
c
c            a0 = a
c            b0 = b
c            c0 = c
c            e0 = e
***** Input chemical reaction *****

       call chem(a,b,c,d,e,BETAU,BETAV,BETAW,gamma12,gamma13,
```

```
     1 gamma21,gamma23,gamma31,gamma32,BETAT,GAMUT,GAMVT,GAMWT,
     2 gamuy, gamvy,gamwy,ALFAT,drs,fvdata,npts,dt,u,v,w,q,t,
     3 iu0,jv0,kw0,itypsln,solntyp)


c         if(itypsln.eq.13)go to 99
          if(itypsln.eq.13) then
              exit
          end if

*              ***** Begin statistical analyses *****
         if(k.eq.1.or.(iu0max.eq.1.and.jv0max.eq.1.and.kw0max.
     1    eq.1))then
        call stat1(q,dqdt,dqdx,dqdy,dqdz,corfns,pdfs,s2,s3,s4,
     1      s6,f,s,sm,qpcor,qpsq,qmax,qmin,qavg,rlscl,tintscl,
     2      acl1,dt,npts,nstat,npdf,nauto,itype,nlsf)
           if(iu0max.gt.1.or.jv0max.gt.1.or.kw0max.gt.1)then
c             fvdata(iu0,jv0, kw0, 2) = qpcor(1,2)
            fvdata(iu0,jv0,kw0,2) = f(9)
            fvdata(iu0,jv0,kw0,3) = s(9)


c             fvdata(iu0,jv0, kw0, 5) = s(3)
c             fvdata(iu0,jv0, kw0, 6) = f(3)
          end if
         end if

c         if(k.eq.1)then
           umin1 = qmin(1)
           umax1 = qmax(1)
           vmin1 = qmin(2)
           vmax1 = qmax(2)
           wmin1 = qmin(3)
        wmax1 = qmax(3)
           avgu1 = qavg(1)
           avgv1 = qavg(2)
        avgw1 = qavg(3)
           avgdiv = 2.d0*betau*(1.d0-avgu1-avgv1)/(avgu1+avgv1)

c           fvdata(iu0,jv0,kw0,7) = avgdiv

c         else
c           umin2 = qmin(1)
c           umax2 = qmax(1)
c           vmin2 = qmin(2)
c           vmax2 = qmax(2)
c           wmin2 = qmin(3)
```

```
c            wmax2 = qmax(3)
c            avgu2 = qavg(1)
c            avgv2 = qavg(2)
c            avgw2 = qavg(3)
c            avgdiv = 2.d0*beta2*(1.d0-avgu2-avgv2)/(avgu+avgv2)
c          end if

*
*  Print selected scalar statistics to FieldView data file
*  and to ftn04 if only one case is being considered
*
c          if(iu0max.eq.1.and.jv0max.eq.1.and.kw0max.eq.1)then
*
*  Output structure functions (orders 2, 3, 4 and 6) for
*  variables and derivatives vs a length increment
*  determined using the Taylor hypothesis; the files are
*  output.sfX ,output.stX, output.sxX, output.syX.

ccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc            write(*,*)'k =',k
            if(k.eq.1)then
              open(11,file='output.sf1',status='unknown')
              open(12,file='output.st1',status='unknown')
              open(13,file='output.sx1',status='unknown')
              open(14,file='output.sy1',status='unknown')
              open(15,file='output.sz1',status='unknown')
              open(16,file='output.pwr1',status='unknown')
              open(17,file='output.vel1',status='unknown')
              open(18,file='output.ac1',status='unknown')
              open(19,file='output.pdf1',status='unknown')
              write(4,*)'beta = ',betau,'   gamma = ',gamma12
              write(4,*)'      '
              write(4,200)(qmn(i),i=1,itype2)
              write(4,200)(qmx(i),i=1,itype2)
              write(4,200)(qbar(i),i=1,itype2)
              write(4,200)(q2(i),i=1,itype2)
              write(4,200)(qcc(i),i=1,itype2)
              write(4,200)(qts(i),i=1,itype2)
              write(4,200)(qacl1(i),i=1,itype2)
              write(4,200)(qflt(i),i=1,5*(itype2))
              write(4,200)(qskw(i),i=1,5*(itype2))
              do i=1,itype2
               write(4,200)qsf2(i),qsf3(i),qsf4(i),qsf6(i)
              end do
              write(4,*)'      '
              write(4,*)'Summary data for trajectory #1: ',
     1                  'avg div =', avgdiv
```

```
        else
          open(11,file='output.sf2',status='unknown')
          open(12,file='output.st2',status='unknown')
          open(13,file='output.sx2',status='unknown')
          open(14,file='output.sy2',status='unknown')
          open(15,file='output.sz2',status='unknown')
          open(16,file='output.pwr2',status='unknown')
          open(17,file='output.vel2',status='unknown')
          open(18,file='output.ac2',status='unknown')
          open(19,file='output.pdf2',status='unknown')
          write(4,*)'Summary data for trajectory #2: ','
     1                avg div =', avgdiv
        end if
        write(4,201)(qmin(i),i=1,itype2)
        write(4,201)(qmax(i),i=1,itype2)
        write(4,201)(qavg(i),i=1,itype2)
        write(4,201)(qpsq(i),i=1,itype2)
        write(4,201)((qpcor(j,i),i=j+1,itype2),j=1,itype2-1)
        write(4,201)(tintscl(i),i=1,itype2)
        write(4,201)(acl1(i),i=1,itype2)
        write(4,201)(f(i),i=1,5*itype2)
        write(4,201)(s(i),i=1,5*itype2)
        do i=1,itype2
         write(4,201)(sm(i+(j-1)*itype2),j=1,4)
        end do
        write(4,*)'  '

        do l=1,nlsf
       write(11,201)rlscl(l),(s2(i,l),s3(i,l),s4(i,l),
     1          s6(i,l ), i=1,itype2)
       write(12,201)rlscl(l),(s2(i+itype2,l),s3(i+itype2,l),
     1           s4(i+itype2,l),s6(i+itype2,l),i=1,itype2)
      write(13,201)rlscl(l),(s2(i+2*itype2,l),
     1          s3(i+2*itype2,l), s4(i+2*itype2,l),
     2          s6(i+2*itype2,l),i=1,itype2)
       write(14,201)rlscl(l),(s2(i+3*itype2,l),
     1          s3(i+3*itype2,l), s4(i+3*itype2,l),
     2          s6(i+3*itype2,l),i=1,itype2)
      write(15,201)rlscl(l),(s2(i+4*itype2,l),
     1          s3(i+4*itype2,l), s4(i+4*itype2,l),
     2          s6(i+4*itype2,l),i=1,itype2)
        end do
        close(11)
        close(12)
        close(13)
        close(14)
      close(15)
```

```
            do i=1,npts
             write(17,201)(i-1)*dt,q(1,i),q(2,i),q(3,i)
            end do
            close(17)

            do j=0,nauto
             write(18,201)j*dt,(corfns(i,j),i=1,itype2)
            end do
            close(18)

            do i=1,itype2
             dq(i) = (qmax(i)-qmin(i))/npdf
            end do
            do j=1,npdf
            write(19,201)(qmin(i)+j*dq(i)-qavg(i),pdfs(i,j),
           1 i=1,itype2)
            end do
            close(19)
c          end if

          icntr = 0
          fn = zero
c          if(k.eq.2)then
c            u = uf
c            umin1 = umin2
c            umax1 = umax2
c          end if
          do i=npts-nfft2+1,npts
           icntr = icntr + 1
           fn(icntr) = w(i)
          end do
           if(abs(umax1-umin1).lt.1.d-12.or.(abs(fn(icntr)
     1      -fn(icntr-1)).lt.1.d-07.and.abs(fn(icntr-1)
     2      -fn(icntr-2)).lt.1.d-07))then
             itypsln = 0
             solntyp = 0.d0
             fvdata(iu0,jv0,kw0,2) = 0.d0
             fvdata(iu0,jv0,kw0,3) = 0.d0
           else
             if(k.eq.1.or.(iu0max.eq.1.and.jv0max.eq.1.
     1        and.kw0max.eq.1)) then
              frq = zero
              pwr = zero
              call psd(fn,frq,pwr,dt,nfft2,nfft)
ccc             fvdata(iu0,jv0,kw0,1) = solntyp
              if(iu0max.eq.1.and.jv0max.eq.1.and.
```

```
       1        kw0max.eq.1)then
                 do i=1,nfft
                  write(16,201)frq(i),pwr(i)
                 end do
                 close(16)
               end if
             end if
             call psdanlyzr(frq,pwr,solntyp,nfft,itypsln)

cc     write(*,*)'solution type from psdanlyzr is', solntyp
             fvdata(iu0,jv0,kw0,1) = solntyp
             if(itypsln.eq.1)then
               fvdata(iu0,jv0,kw0,3) = 1.d0
               fvdata(iu0,jv0,kw0,2) = 0.d0
             end if
ccc             write(*,*)'solntyp =',solntyp
             if(itypsln.gt.13)then
             end if
           end if
c    99  continue
ccc        if(iu0max.eq.1.and.jv0max.eq.1.and.kw0max.eq.1)then
ccc       write(*,*)iu0,jv0,kw0,solntyp
ccc        end if
          end do ! k-loop for trajectories (line ~182)
         end do ![iu0]
        end do ![jv0]
       end do ![kw0]

!$OMP END DO
!$OMP END PARALLEL
       endtime=omp_get_wtime()
       time=endtime-begin
       write(*,*) 'time=',time

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

       RTSEC = SECNDS(RTSEC)
       SIXTY = 60.0
       RTHRS = RTSEC/SIXTY**2
       RTMNS = RTSEC/SIXTY

       write(26,*)
       write(26,'(A,I4,A1,I2.2,A1,F5.2,6X,A,F10.2,A)')
      1  '--> Execution time: ',
      2  int(rthrs), ':', int(rtmns)-int(rthrs)*60,
      3  ':', MOD(RTSEC,SIXTY), '( ', rtsec, ' sec)'
       write(26,*)
```

```
      if(iu0max.gt.1.or.jv0max.gt.1.or.kw0max.gt.1)then
        open(8,file='3Drgmmp.xyz',status='unknown')
        open(9,file='3Drgmmp.qqq',status='unknown')
        write(8,202)iu0max,jv0max,kw0max
        write(8,201)(((var1(i,j,k),i=1,iu0max),j=1,jv0max),
     1       k=1,kw0max),(((var2(i,j,k),i=1,iu0max),j=1,jv0max),
     2       k=1,kw0max),(((var3(i,j,k),i=1,iu0max),j=1,jv0max),
     3       k=1,kw0max)
        write(9,202)iu0max,jv0max,kw0max,ivars
!       write(9,201)((((fvdata(i,j,k,l),i=1,iu0max),j=1,jv0max),
!    1      k=1,kw0max),l=1,ivars)
        do l=1, ivars
         do k=1, kw0max
        do j=1, jv0max
         do i=1, iu0max

        write(9,201) fvdata(i,j,k,l)

        end do
       end do
      end do
        end do
      end if
      write(*,*)'  '
      write(*,*)'                *** Execution completed ***'
      write(*,*)'  '
      write(26,*)'Execution completed'
      close(26)

      stop
      end

*_____________________________________________________________*

      subroutine chem(a,b,e,d,c,BETAU,BETAV,BETAW,gam12,gam13,
     1          gam21, gam23,gam31,gam32,BETAT,GAMUT,GAMVT,
     2          GAMWT,gamuy,gamvy, gamwy,ALFAT,drs,fvdata,nstps,
     3          dt,u,v,w1,q,t,iu0,jv0,kw0,itypsln,solntyp)

      IMPLICIT REAL*8 (A-H,O-Z)

      PARAMETER (imx=50001,ivr=3)
      PARAMETER (MXSPCS=10,MXRCTN=20)
      PARAMETER (nu0=1001,nv0=1001,nw0=2,nvar=3)
      PARAMETER (R0=8.314D-3)   ! Gas constant [kJ/(g mol K)]
```

```
      DIMENSION u(0:imx),v(0:imx),w1(0:imx),q(ivr,0:imx)

      DIMENSION D(MXSPCS),D0(MXSPCS),D00(MXSPCS),DFM(MXRCTN),
     1     DBM(MXRCTN),W(MXSPCS),ALFATY(MXSPCS),BETAY(MXSPCS),
     1     GAMUY(MXSPCS),GAMVY(MXSPCS),GAMWY(MXSPCS),G(MXSPCS),
     2     H(MXSPCS)    !Add 2 terms change

      DIMENSION CF(MXSPCS,MXRCTN),CB(MXSPCS,MXRCTN),
     1      yrslvd(MXSPCS),drs(mxspcs),fvdata(nu0,nv0,nw0,nvar)

      DIMENSION CYIYJ(MXSPCS,MXSPCS),YBAR(MXSPCS),CUY(MXSPCS),
     1        CVY(MXSPCS),CWY(MXSPCS),CTY(MXSPCS),YSQ(MXSPCS),
     2        y(MXSPCS)                    !Add 1 terms change

      DIMENSION RMLNBR(MXSPCS,MXRCTN),PMLNBR(MXSPCS,MXRCTN),
     1    RMLWT(MXSPCS),PREXPF(MXRCTN),ALFAF(MXRCTN),
     2    AEF(MXRCTN),PREXPB(MXRCTN),ALFAB(MXRCTN),AEB(MXRCTN),
     3    FWDRTE(MXRCTN),BWDRTE(MXRCTN),SRMLN(MXRCTN),
     4    SPMLN(MXRCTN),DA(MXRCTN)

      DIMENSION RMLNBR(MXSPCS,MXRCTN),PMLNBR(MXSPCS,MXRCTN),
     1   RMLWT(MXSPCS),PREXPF(MXRCTN),ALFAF(MXRCTN),
     2   AEF(MXRCTN),PREXPB(MXRCTN),ALFAB(MXRCTN),AEB(MXRCTN),
     3   FWDRTE(MXRCTN),BWDRTE(MXRCTN),SRMLN(MXRCTN),
     4   SPMLN(MXRCTN),DA(MXRCTN)

      DIMENSION MPITRS(MXRCTN),MPMOD(MXRCTN)

      DATA TL0,PL,RHOL /2095.6D0,1.01D5,0.263D0/
      DATA RMLWT /2.01588D0,31.9988D0,18.01528D0,17.0074D0,
     1         1.00794D0, 15.9994D0,33.d0,28.0134D0,
     2         14.0067d0,30.0061d0/
      DATA G /0.120D0,1.D0,0.743D0,0.462D0,0.044D0,
     1           0.276D0,0.d0,3*0.D0/
      DATA H /0.114D0,0.128D0,-1.D0,0.706D0,0.440D0,
     1            0.443D0,0.443d0, 0.121D0,2*0.D0/

c      DATA H /58.93D0,65.47D0,-153.38D0,0.706D0,0.440D0,
     1          0.443D0,0.443d0, 62.27D0,2*0.D0/  !checking

      DATA PREXPF /3.52D16,5.06D4,1.17D9,5.75D19,7.08D13,
     1       1.66D13, 2.89D13,4.00D22,1.30D18,1.8D8,1.8D4,
     2       7.1D7,8*0.D0/ !2.0D14/3.52D16,
       !1.5D14/1.7D14,1.00D8/1.17D9
      DATA PREXPB /7.04D13,3.03D4,1.28D10,0.D0,0.D0,2.69D12,
     1      0.D0, 1.03D23,3.04D17,3.8D7,3.8D3,1.7D8,8*0.D0/
      DATA ALFAF /-0.7D0,2.67D0,1.3D0,-1.4D0,0.D0,0.D0,0.D0,
```

```
     &                -2.0D0,-1.0,0.D0,1.D0,0.D0,8*0.D0/
      DATA ALFAB /-0.26D0,2.63D0,1.19D0,0.D0,0.D0,0.36D0,0.D0,
     &            -1.75D0,-0.65D0,0.D0,1.D0,0.D0,8*0.D0/
      DATA AEF /7.142D1,2.632D1,1.521D1,0.D0,1.23D0,3.44D0,
      &    -2.08D0,  0.D0,0.D0,3.19D2,3.89D1,3.74D0,8*0.D0/
      DATA AEB /0.60D0,2.023D1,7.825D1,0.D0,0.D0,2.3186D2,0.D0,
     &     4.9614D2,4.3309D2,3.53D0,1.731D2,2.0419D2,8*0.D0/

  200 FORMAT(1X,16(1PE13.6,1X))

c      NSTPS = 20000
c      DT = 0.0001
      TRBTSCL = 1.D-5
      tl = tl0

      NSPCS = 8
      NFRCTN = 9
      NBRCTN = 9
      NRCTNS = NFRCTN + NBRCTN
      NSTATS = NSTPS/2

c      write(*,*) MXSPCS,NSTATS
      fctr = 0.06d0
      fctr1 = 1.d0 - fctr
      FCTRT = 2.8D0     !2./2.8
      cinit = 0.035d0


******************2nd pt:(5,25)************
c      drs(1) = 0.02732832d0
c      drs(2) = 0.00147168d0   !0.00147168
c      drs(3) = 0.11232d0
c      drs(4) = 2.49459d-3
c      drs(5) = 3.75054d-2
c      drs(6) = 9.36488d-10
c      drs(7) = 3.64748d-13
c      drs(8) = 0.3586d0   !Add one term

      avgmlwt = 0.d0
      do i=1,nspcs
       avgmlwt = avgmlwt + drs(i)*rmlwt(i)
      end do

      sum=0.d0
      do i=1,nspcs
         sum=sum+drs(i)
      end do
```

```
ccccc!!!!!      write(*,*) sum     !test

      summfrc = 0.d0
      do i=1,nspcs
       yrslvd(i) = drs(i)*rmlwt(i)/avgmlwt
       summfrc = summfrc + yrslvd(i)
      end do

c hpc     write(*,*) avgmlwt,summfrc,yrslvd(8)

      h2init = drs(1)*fctr*rmlwt(1)/avgmlwt    !mass fractions
      o2init = drs(2)*fctr*rmlwt(2)/avgmlwt
      h2oinit = drs(3)*fctr*rmlwt(3)/avgmlwt
      ohinit = drs(4)*fctr*rmlwt(4)/avgmlwt
      hinit = drs(5)*fctr*rmlwt(5)/avgmlwt
      oinit = drs(6)*fctr*rmlwt(6)/avgmlwt
      ho2init = drs(7)*fctr*rmlwt(7)/avgmlwt
      HN2init = drs(8)*fctr*rmlwt(8)/avgmlwt      !add one term

c      summ=h2init+o2init+h2oinit+ohinit+hinit+oinit+ho2init
c      write(*,*)  'n2=', HN2init ! 'testh2o',h2oinit  !test

      cffctr = 1.d0

      DO J=1,NFRCTN
       DO I=1,NSPCS
        CF(I,J) = 0.D0
        RMLNBR(I,J) = 0.D0
        PMLNBR(I,J) = 0.D0
       END DO
      END DO

      DO J=1,NBRCTN
       DO I=1,NSPCS
        CB(I,J) = 0.D0
       END DO
      END DO


      DO I=1,NSPCS
       BETAY(I) = 0.125D0/sqrt(rmlwt(i)) ! .085 < betay < .1275
      END DO


c      ALFAT = 0.D0      ! ==> No buoyancy effects if set to 0

*** Add terms to 3D
```

```
 ALFATY(1) = 1.D0*g(1)*(GAMUT*gamuy(1)+GAMVT*gamvy(1)+
1             GAMWT*gamwy(1))         ! H2
 ALFATY(2) = 1.D0*g(2)*(GAMUT*gamuy(2)+GAMVT*gamvy(2)+
1             GAMWT*gamwy(2))         ! O2
 ALFATY(3) = 1.D0*g(3)*(GAMUT*gamuy(3)+GAMVT*gamvy(3)+
1             GAMWT*gamwy(3))         ! H2O
 ALFATY(4) = 1.D0*g(4)*(GAMUT*gamuy(4)+GAMVT*gamvy(4)+
1             GAMWT*gamwy(4))         ! OH
 ALFATY(5) = 1.D0*g(5)*(GAMUT*gamuy(5)+GAMVT*gamvy(5)+
1             GAMWT*gamwy(5))         ! H
 ALFATY(6) = 1.D0*g(6)*(GAMUT*gamuy(6)+GAMVT*gamvy(6)+
1             GAMWT*gamwy(6))         ! O
 ALFATY(7) = 1.D0*g(7)*(GAMUT*gamuy(7)+GAMVT*gamvy(7)+
1             GAMWT*gamwy(7))         ! HO2
 ALFATY(8) = 1.D0*g(8)*(GAMUT*gamuy(8)+GAMVT*gamvy(8)+
1             GAMWT*gamwy(8))         ! hn2--N2

 DO J=1,NFRCTN
  IF(J.EQ.1)THEN
    RMLNBR(5,J) = 1.D0
    RMLNBR(2,J) = 1.D0
    PMLNBR(4,J) = 1.D0
 PMLNBR(6,J) = 1.D0
    CF(5,J) = -RMLWT(5)
     CF(2,J) = -RMLWT(2)
     CF(4,J) = RMLWT(4)
     CF(6,J) = RMLWT(6)
    CB(5,J) = -CF(5,J)
     CB(2,J) = -CF(2,J)
     CB(4,J) = -CF(4,J)
     CB(6,J) = -CF(6,J)
  END IF
  IF(J.EQ.2)THEN
    RMLNBR(1,J) = 1.D0
    RMLNBR(6,J) = 1.D0
    PMLNBR(4,J) = 1.D0
    PMLNBR(5,J) = 1.D0
    CF(1,J) = -RMLWT(1)
     CF(6,J) = -RMLWT(6)
    CF(4,J) = RMLWT(4)
    CF(5,J) = RMLWT(5)
    CB(1,J) = -CF(1,J)
     CB(6,J) = -CF(6,J)
     CB(4,J) = -CF(4,J)
     CB(5,J) = -CF(5,J)
  END IF
  IF(J.EQ.3)THEN
```

```
    RMLNBR(1,J) = 1.D0
   RMLNBR(4,J) = 1.D0
   PMLNBR(3,J) = 1.D0
   PMLNBR(5,J) = 1.D0
   CF(1,J) = -RMLWT(1)
   CF(4,J) = -RMLWT(4)
   CF(3,J) = RMLWT(3)
   CF(5,J) = RMLWT(5)
   CB(1,J) = -CF(1,J)
    CB(4,J) = -CF(4,J)
    CB(3,J) = -CF(3,J)
    CB(5,J) = -CF(5,J)
END IF
IF(J.EQ.4)THEN
   RMLNBR(5,J) = 1.D0
   RMLNBR(2,J) = 1.D0
   PMLNBR(7,J) = 1.D0
   CF(5,J) = -RMLWT(5)
   CF(2,J) = -RMLWT(2)
   CF(7,J) = RMLWT(7)
END IF
IF(J.EQ.5)THEN
    RMLNBR(7,J) = 1.D0
   RMLNBR(5,J) = 1.D0
   PMLNBR(4,J) = 2.D0
   CF(7,J) = -RMLWT(7)
   CF(5,J) = -RMLWT(5)
    CF(4,J) = 2.d0*RMLWT(4)
END IF
IF(J.EQ.6)THEN
    RMLNBR(7,J) = 1.D0
   RMLNBR(5,J) = 1.D0
   PMLNBR(1,J) = 1.D0
    PMLNBR(2,J) = 1.D0
   CF(7,J) = -RMLWT(7)
   CF(5,J) = -RMLWT(5)
    CF(1,J) = RMLWT(1)
    CF(2,J) = RMLWT(2)
   CB(7,J) = -CF(7,J)
    CB(5,J) = -CF(5,J)
    CB(1,J) = -CF(1,J)
    CB(2,J) = -CF(2,J)
END IF
IF(J.EQ.7)THEN
    RMLNBR(7,J) = 1.D0
   RMLNBR(4,J) = 1.D0
   PMLNBR(3,J) = 1.D0
```

```
          PMLNBR(2,J) = 1.D0
          CF(7,J) = -RMLWT(7)
          CF(4,J) = -RMLWT(4)
           CF(3,J) = RMLWT(3)
          CF(2,J) = RMLWT(2)
        END IF
        IF(J.EQ.8)THEN
           RMLNBR(5,J) = 1.D0
          RMLNBR(4,J) = 1.D0
          PMLNBR(3,J) = 1.D0
          CF(5,J) = -RMLWT(5)
          CF(4,J) = -RMLWT(4)
           CF(3,J) = RMLWT(3)
          CB(5,J) = -CF(5,J)
           CB(4,J) = -CF(4,J)
           CB(3,J) = -CF(3,J)
        END IF
        IF(J.EQ.9)THEN
           RMLNBR(5,J) = 2.D0
          PMLNBR(1,J) = 1.D0
          CF(5,J) = -2.d0*RMLWT(5)
          CF(1,J) = RMLWT(1)
          CB(5,J) = -CF(5,J)
           CB(1,J) = -CF(1,J)
        END IF

        SRMLN(J) = 0.D0
        SPMLN(J) = 0.D0
        DO I=1,NSPCS
         SRMLN(J) = SRMLN(J) + RMLNBR(I,J)
         SPMLN(J) = SPMLN(J) + PMLNBR(I,J)
         cf(i,j) = cf(i,j)*cffctr
        END DO
       END DO

*** i-loop for backward reaction
C       Do i=1,

c      a0 = 0.83
c      B0 = 0.07
c      e0 = 0.01     !Add one term change
c      C0 = 0.05D0      ! c0 = 0.05/0.1

           a0 = a
           b0 = b
           e0 = e
           c0 = c
```

```
      D0(1) = h2init
      D0(2) = o2init
      D0(3) = h2oinit
      D0(4) = ohinit
      D0(5) = hinit
      D0(6) = oinit
      D0(7) = ho2init
      D0(8) = hn2init     !add one term

c      Do i=1,8
c         print*,'d0(',i,')=', d0(i)
c      End do

      TIME = 0.D0
      summwt = 0.d0

      avgmlwt = 0.d0
      do i=1,nspcs
       avgmlwt = avgmlwt + (d0(i)+yrslvd(i))/rmlwt(i)
      end do
      avgmlwt = 1.d0/avgmlwt

      summfrc = 0.d0
      do i=1,nspcs
       y(i) = d0(i) + yrslvd(i)*fctr1
       summfrc = summfrc + y(i)
      end do

      z = (8.d0*y(1)-y(2)+1.d0)/9.d0

ccccccc!!!      write(*,*) avgmlwt,summfrc,z

      WRITE(11,200)TIME,A0,B0,E0,c0,(D0(I),I=1,NSPCS)
      WRITE(8,200)TIME,A0,B0,E0,tl,(y(I),I=1,NSPCS),
     1          z,avgmlwt,summfrc

      TAUF = TRBTSCL


cccc      print*, ALFAT
*  iterate equations for a, b, c for npts time steps
c    write(*,*)'  '
c    write(*,*)'Beginning time evolution loop for trajectory',k
          do n=1,nstps
           t = (n-1)*dt
           TIME = N*DT
```

```
c            a0 = a
c            b0 = b
c            e0 = e
c            c0 = c

c           print*, 'NSTPS=', nstps,'Time=', TIME
c           stop
c          stop
c          DO N=1,NSTPS
c          t  = (n-1)*dt

        IF(N.EQ.NSTATS)THEN
          UBAR = A0
          VBAR = B0
          WBAR = E0         !Add one term
          TBAR = C0
          USQ = A0**2
          VSQ = B0**2
          WSQ = E0**2       !Add one term
          TSQ = C0**2
          CTU = A0*C0
          CTV = B0*C0
          CTW = E0*C0       !Add one term
          CUV = A0*B0
          CUW = A0*E0       !Add one term
          CVW = B0*E0       !Add one term

          DO J=1,NSPCS
           YBAR(J) = D0(J) !!D0(j)/D0(j)
           YSQ(J) = D0(J)**2
           CUY(J) = A0*D0(J)
           CVY(J) = B0*D0(J)
           CWY(J) = E0*D0(J)  !Add one term
           CTY(J) = C0*D0(J)
           DO I=1,NSPCS
            CYIYJ(I,J) = D0(I)*D0(J)
           END DO
          END DO
        END IF

        FRTMX = 0.D0
        BCTMX = 0.D0
        DO J=1,NRCTNS
          IF(J.LE.NFRCTN)THEN
           IF(ALFAF(J).LT.0.D0)THEN
             TALFA = 1.D0/TL**(-ALFAF(J))
           ELSE IF(ALFAF(J).GT.0.D0)THEN
```

```
            TALFA = TL**ALFAF(J)
          ELSE
            TALFA = 1.D0
          END IF
          FWDRTE(J) = PREXPF(J)*TALFA*EXP(-AEF(J)/(R0*TL))
c       if (j==5) then
c        co=7.2071d-5
c        ch=7.2071d-5
c        coh=9.9986d-1
c    fwdrte(j)=fwdrte(5)*co*ch/coh   ! the third body M
c       end if
          IF(FWDRTE(J).GT.FRTMX) FRTMX = FWDRTE(J)
          SRMLN1 = 1.D0-SRMLN(J)
          IF(ABS(SRMLN1).LT.1.D-12)THEN
            RHOI = 1.D0
          ELSE
            RHOI = 1.D0/RHOL**ABS(SRMLN1)
          END IF
          TAUC = RHOI/FWDRTE(J)
ccc          write(*,*)j,tauc
          DA(J) = TAUF/TAUC
          IF(DA(J).GE.1.D0)THEN
            MPMOD(J) = 0.d0
            MPITRS(J) = DA(J) + 1  ! MPITRS
          ELSE
            MPMOD(J) = 1.D0/DA(J) + 1
            MPITRS(J) = 1.d0
          END IF
C           print*, j,FWDRTE(j),FRTMX
c           print*, J,DA(J),MPMOD(J),MPITRS(J)
        ELSE
            NDX = J - NFRCTN
          IF(ALFAB(NDX).LT.0.D0)THEN
            TALFB = 1.D0/TL**(-ALFAB(NDX))
          ELSE IF(ALFAB(NDX).GT.0.D0)THEN
            TALFB = TL**ALFAB(NDX)
          ELSE
            TALFB = 1.D0
          END IF
          BWDRTE(NDX) =PREXPB(NDX)*TALFB*EXP(-AEB(NDX)/(R0*TL))
          IF(BWDRTE(NDX).GT.BCTMX)BCTMX = BWDRTE(NDX)
          SPMLN1 = 1.D0-SPMLN(NDX)
          IF(ABS(SPMLN1).LT.1.D-12)THEN
            RHOI = 1.D0
          ELSE
            RHOI = 1.D0/RHOL**ABS(SPMLN1)
          END IF
```

```
*****Attention DA(j) not exist when BWDRTE(NDX)=0.d0
          TAUC = RHOL**(1.D0-SPMLN(NDX))/BWDRTE(NDX)
          DA(J) = TAUF/TAUC
          IF(DA(J).GE.1.D0)THEN
            MPMOD(J) = 0.d0
            MPITRS(J) = DA(J) + 1  !  MPITRS
          ELSE
            MPMOD(J) = 1.D0/DA(J) + 1
            MPITRS(J) = 1.D0
          END IF
c          print*, NDX,DA(J),MPMOD(J),MPITRS(J)
        END IF
       END DO
c          stop

       DO J=1,NRCTNS
       IF(J.LE.NFRCTN)THEN
          FWDRTE(J) = FWDRTE(J)/FRTMX
          MPITRS(J) = FWDRTE(J)*10. + 1
c          print*, J,FWDRTE(J)
       ELSE
          NDX = J - NFRCTN
          BWDRTE(NDX)=BWDRTE(NDX)/BCTMX
c          print*, NDX,BWDRTE(NDX)
       END IF
ccc         FWDRTE(J) = 0.d0
ccc         if(n.eq.1)write(*,*)j,MPITRS(J)
ccc         MPITRS(J) = 1
C       print*, J FWDRTE(J)
       END DO
c           stop

      if(n.eq.nstats)then
        do j=1,nrctns
        IF(J.LE.NFRCTN)THEN
           write(10,201)j,mpmod(j),MPITRS(J),DA(J),FWDRTE(J)
        ELSE
          NDX = J - NFRCTN
          write(10,201)NDX,mpmod(j),MPITRS(J),DA(J),BWDRTE(NDX)
        End if
        end do
  201 format(1x,i1,2(1x,i3),3x,2(1pe10.3,1x))
      end if

*  Fluid flow calculations
```

```
        if(n.eq.1)then
          suma = 0.d0
          sumb = 0.d0
          sume = 0.d0        ! Add one term
          do m=1,nstps
          A = A0*BETAU*(1.D0-A0) - GAM12*A0*B0 - GAM13*A0*E0
     1        + ALFAT*C0
          B = B0*BETAV*(1.D0-B0) - GAM21*A0*B0 - GAM23*B0*E0
     1        + ALFAT*C0
          E = E0*BETAW*(1.D0-E0) - GAM31*A0*E0 - GAM32*B0*E0
     1        + ALFAT*C0         !Add terms
           suma = suma + a
           sumb = sumb + b
           sume = sume + e
           if(m.eq.1)then
             asve = a
             bsve = b
             esve = e
           end if
           a0 = a
           b0 = b
           e0 = e
          end do

          abr = suma/(nstps-1)
          bbr = sumb/(nstps-1)
          ebr = sume/(nstps-1)
**deal with velocity a0,b0,c0
          a = asve
          b = bsve
          e = esve
c hpc      write(*,*) abr,bbr,ebr

        else
          A = A0*BETAU*(1.D0-A0) - GAM12*A0*B0 - GAM13*A0*E0
     1        + ALFAT*C0
          B = B0*BETAV*(1.D0-B0) - GAM21*A0*B0 - GAM23*B0*E0
     1        + ALFAT*C0
          E = E0*BETAW*(1.D0-E0) - GAM31*A0*E0 - GAM32*B0*E0
     1        + ALFAT*C0         !Add terms
        end if

        A0 = A - abr
        B0 = B - bbr
        E0 = E - ebr  ! Add one term

*  Chemical kinetics
```

```
      w=0.d0
        Do J=1,NSPCS
        IF(J.EQ.1)THEN
       DFM(9) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
     1         + D0(4)/RMLWT(4)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2         + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
       DBM(9) = D0(2)/RMLWT(2)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
     1         + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2         + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
       w(1)=CF(1,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
     1      + CB(1,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
     2      + CF(1,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
     2      + CB(1,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
     3      + CF(1,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
     3      + CB(1,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
     4      + CF(1,9)/RMLWT(5)**2*FWDRTE(9)*DFM(9)*D0(5)**2
     4      + CB(1,9)/RMLWT(1)*BWDRTE(9)*DBM(9)*D0(1)
       D00(1)= -(BETAY(1)+GAMUY(1)*A0+GAMVY(1)*B0+GAMWY(1)*E0)
     1               *D0(1)+ w(1)
       D(1) =  D00(1) + h2init

        END IF
        IF(J.EQ.2)THEN
       DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
     1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
     2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
       w(2) =CF(2,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
     1      + CB(2,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
     2      + CF(2,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
     2      *DFM(4)*D0(2)*D0(5)
     3      + CF(2,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
     3      + CB(2,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
     4      + CF(2,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
     5   + CF(2,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
     5  + CB(2,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
       D00(2)=-(BETAY(2)+GAMUY(2)*A0+GAMVY(2)*B0+GAMWY(2)*E0)
     1             *D0(2)+w(2)
       d(2) = D00(2) + o2init
        END IF
        IF(J.EQ.3)THEN
       DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
     1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
     2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
       DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
     1         + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
     2         + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
       w(3) =CF(3,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
```

```
1      + CB(3,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
2      + CF(3,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
3      + CF(3,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
3      *DFM(8)*D0(4)*D0(5)
3      + CB(3,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
   D00(3)= -(BETAY(3)+GAMUY(3)*A0+GAMVY(3)*B0+GAMWY(3)*E0)
1               *D0(3)+w(3)
   D(3) = D00(3) + h2oinit
     END IF
     IF(J.EQ.4)THEN
    DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    w(4) =CF(4,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
1      + CB(4,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2      + CF(4,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2      + CB(4,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3      + CF(4,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
3      + CB(4,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
4      + CF(4,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
5      + CF(4,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
6      + CF(4,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
6      *DFM(8)*D0(4)*D0(5)
6      + CB(4,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
7   + CF(4,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
7   + CB(4,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
   D00(4)= -(BETAY(4)+GAMUY(4)*A0+GAMVY(4)*B0+GAMWY(4)*E0)
1              *D0(4)+w(4)
   D(4) =  D00(4) + ohinit
     END IF
     IF(J.EQ.5)THEN
    DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    DFM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
1          + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2          + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    DBM(8) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(4)/RMLWT(4)
1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    DFM(9) = D0(1)/RMLWT(1)+D0(2)/RMLWT(2)+D0(3)/RMLWT(3)
1          + D0(4)/RMLWT(4)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
    DBM(9) = D0(2)/RMLWT(2)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
```

```
1          + D0(5)/RMLWT(5)+D0(6)/RMLWT(6)+D0(7)/RMLWT(7)
2          + D0(8)/RMLWT(8)+D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
   w(5) =CF(5,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
1       + CB(5,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2       + CF(5,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2       + CB(5,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3       + CF(5,3)/(RMLWT(1)*RMLWT(4))*FWDRTE(3)*D0(1)*D0(4)
3       + CB(5,3)/(RMLWT(3)*RMLWT(5))*BWDRTE(3)*D0(3)*D0(5)
4       + CF(5,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
4       *DFM(4)*D0(2)*D0(5)
5       + CF(5,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
6       + CF(5,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
6       + CB(5,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
7       + CF(5,8)/(RMLWT(5)*RMLWT(4))*FWDRTE(8)
7       *DFM(8)*D0(4)*D0(5)
7       + CB(5,8)/RMLWT(3)*BWDRTE(8)*DBM(8)*D0(3)
8       + CF(5,9)/RMLWT(5)**2*FWDRTE(9)*DFM(9)*D0(5)**2
8       + CB(5,9)/RMLWT(1)*BWDRTE(9)*DBM(9)*D0(1)
9  + CF(5,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
9  + CB(5,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
  D00(5)= -(BETAY(5)+GAMUY(5)*A0+GAMVY(5)*B0+GAMWY(5)*E0)
1              *D0(5) +w(5)
  D(5) =  D00(5) + hinit
    END IF
    IF(J.EQ.6)THEN
   w(6) =CF(6,1)/(RMLWT(2)*RMLWT(5))*FWDRTE(1)*D0(2)*D0(5)
1     + CB(6,1)/(RMLWT(4)*RMLWT(6))*BWDRTE(1)*D0(4)*D0(6)
2     + CF(6,2)/(RMLWT(1)*RMLWT(6))*FWDRTE(2)*D0(1)*D0(6)
2     + CB(6,2)/(RMLWT(4)*RMLWT(5))*BWDRTE(2)*D0(4)*D0(5)
3   + CF(6,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
3  + CB(6,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
4   + CF(6,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
4  + CB(6,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
  D00(6)= -(BETAY(6)+GAMUY(6)*A0+GAMVY(6)*B0+GAMWY(6)*E0)
1               *D0(6)+w(6)
  D(6) =  D00(6) + oinit
    END IF
    IF(J.EQ.7)THEN
   DFM(4) = D0(1)/RMLWT(1)+D0(3)/RMLWT(3)+D0(4)/RMLWT(4)
1         + D0(6)/RMLWT(6)+D0(7)/RMLWT(7)+D0(8)/RMLWT(8)
2         + D0(9)/RMLWT(9)+D0(10)/RMLWT(10)
   w(7) = CF(7,4)/(RMLWT(5)*RMLWT(2))*FWDRTE(4)
1       *DFM(4)*D0(2)*D0(5)
2     + CF(7,5)/(RMLWT(7)*RMLWT(5))*FWDRTE(5)*D0(7)*D0(5)
3     + CF(7,6)/(RMLWT(7)*RMLWT(5))*FWDRTE(6)*D0(7)*D0(5)
3     + CB(7,6)/(RMLWT(1)*RMLWT(2))*BWDRTE(6)*D0(1)*D0(2)
4     + CF(7,7)/(RMLWT(7)*RMLWT(4))*FWDRTE(7)*D0(7)*D0(4)
```

```
      D00(7)= -(BETAY(7)+GAMUY(7)*A0+GAMVY(7)*B0+GAMWY(7)*E0)
     1            *D0(7) +w(7)
      D(7) =  D00(7) + ho2init
        END IF
        IF(J.EQ.8)THEN
      w(8)=CF(8,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
     1  + CB(8,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
      D00(8)= -(BETAY(8)+GAMUY(8)*A0+GAMVY(8)*B0+GAMWY(8)*E0)
     1             *D0(8)+w(8)
      D(8) = D00(8)+ hn2init
        END IF
        IF(J.EQ.9)THEN
     w(9)=CF(9,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
     1  + CB(9,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
     2  + CF(9,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
     2  + CB(9,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
     3   + CF(9,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
     3  + CB(9,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
      D00(9)= -(BETAY(9)+GAMUY(9)*A0+GAMVY(9)*B0+GAMWY(9)*E0)
     1             *D0(9)+w(9)
      D(9) = D00(9)+ hninit
        END IF
        IF(J.EQ.10)THEN
     w(10)=CF(10,10)/(RMLWT(6)*RMLWT(8))*FWDRTE(10)*D0(6)*D0(8)
     1 + CB(10,10)/(RMLWT(10)*RMLWT(9))*BWDRTE(10)*D0(10)*D0(9)
     2 + CF(10,11)/(RMLWT(9)*RMLWT(2))*FWDRTE(11)*D0(9)*D0(2)
     2 + CB(10,11)/(RMLWT(10)*RMLWT(6))*BWDRTE(11)*D0(10)*D0(6)
     3  + CF(10,12)/(RMLWT(9)*RMLWT(4))*FWDRTE(12)*D0(9)*D0(4)
     3 + CB(10,12)/(RMLWT(10)*RMLWT(5))*BWDRTE(12)*D0(10)*D0(5)
      D00(10)= -(BETAY(10)+GAMUY(10)*A0+GAMVY(10)*B0+GAMWY(10)
     1                *E0)*D0(10)+w(10)
      D(10) = D00(10)+ hn0init
         END IF
      END DO    ! end J-loop for reactions

      do i=1,nspcs
       d0(i) = d(i)
      end do

*  Calculate average molecular weight

      avgmlwt = 0.d0
      do i=1,nspcs
       avgmlwt = avgmlwt + (d(i)+yrslvd(i))/rmlwt(i)
      end do
      avgmlwt = 1.d0/avgmlwt
```

```
*  Thermal energy

       SUMC = 0.d0  ! C0     !maybe the key
       DO I=1,NSPCS
        SUMC = SUMC + C0*ALFATY(I)*D(I)  !*G(I)*D(I)
c        SUMFJ = 0.D0
c        SUMBJ = 0.D0
c        DO J=1,NFRCTN
c         PRODFJ = 1.D0
c         IF(CF(I,J).NE.0.D0)THEN
c           DO L=1,NSPCS
c            PRODFJ = PRODFJ*D(L)**RMLNBR(L,J)
c           END DO
c           SUMFJ = SUMFJ + CF(I,J)*PRODFJ*FWDRTE(J)
c         END IF
c        END DO     ! end j-loop over forward reactions
c        IF(NBRCTN.GT.0)THEN
c          DO J=1,NBRCTNS
c           PRODBJ = 1.D0
c           IF(CB(I,J).NE.0.D0)THEN
c             DO L=1,NSPCS
c              PRODBJ = PRODBJ*D(L)**PMLNBR(L,J)
c             END DO
c             SUMBJ = SUMBJ + CB(I,J)*PRODBJ*BWDRTE(J)
c           END IF
c          END DO   ! end j-loop over backward reactions
c        END IF
        SUMC = SUMC -H(I)*W(I) ! (SUMFJ-SUMBJ)
       END DO        ! end i-loop over species equations

       C = (SUMC-GAMUT*A0*C0-GAMVT*B0*C0-GAMWT*E0*C0)
     1     /(1.d0+BETAT) + cinit
        if (c.ge.c0) sgnc=1.d0
        if (c.lt.c0) sgnc=-1.d0

!       C0=C !!!!!!!
c       TL = TL0 + fctr*TL0*C
        TL = TL0 + fctrt*TL0*C*sgnc
c    TL = TL0 + fctr*TL0*0.5d0*(C+C0)*FCTRT

       summfrc = 0.d0
       do i=1,nspcs
        if(i.eq.4)then
          fctr2 = 1.06*fctr1
        else
          fctr2 = fctr1
        end if
```

```
         y(i) = d(i) + yrslvd(i)*fctr2
         summfrc = summfrc + y(i)
        end do
c          WRITE(*,*) summfrc

        z = (8.d0*y(1)-y(2)+1.d0)/9.d0
ccc        y(3) = 1.d0 - (y(1)+y(2)+y(4)+y(5)+y(6))
        WRITE(11,200)TIME,A0,B0,E0,c,SUMC,(d0(I),I=1,NSPCS)
        WRITE(8,200)TIME,A0,B0,E0,tl,(y(I),I=1,NSPCS)
       1           z,avgmlwt,summfrc !tl,

        IF(N.GE.NSTATS)THEN
          SUMMWT = SUMMWT + AVGMLWT
          AVGMFRC = AVGMFRC + SUMMFRC
          UBAR = UBAR + A0
          VBAR = VBAR + B0
          WBAR = WBAR + E0   !Add one term
          TBAR = TBAR + TL
          USQ = USQ + A0**2
          VSQ = VSQ + B0**2
          WSQ = WSQ + E0**2  !Add one term
          TSQ = TSQ + C**2
          CTU = CTU + A0*C
          CTV = CTV + B0*C
          CTW = CTW + E0*C   !Add one term
          CUV = CUV + A0*B0
          CUW = CUW + A0*E0  !Add one term
          CVW = CVW + B0*E0  !Add one term
          DO J=1,NSPCS
           YBAR(J) = YBAR(J) + d(J)
           YSQ(J) = YSQ(J) + d(J)*d(J)
           CUY(J) = CUY(J) + A0*d(J)
           CVY(J) = CVY(J) + B0*d(J)
           CWY(J) = CWY(J) + E0*d(J)  !Add one term
           CTY(J) = CTY(J) + C*d(J)
           DO I=1,NSPCS
            CYIYJ(I,J) = CYIYJ(I,J) + d(I)*d(J)
           END DO
          END DO
        END IF


        A0 = A
        B0 = B
        E0 = E !Add one term
        C0 = C
            if(dabs(a).gt.5.d0.or.dabs(b).gt.5.d0.or.
```

```
     1           dabs(e).gt.5.d0)then      !!!change  5/50
              itypsln = 13
              solntyp = 13.d0
              fvdata(iu0,jv0,kw0,1) = 13.d0
              fvdata(iu0,jv0,kw0,2) = 0.0d0
              fvdata(iu0,jv0,kw0,3) = 2.d0
c              fvdata(iu0,jv0,kw0,4) = -1.d0
c              fvdata(iu0,jv0,kw0,5) = 7.d0

c              fvdata(iu0,jv0,kw0,6) = -1.d0
c              fvdata(iu0,jv0,kw0,7) = 3.d0
              exit
            end if
            if(dabs(a).lt.0.1**20d0)then
              a=0.d0
              itypsln = 0.d0
              solntyp = 0.d0
              fvdata(iu0,jv0,kw0,1) = 0.d0
              fvdata(iu0,jv0,kw0,2) = 0.d0
              fvdata(iu0,jv0,kw0,3) = 0.d0
              exit
            end if
            if(dabs(b).lt.0.1**20d0)then
              b=0.d0
            end if

c           if(k.eq.1)then
             u(n) = a
             v(n) = b
             w1(n) = e  !change
             q(1,n) = a
             q(2,n) = b
             q(3,n) = e !change
c           else
c            uf(i) = a
c            vf(i) = b
c            wf(i) = c
c            q(1,i) = a
c            q(2,i) = b
c            q(3,i) = c
c           end if
           end do !end n-loop
           write(*,*)'Time evolution ended.'
          write(*,*)'  '

c       END DO           ! end n-loop
```

```
NAVG = NSTPS - NSTATS - 1

UBAR = UBAR/NAVG
VBAR = VBAR/NAVG
WBAR = WBAR/NAVG !Add one term
TBAR = TBAR/NAVG
SUMMWT = SUMMWT/NAVG
AVGMFRC = AVGMFRC/NAVG
USQ = USQ/NAVG
VSQ = VSQ/NAVG
WSQ = WSQ/NAVG   !Add one term
TSQ = TSQ/NAVG
CUV = CUV/sqrt(USQ*VSQ)/NAVG
CUW = CUW/sqrt(USQ*WSQ)/NAVG   !Add one term
CVW = CVW/sqrt(VSQ*WSQ)/NAVG   !Add one term
CTU = CTU/sqrt(USQ*TSQ)/NAVG
CTV = CTV/sqrt(VSQ*TSQ)/NAVG
CTW = CTW/sqrt(WSQ*TSQ)/NAVG   !Add one term
DO J=1,NSPCS
 YBAR(J) = YBAR(J)/NAVG
 YSQ(J) = YSQ(J)/NAVG
 CUY(J) = CUY(J)/sqrt(USQ*YSQ(J))/NAVG
 CVY(J) = CVY(J)/sqrt(VSQ*YSQ(J))/NAVG
 CWY(J) = CWY(J)/sqrt(WSQ*YSQ(J))/NAVG  !Add one term
 CTY(J) = CTY(J)/sqrt(TSQ*YSQ(J))/NAVG
END DO
DO J=1,NSPCS
 DO I=1,NSPCS
  CYIYJ(I,J) = CYIYJ(I,J)/sqrt(YSQ(I)*YSQ(J))/NAVG
 END DO
END DO

WRITE(9,*)'  '
WRITE(9,*)'  '
WRITE(9,*)'  Time Averages'
WRITE(9,*)'  '
WRITE(9,*)' UBAR = ',UBAR
WRITE(9,*)' VBAR = ',VBAR
WRITE(9,*)' WBAR = ',WBAR    !Add one term
WRITE(9,*)' TBAR = ',TBAR
WRITE(9,*)' MLWT = ',SUMMWT
WRITE(9,*)' MFRC = ',AVGMFRC
DO I=1,NSPCS
 WRITE(9,*)' YBAR(',I,') = ',YBAR(I)
END DO
WRITE(9,*)'  '
WRITE(9,*)'  Variances'
```

```
      WRITE(9,*)' USQ = ',USQ
      WRITE(9,*)' VSQ = ',VSQ
      WRITE(9,*)' WSQ = ',WSQ     !Add one term
      WRITE(9,*)' TSQ = ',TSQ
      DO I=1,NSPCS
       WRITE(9,*)' YSQ(',I,') = ',YSQ(I)
      END DO
      WRITE(9,*)'  '
      WRITE(9,*)'  Correlations'
      WRITE(9,*)' CUV = ',CUV
      WRITE(9,*)' CUW = ',CUW     !Add one term
      WRITE(9,*)' CVW = ',CVW     !Add one term
      WRITE(9,*)' CTU = ',CTU
      WRITE(9,*)' CTV = ',CTV
      WRITE(9,*)' CTW = ',CTW     !Add one term
      WRITE(9,*)'  '
      write(9,*)' J ',','    ',','    CUY(J)    ',','    ',','     CVY(J)
',
     1           '    ',','   CWY(J)    ',','    ',','   CTY(J)    '
      do j=1,NSPCS
      write(9,'(1x,I2,3(3x,1pe13.6))')j,CUY(J),CVY(J),CWY(J),CTY(J)
      end do
      WRITE(9,*)'  '
      DO I=1,NSPCS
       DO J=I+1,NSPCS
        WRITE(9,*)' CY(',I,',',J,') = ',CYIYJ(I,J)
       END DO
      END DO

      return

      END


*_____________________________________________________________________*

      subroutine stat1(q,dqdt,dqdx,dqdy,dqdz,corfns,pdfs,s2,s3,
     1      s4,s6,f, s,sm,qpcor,qpsq,qmax,qmin,qavg,rlscl,
     2      tintscl, acl1,dt,npts,nstat,npdf,nauto,itype,nlsf)

      implicit real*8 (a-h,o-z)

      parameter(imx=50001,ivr=3,namx=1000,npdfmx=1000,nlmx=100)

*  Basic variables and derivatives
      dimension q(ivr,0:imx),dqdt(ivr,imx),dqdx(ivr,imx),
     1          dqdy(ivr,imx), dqdz(ivr,imx)
```

```
*  Classical turbulence statistics:  correlations,
*  autocorrelations, PDFs, means, flatness and skewness
      dimension corfns(ivr,0:namx),pdfs(ivr,npdfmx),
     1     qpcor(ivr,ivr), qpsq(ivr),qmax(ivr),qmin(ivr),
     2     qavg(ivr),tintscl(ivr), acl1(ivr),f(5*ivr),s(5*ivr)

*  Statistical quantities arising in the Kolmogorov theories:
structure
*  functions of various orders
      dimension s2(5*ivr,nlmx),s3(5*ivr,nlmx),s4(5*ivr,nlmx),
     1          s6(5*ivr,nlmx),sm(5*ivr),rlscl(nlmx)

*  Temporary arrays
      dimension dq(ivr),qsq(0:ivr+1)

      data sclfctr /1.d1/   ! multiplicative factor for
             !         application of Taylor's hypothesis

      data mxsf /5/          ! maximum number of structure
      !        function evaluations as a function of "distance"

*
*  1. Initialize (zero) arrays
*
      dqdt = 0.d0
      dqdx = 0.d0
      dqdy = 0.d0
      dqdz = 0.d0
      corfns = 0.d0
      pdfs = 0.d0
      s2 = 0.d0
      s3 = 0.d0
      s4 = 0.d0
      s6 = 0.d0
      sm = 0.d0
      s = 0.d0
      f = 0.d0
      qpcor = 0.d0
      qpsq = 0.d0
      qmax = 0.d0
      qmin = 1.d0
      qavg = 0.d0
      tintscl = 0.d0
      acl1 = 0.d0
      dq = 0.d0
*
```

```
*  2. Calculate maxs, mins, avgs, etc.
*
      navg = npts - nstat
      navg1 = navg - 1
      itype2 = itype + 2

      do j=1,itype2
       do i=nstat,npts
        if(q(j,i).gt.qmax(j))then
       qmax(j) = q(j,i)
     endif
        if(q(j,i).lt.qmin(j))then
       qmin(j) = q(j,i)
     endif
        qavg(j) = qavg(j) + q(j,i)
       end do
      end do

      qavg = qavg/navg1

*
*  3. Calculate averages of squared fluctuations,
*  correlations, pdfs.
      dq = (qmax-qmin)/npdf

      do j=1,itype2
       do i=nstat,npts
        difq = q(j,i) - qavg(j)
        qpsq(j) = qpsq(j) + difq**2
        do k=j+1,itype2
         qpcor(j,k) = qpcor(j,k) + difq*(q(k,i)-qavg(k))
        end do  ![k]
        do k=1,npdf
         qlwr = qmin(j) + (k-1)*dq(j)
         qupr = qmin(j) + k*dq(j)
         if(k.eq.1.and.q(j,i).ge.qlwr.and.q(j,i).lt.qupr)then
           pdfs(j,k) = pdfs(j,k) + 1.d0
           exit
         end if
         if(k.gt.1.and.q(j,i).gt.qlwr.and.q(j,i).le.qupr)then
           pdfs(j,k) = pdfs(j,k) + 1.d0
           exit
         end if
        end do  ![k]
       end do  ![i]
      end do  ![j]
```

```
      qpcor = qpcor/navg1
      qpsq = qpsq/navg1
      pdfs = pdfs/(navg+1)

      do j=1,itype2
       do k=j+1,itype2
        qpcor(j,k) = qpcor(j,k)/dsqrt(qpsq(j)*qpsq(k))
       end do  ![k]
      end do  ![j]

*
*  4. Calculate autocorrelations, integral time scales,
*      L1 norm of autocorrelation functions.
*
      do j=1,itype2
       do k=0,nauto
        do i=nstat,npts-k
         corfns(j,k) = corfns(j,k)+(q(j,i)-qavg(j))*
     1            (q(j,i+k)-qavg(j))
        end do  ![i]
        corfns(j,k) = corfns(j,k)/((navg1-k)*qpsq(j))
        tintscl(j) = tintscl(j) + corfns(j,k)
        acl1(j) = acl1(j) + dabs(corfns(j,k))
       end do  ![k]
      end do  ![j]

      tintscl = tintscl/nauto
      acl1 = acl1/nauto

*
*  5. Calculate derivative time series:  use mean-squared
*     fluctuations to estimate velocity scales; then apply
*     the Taylor hypothesis to obtain spatial derivatives.
*
      urs = sclfctr*sqrt(qpsq(1))
      vrs = sclfctr*sqrt(qpsq(2))
      wrs = sclfctr*sqrt(qpsq(3))
      mdt = int(urs) + 1
      ndt = int(vrs) + 1
      ldt = int(wrs) + 1

      do j=1,itype2
       do i=nstat,npts
        dqdt(j,i) = q(j,i) - q(j,i-1)
        dqdx(j,i) = (q(j,i)-q(j,i-1))/(q(1,i)+q(1,i-1))
        dqdy(j,i) = (q(j,i)-q(j,i-1))/(q(2,i)+q(2,i-1))
      dqdz(j,i) = (q(j,i)-q(j,i-1))/(q(3,i)+q(3,i-1))
```

```
ccc         dqdt(j,i) = q(j,i) - q(j,i-1)
ccc         dqdx(j,i) = q(j,i) - q(j,i-mdt)
ccc         dqdy(j,i) = q(j,i) - q(j,i-ndt)
ccc         dqdz(j,i) = q(j,i) - q(j,i-ldt)
       end do  ![i]
      end do  ![j]
ccc      dqdt = dqdt/dt
ccc      dqdx = dqdx/(mdt*dt)
ccc      dqdy = dqdy/(ndt*dt)
ccc      dqdz = dqdz/(ldt*dt)

*
*  6. Calculate flatness and skewness of variables and
*      derivatives.
      do j=1,itype2
       qsq = 0.d0
       do i=nstat,npts
        difq = q(j,i) - qavg(j)
        qsq(0) = qsq(0) + difq**2
        qsq(1) = qsq(1) + dqdt(j,i)**2
        qsq(2) = qsq(2) + dqdx(j,i)**2
        qsq(3) = qsq(3) + dqdy(j,i)**2
     qsq(4) = qsq(4) + dqdz(j,i)**2
        s(j) = s(j) + difq**3
        s(j+itype2) = s(j+itype2) + dqdt(j,i)**3
        s(j+2*itype2) = s(j+2*itype2) + dqdx(j,i)**3
        s(j+3*itype2) = s(j+3*itype2) + dqdy(j,i)**3
        s(j+4*itype2) = s(j+4*itype2) + dqdz(j,i)**3
        f(j) = f(j) + difq**4
        f(j+itype2) = f(j+itype2) + dqdt(j,i)**4
        f(j+2*itype2) = f(j+2*itype2) + dqdx(j,i)**4
        f(j+3*itype2) = f(j+3*itype2) + dqdy(j,i)**4
        f(j+4*itype2) = f(j+4*itype2) + dqdz(j,i)**4
       end do  ![i]
       if(qsq(0).le.1.d-4)then
         s(j) = 0.d0
         f(j) = 1.d0/navg
       else
         s(j) = s(j)/qsq(0)**1.5d0
         f(j) = f(j)/qsq(0)**2
       end if
       if(qsq(1).le.1.d-4)then
         s(j+itype2) = 0.d0
         f(j+itype2) = 1.d0/navg
       else
         s(j+itype2) = -s(j+itype2)/qsq(1)**1.5d0
         f(j+itype2) = f(j+itype2)/qsq(1)**2
```

```
        end if
        if(qsq(2).le.1.d-4)then
          s(j+2*itype2) = 0.d0
          f(j+2*itype2) = 1.d0/navg
        else
          s(j+2*itype2) = -s(j+2*itype2)/qsq(2)**1.5d0
          f(j+2*itype2) = f(j+2*itype2)/qsq(2)**2
        end if
        if(qsq(2).le.1.d-4)then
          s(j+3*itype2) = 0.d0
          f(j+3*itype2) = 1.d0/navg
        else
          s(j+3*itype2) = -s(j+3*itype2)/qsq(3)**1.5d0
          f(j+3*itype2) = f(j+3*itype2)/qsq(3)**2
        end if
        if(qsq(3).le.1.e-12)then
          s(j+4*itype2) = 0.d0
       f(j+4*itype2) = 1.d0/navg
        else
          s(j+4*itype2) = -s(j+4*itype2)/qsq(4)**1.5d0
       f(j+4*itype2) = f(j+4*itype2)/qsq(4)**2
        endif
       end do  ![j]

       f = navg*f
       s = sqrt(dfloat(navg))*s

*
*  7. Calculate structure functions of orders 2, 3, 4 and 6
*  for variables and their derivatives. Shell model structure
*  functions (as defined by Jensen et al., 1991) are calculated
*  first, followed by those of the basic variables, and then
*  those of the t, x and y derivatives.
*
       velscl = sqrt(urs**2+vrs**2+wrs**2)
       phi2 = datan(sqrt(qpsq(2)/qpsq(1)))
       phi3 = datan(sqrt(qpsq(3)/qpsq(1)))
       nlsf = max(min(mdt,ndt,ldt),mxsf)
       do j=1,itype2
        do i=nstat,npts-nlsf
         do k=1,nlsf
          rlscl(k) = k*dt*velscl
          difq1k = q(1,i+k) - q(1,i)
          difq2k = q(2,i+k) - q(2,i)
       difq3k = q(3,i+k) - q(3,i)
          if(k.eq.1)then
            sm(j) = sm(j) + q(j,i)**2
```

```
        sm(j+itype2) = sm(j+itype2) + q(j,i)**3
        sm(j+2*itype2) = sm(j+2*itype2) + q(j,i)**4
        sm(j+3*itype2) = sm(j+3*itype2) + q(j,i)**6
      end if
      if(j.eq.1)then
        s2(j,k) = s2(j,k) + (difq1k*cos(phi2)**2+
     1                       difq2k*sin(phi2)**2+
     1                       difq3k*sin(phi3)**2)**2
        s3(j,k) = s3(j,k) + (difq1k*cos(phi2)**2+
     1                       difq2k*sin(phi2)**2+
     1                       difq3k*sin(phi3)**2)**3
        s4(j,k) = s4(j,k) + (difq1k*cos(phi2)**2+
     1                       difq2k*sin(phi2)**2+
     1                       difq3k*sin(phi3)**2)**4
        s6(j,k) = s6(j,k) + (difq1k*cos(phi2)**2+
     1                       difq2k*sin(phi2)**2+
     1                       difq3k*sin(phi3)**2)**6
      end if
      if(j.eq.2)then
        s2(j,k) = s2(j,k) + (((difq1k+difq2k+difq3k)*
     1                 sin(phi2)*cos(phi2)*sin(phi3))**2)**2
        s3(j,k) = s3(j,k) + (((difq1k+difq2k+difq3k)*
     1                 sin(phi2)*cos(phi2)*sin(phi3))**2)**3
        s4(j,k) = s4(j,k) + (((difq1k+difq2k+difq3k)*
     1                 sin(phi2)*cos(phi2)*sin(phi3))**2)**4
        s6(j,k) = s6(j,k) + (((difq1k+difq2k+difq3k)*
     1                 sin(phi2)*cos(phi2)*sin(phi3))**2)**6
      end if
      if(j.gt.2)then
        difqjk = q(j,i+k) - q(j,i)
        s2(j,k) = s2(j,k) + difqjk**2
        s3(j,k) = s3(j,k) + difqjk**3
        s4(j,k) = s4(j,k) + difqjk**4
        s6(j,k) = s6(j,k) + difqjk**6
      end if
      difdqt = dqdt(j,i+k) - dqdt(j,i)
      difdqx = dqdx(j,i+k) - dqdx(j,i)
      difdqy = dqdy(j,i+k) - dqdy(j,i)
   difdqz = dqdz(j,i+k) - dqdz(j,i)
      s2(j+ivr,k) = s2(j+ivr,k) + difdqt**2
      s2(j+2*ivr,k) = s2(j+2*ivr,k) + difdqx**2
      s2(j+3*ivr,k) = s2(j+3*ivr,k) + difdqy**2
   s2(j+4*ivr,k) = s2(j+4*ivr,k) + difdqz**2
      s3(j+ivr,k) = s3(j+ivr,k) + difdqt**3
      s3(j+2*ivr,k) = s3(j+2*ivr,k) + difdqx**3
      s3(j+3*ivr,k) = s3(j+3*ivr,k) + difdqy**3
   s3(j+4*ivr,k) = s3(j+4*ivr,k) + difdqz**3
```

```
         s4(j+ivr,k) = s4(j+ivr,k) + difdqt**4
         s4(j+2*ivr,k) = s4(j+2*ivr,k) + difdqx**4
         s4(j+3*ivr,k) = s4(j+3*ivr,k) + difdqy**4
       s4(j+4*ivr,k) = s4(j+4*ivr,k) + difdqz**4
         s6(j+ivr,k) = s6(j+ivr,k) + difdqt**6
         s6(j+2*ivr,k) = s6(j+2*ivr,k) + difdqx**6
         s6(j+3*ivr,k) = s6(j+3*ivr,k) + difdqy**6
       s6(j+4*ivr,k) = s6(j+4*ivr,k) + difdqz**6
        end do  ![k]
       end do  ![i]
      end do  ![j]

      sm = sm/navg1
      s2 = s2/navg1
      s3 = s3/navg1
      s4 = s4/navg1
      s6 = s6/navg1

      return

      end

*---------------------------------------------*

      subroutine psd(f,w,p,dt,nstop,nfft)


      implicit real*8 (a-h,o-z)
      parameter (imx=50001,nfftmx=16384)
      dimension f(0:imx),t(nfftmx),p(nfftmx),w(nfftmx)

      iavg = 0
      iwndo = 1
      list = 0
      nstrt = 1
      nrmlzf = 0
      nrmlzt = 0
      nrmlzp = 0
      nrmlzw = 0
      tmax = 1.d0
ccc   fmax = 0.001d0
      do i=1,nstop
       t(i) = (i-1)*dt
      end do

      nfft = nstop
      if(nrmlzt.ge.0)tmax = t(nstop)
```

```
      if(nstrt.ne.1)then
        do i=1,nfft+1
         f(i) = f(i+nstrt-1)
         t(i) = t(i+nstrt-1)
        end do
      end if
      if(nrmlzf.gt.0)call nrmlze(f,fmax,nfft)
      if(nrmlzt.gt.0)call nrmlze(t,tmax,nfft)
      if(nrmlzf.lt.0.or.nrmlzt.lt.0)then
        do i=1,nfft
         if(nrmlzt.lt.0)t(i) = tmax*t(i)
         if(nrmlzf.lt.0)f(i) = fmax*f(i)
        end do
      end if
      if(iavg.gt.0)then
        sum = 0.d0
        do j=1,nfft
         sum = sum + f(j)
        end do
        avg = sum/nfft
        do j=1,nfft
         f(j) = f(j) - avg
        end do
      end if
      dt = t(2) - t(1)
      call pwrspc(f,p,w,dt,nfft,iwndo)
      if(nrmlzw.gt.0)call nrmlze(w,wmax,nfft)
      if(nrmlzp.gt.0)call nrmlze(p,pmax,nfft)

      return
      end

*-------------------------------------------*

      subroutine nrmlze(g,gmax,n)
      implicit real*8 (a-h,o-z)
      dimension g(*)

      if(gmax.le.1.d-12)then
        gmax = dabs(g(1))
        do i=2,n
         if(dabs(g(i)).gt.gmax)gmax = dabs(g(i))
        end do
      end if

      do i=1,n
       g(i) = g(i)/gmax
```

```
      end do

      return
      end

*-------------------------------------------------------*

      subroutine pwrspc(psdr,smag,sarg,dt,nfft,iwndo)
      implicit real*8(a-h,o-z)
      parameter (npt=16384)
      dimension psdi(npt)
      dimension psdr(*),smag(*),sarg(*)

      rnfft = dfloat(nfft) + 0.1d0
      iexp = dlog10(rnfft)/dlog10(2.d0) + 0.1d0

      df = 1.d0/(nfft*dt)
      do i=1,nfft
       psdi(i) = 0.d0
      end do

      call ctfft(psdr,psdi,iexp)
      if(iwndo.gt.0)call window(psdr,psdi,smag,sarg,nfft,iwndo)
      call mgntd(psdr,psdi,smag,sarg,df,nfft)

      return
      end

*-------------------------------------------------------*

      subroutine mgntd(si,sq,smag,sarg,d,nfft)
      implicit real*8(a-h,o-z)
      dimension si(*),sq(*),smag(*),sarg(*)

      nfft2 = nfft/2
      bias = nfft2*d
      do i=1,nfft
       amp = dsqrt(si(i)**2)
       amp = si(i)**2 + sq(i)**2
       tmp = amp
       if(amp.le.1.d-15)amp = 1.d-15
       smag(i) = 1.d1*dlog10(amp)
ccc    smag(i) = amp
       sarg(i) = (i-1)*d-bias
      end do
      call normal(smag,nfft)
      do i=1,nfft2
```

```
        indx = nfft2+i
        temp = smag(i)
        smag(i) = smag(indx)
        smag(indx) = temp
      end do
      do i=1,nfft2
        smag(i) = smag(nfft2+i)
        sarg(i) = sarg(nfft2+i)
      end do
      nfft = nfft2

      return
      end

*-------------------------------------------------------*

      subroutine ctfft(x,y,mfft)
      implicit real*8 (a-h,o-z)
      integer rep,disp
      dimension x(*),y(*)

      pi = 4.0d0*datan(1.0d0)
      n = 2**mfft
      call order(x,y,n)
      do i=1,mfft
        rep = 2**i
        disp = rep/2
        arg = 2.d0*pi/rep
        do j=1,disp
          twf = (j-1)*arg
          c = dcos(twf)
          s = dsin(twf)
          do k=j,n,rep
            j2 = k+disp
            t1 = c*x(j2) + s*y(j2)
            t2 = -s*x(j2) + c*y(j2)
            x(j2) = x(k) - t1
            y(j2) = y(k) - t2
            x(k) = x(k) + t1
            y(k) = y(k) + t2
          end do ![k]
        end do  ![j]
      end do   ![i]

      do i=1,n
        x(i) = x(i)/n
        y(i) = y(i)/n
```

```
      end do

      return
      end

*---------------------------------------------*

      subroutine order(x,y,n)
      implicit real*8(a-h,o-z)
      dimension x(*),y(*)

      nd2=n/2
      nm1=n-1
      j=1

      do 30 i=1,nm1
      if(i.ge.j)go to 10
      t1=x(j)
      x(j)=x(i)
      x(i)=t1
      t2=y(j)
      y(j)=y(i)
      y(i)=t2
  10  k=nd2
  20  if(k .ge. j)go to 30
      j=j-k
      k=k/2
      go to 20
   30 j=j+k

      return
      end

*-------------------------------------------------*

      subroutine normal(smag,nfft)
      implicit real*8 (a-h,o-z)
      dimension smag(*)

      big = 0.d0
      do i=1,nfft
       if(smag(i).gt.big)big = smag(i)
      end do
      do i=1,nfft
       tmp = smag(i)
       smag(i) = smag(i) - big
      end do
```

```
      return
      end

*-------------------------------------------------------*

      subroutine window(psdr,psdi,smag,sarg,nfft,iwndo)
      implicit real*8 (a-h,o-z)
      dimension psdr(*),psdi(*),smag(*),sarg(*),a(3)
      data a /-0.1817d0,-0.1707d0,-0.1476d0/
      data ampltd /0.9245d0/
      nfft1 = nfft - 1
      nfft2 = nfft - 2
      do 30 n=1,nfft
      if(n.eq.1)then
        xm1 = psdr(nfft)
        ym1 = psdi(nfft)
        xm2 = psdr(nfft1)
        ym2 = psdi(nfft1)
        xm3 = psdr(nfft2)
        ym3 = psdi(nfft2)
        go to 20
      else if(n.eq.2)then
        xm2 = psdr(nfft)
        ym2 = psdi(nfft)
        xm3 = psdr(nfft1)
        ym3 = psdi(nfft1)
        go to 15
      else if(n.eq.3)then
        xm3 = psdr(nfft)
        ym3 = psdi(nfft)
        go to 10
      else if(n.eq.nfft2)then
        xp3 = psdr(1)
        yp3 = psdi(1)
        go to 5
      else if(n.eq.nfft1)then
        xp2 = psdr(1)
        yp2 = psdi(1)
        xp3 = psdr(2)
        yp3 = psdi(2)
        go to 5
      else if(n.eq.nfft)then
        xp1 = psdr(1)
        yp1 = psdi(1)
        xp2 = psdr(2)
        yp2 = psdi(2)
```

```
        xp3 = psdr(3)
        yp3 = psdi(3)
        go to 5
      end if
    5 xm3 = psdr(n-3)
      ym3 = psdi(n-3)
   10 xm2 = psdr(n-2)
      ym2 = psdi(n-2)
   15 xm1 = psdr(n-1)
      ym1 = psdi(n-1)
   20 x0 = psdr(n)
      y0 = psdi(n)
      if(n.eq.nfft)go to 25
      xp1 = psdr(n+1)
      yp1 = psdi(n+1)
      if(n.eq.nfft1)go to 25
      xp2 = psdr(n+2)
      yp2 = psdi(n+2)
      if(n.eq.nfft2)go to 25
      xp3 = psdr(n+3)
      yp3 = psdi(n+3)
   25 smag(n) = x0 + a(1)*(xm1+xp1) + a(2)*(xm2+xp2)
     1                + a(3)*(xm3+xp3)
   30 sarg(n) = y0 + a(1)*(ym1+yp1) + a(2)*(ym2+yp2)
     1                + a(3)*(ym3+yp3)
      do 35 n=1,nfft
      psdr(n) = ampltd*smag(n)
   35 psdi(n) = ampltd*sarg(n)

      return
      end

*-------------------------------------------------*


      subroutine psdanlyzr(frq,pwr,solntyp,nfft,itypsln)

      implicit real*8 (a-h,o-z)

      parameter(nfftmx=16384)

      dimension frq(nfftmx),pwr(nfftmx)

      dimension isvcntr(nfftmx),jsvcntr(nfftmx)

      nsyflg = 0
      cutoff = -150.d0
```

```
      fctr1 = 0.001d0
      fctr2 = 0.01d0
      plvl = 20.d0

      rmin = cutoff
      psum = 0.d0
      do i=1,nfft
       if(pwr(i).lt.rmin)rmin = pwr(i)
       psum = psum + pwr(i)
      end do

      pavg = psum/nfft

ccc      write(*,*)'pavg =',pavg
      if(rmin.lt.cutoff)cutoff = rmin

      icntr = 0
      ipcntr = 0
      ispcr = 9
      itypsln = 0
      iexp = 0
      solntyp = 0.d0
      isvcntr = 0
      jsvcntr = 0

*  Count number of peaks, and store locations at which they
*  occur, assuming "clean" power spectrum
      do i=5,nfft
       dpm = 0.d0
       dp = 0.d0
       if(pwr(i).gt.pavg)then
         dpm = pwr(i) - pwr(i-1)
         if(i.lt.nfft)then
           dpp = pwr(i+1) - pwr(i)
           dp = dpm*dpp
         end if
         if(dp.le.0.d0.and.dpm.gt.0.d0)then
           icntr = icntr + 1    !---increment number of peaks
           isvcntr(icntr) = i   !---store location of peak
         end if
       end if
       if(dabs(pwr(i)-cutoff).lt.1.d-12)ipcntr = ipcntr + 1
ccc    !---count nonpeaks
      end do   ! end i-loop
ccc      write(*,*)'icntr =',icntr,'  ipcntr =',ipcntr,
ccc     1 '  isvcntr(icntr) =',isvcntr(icntr)
      icntr0 = icntr
```

```
*  Begin corrections needed to account for noisy data
      nsyflg = 0
      if(icntr.gt.ipcntr.or.icntr.gt.nfft/4)then
        nsyflg = 1            ! noisy w/ fundamental
        ipwr = 0
        do i=1,icntr
         if(pwr(isvcntr(i)).ge.pavg+plvl)ipwr = ipwr + 1
        end do
        if(ipwr.eq.0)then
          nsyflg = 2          ! noisy w/o fundamental
ccc        write(*,*)'nsyflg =',nsyflg
          icntr = 0
          go to 20
        end if
ccc        write(*,*)'nsyflg =',nsyflg
        jcntr = 0
        dpm = 0.d0
        do j=ispcr,nfft
*           !---count prominent peaks inside noisy data
         dpm = 0.d0
         dp = 0.d0
         if(pwr(j).gt.plvl+pavg)then
           dpm = pwr(j) - pwr(j-1)
           if(j.lt.nfft)then
             dpp = pwr(j+1) - pwr(j)
             dp = dpm*dpp
           end if
           if(dp.le.0.d0.and.dpm.gt.0.d0)then
             jcntr = jcntr + 1   !---increment noisy peak count
             jsvcntr(jcntr) = j
           end if
         end if
         if(j.eq.nfft.and.jsvcntr(jcntr).ge.
       1   int(0.999d0*nfft))then
           plclavg = 0.d0
           do jj=nfft-3*ispcr,nfft-ispcr
            plclavg = plclavg + pwr(jj)
           end do
           plclavg = plclavg/(2*ispcr)
           if(pwr(nfft).gt.0.25d0*plvl+plclavg)then
             jcntr = jcntr + 1  !---increment noisy peak count
             jsvcntr(jcntr) = j
           end if
         end if
        end do   ! end j-loop
ccc      write(*,*)'jcntr =',jcntr,'
```

```
ccc      jsvcntr(jcntr) =',jsvcntr(jcntr)
*  Refine the peak count for noisy data by finding the maximum
*  peak within each neighborhood of a selected peak
        icntr = jcntr
        isvcntr = jsvcntr
        jcntr = 0
        jsvcntr = 0
        do i=1,icntr
         if(isvcntr(i).gt.2*ispcr.and.isvcntr(i).
       1                 lt.nfft-ispcr)then
           jstrt = isvcntr(i) - ispcr
           jstop = isvcntr(i) + ispcr
         else if(isvcntr(i).le.2*ispcr)then
           jstrt = ispcr
           jstop = isvcntr(i) + ispcr
         else if(isvcntr(i).ge.nfft-ispcr)then
           jstrt = isvcntr(i) - ispcr
           jstop = nfft
         end if
         jsv = 0
         pmax = pwr(jstrt)
         do j=jstrt,jstop
          if(pwr(j).gt.pmax)then
            pmax = pwr(j)
            jsv = j
          end if
         end do  ! end j-loop
         if(jsv.gt.0)then
           jcntr = jcntr + 1
           jsvcntr(jcntr) = jsv
         end if
        end do  ! end i-loop
        if(jcntr.gt.0)then
          icntr = jcntr
          isvcntr = jsvcntr
        end if
      end if

      if(icntr.gt.1)then   !---remove spurious peaks from
        do k=1,icntr0     !---non-noisy and noisy data
         jcntr = 0
         if(k.gt.1)jsvcntr = isvcntr
         do i=1,icntr-1
          idfsvcnt = isvcntr(i+1) - isvcntr(i)
          if(idfsvcnt.le.ispcr)then
            if(pwr(isvcntr(i)).le.pwr(isvcntr(i+1)))then
              jsvcntr(i+1) = isvcntr(i+1)
```

```
              jsvcntr(i) = 0
             else
            if(jsvcntr(i).ne.0.or.k.eq.1)jsvcntr(i) = isvcntr(i)
            jsvcntr(i+1) = 0
             end if
           else
            if(jsvcntr(i).ne.0.or.k.eq.1)jsvcntr(i) = isvcntr(i)
            if(i.eq.icntr-1)jsvcntr(i+1) = isvcntr(i+1)
           end if
          end do

          do i=1,icntr
ccc           write(*,*)'i =',i,' isvcntr(i) =',isvcntr(i),
ccc     1     ' jsvcntr(i) =',jsvcntr(i),' pwr(isvcntr)',
ccc     2     pwr(isvcntr(i))
          end do

          jcntr = icntr
          icntr = 0
          do j=1,jcntr  !---reload peak locations after clean up
           if(jsvcntr(j).gt.0)then
             icntr = icntr + 1
             isvcntr(icntr) = jsvcntr(j)
           end if
          end do
          if(icntr.eq.jcntr)exit
        end do
      end if

ccc      write(*,*)'icntr =',icntr,'  ipcntr =',ipcntr

*  Determine type of behavior based on icntr and nsyflg values
   20 if(icntr.eq.0)then  ! broadband w/o fundamental
        itypsln = 12
        solntyp = 12.d0
        return
      end if

      if(icntr.eq.1)then
        if(nsyflg.eq.0)then    ! periodic
          if(isvcntr(icntr).gt.int(0.999d0*nfft))then
            itypsln = 1
            solntyp = 1.d0
          else              ! periodic w/ different fundamental
            itypsln = 2
            solntyp = 2.d0
          end if
```

```
          return
        else if(nsyflg.eq.1)then ! broadband w/ fundamental
          if(isvcntr(icntr).gt.int(0.999d0*nfft))then
            itypsln = 10
            solntyp = 10.d0
          else           ! broadband w/ different fundamental
            itypsln = 11
            solntyp = 11.d0
          end if
          return
        end if
      end if

      if(icntr.gt.1)then ! check for subharmonic,
                                 ! quasiperiodic and phase lock
*  Determine whether spectral peaks are evenly spaced
        idfcntr = 1
        frqmx = frq(isvcntr(icntr))
        do i=1,icntr-1
         frq1 = frq(isvcntr(i))
         frq2 = frq(isvcntr(i+1))
         if(dabs(icntr*(frq2-frq1)/frqmx-1.d0).lt.0.01d0)
     1    idfcntr = idfcntr + 1
        end do

*  Determine whether number of peaks is power of 2
        iexp = 0
        if(isvcntr(icntr).gt.int(0.999d0*nfft).and.
     1   mod(icntr,2).eq.0)
     1   then
          do j=1,icntr/2
           if(icntr.eq.2**j)then
             iexp = j
             exit
           end if
          end do
        end if
ccc        write(*,*)'idfcntr =',idfcntr,' iexp =',iexp

        if(iexp.gt.0.and.idfcntr.eq.icntr)then
ccc     ! subharmonic or phase lock
          mgtdcnt = 0
          if(pwr(isvcntr(1)).lt.pwr(isvcntr(2)))mgtdcnt = 1
          do i=3,icntr-1,2
           if(pwr(isvcntr(i)).lt.pwr(isvcntr(i+1)).and.
     1      pwr(isvcntr(i)).lt.pwr(isvcntr(i-1)))
     2      mgtdcnt = mgtdcnt + 1
```

```
          end do

          if(mgtdcnt.eq.icntr/2.or.icntr.eq.2)then !subharmonic
            if(nsyflg.eq.0)then
              itypsln = 3
              solntyp = 3.d0 !+ fctr2*iexp
            else if(nsyflg.eq.1)then
              itypsln = 6
              solntyp = 6.d0 !+ fctr2*iexp
            end if
            return
          else if(mgtdcnt.ne.icntr/2)then      ! phase locked
            if(nsyflg.eq.0)then
              itypsln = 4
              solntyp = 4.d0 !+ fctr1*icntr
            else if(nsyflg.eq.1)then
              itypsln = 7
              solntyp = 7.d0
            end if
            return
          end if
        else if(iexp.eq.0.and.idfcntr.eq.icntr)then
ccc           ! phase locked w/
          if(nsyflg.eq.0)then
ccc           ! arbitrary # of frqs
            itypsln = 4
            solntyp = 4.d0 !+ fctr1*icntr
          else if(nsyflg.eq.1)then
            itypsln = 7
            solntyp = 7.d0
          end if
          return
        end if

        if(idfcntr.ne.icntr)then                ! quasiperiodic
          if(nsyflg.eq.0)then
            itypsln = 5
            solntyp = 5.d0 !+ fctr1*icntr
          else if(nsyflg.eq.1)then
ccc        write(*,*)'isvcntr(icntr) =',isvcntr(icntr)
            if(isvcntr(icntr).gt.int(0.999d0*nfft))then
              itypsln = 8
              solntyp = 8.d0
            else
              itypsln = 9
              solntyp = 9.d0
            end if
```

```
            end if
            return
          end if
        end if

        return

        end

*-------------------------------------------------------------------------*
*-------------------------------------------------------------------------*
```

Bibliography

[1] Nicolas Docqier., Sénastien Candel, Combustion control and sensors: a review. *Progress in Energy and Combustion Science*, 28:107–150, 2002.

[2] G. A. Richards, M. M. McMillian, R. S. Gemmen, W. A. Rogers, and S. R. Cully, Issues for low-emission, fuel-flexible power systems, *Progress in Energy and Combustion Science*, 27(2):141–169, 2001.

[3] W. Meier, S. Prucker, M. H. Cao , and W. Stricker, Characterization of turbulent H_2/N_2/Air jet diffusion flames by single-pulse spontaneous Raman scattering, *Combustion Sci and Tech*, 118:293–312, 1996.

[4] A. Neuber, G. Krieger, M. Tacke, E. Hassel, and J. Janicka. Finite rate chemistry and NO molefraction in non-premixed turbulence flames, *Combustion and Flame*, 113:198–211, 1998.

[5] Suresh K. Aggarwal, Structure of unsteady partially premixed flames and the existence of state relationships, *International Journal of Spray and Combustion Dynamics*, 1(3):339-363, 2009.

[6] Ishwar K. Puri, Suresh K. Aggarwal, Andrew J. Lock. PARTIALLY PREMIXED FLAME (PPF) RESEARCH FOR FIRE SAFETY. Retrieved from: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20040161243.pdf

[7] K. N. C. Bray, M. Champion, P. A. Libby, and N. Swaiminathan, Finite rate chemistry and presumed PDF models for premixed turbulent combustion, *Combustion and Flame*, 146:665–673, 2006.

[8] Matthew J. Dunn, Assaad R. Masri, Robert W. Bilger, and Robert S. Barlow, Finite Rate Chemistry Effects in Highly Sheared Turbulent Premixed Flames, *Flow Turbulence Combust*, 85:621–628, 2010.

[9] R. P. Lindstedt, S. A. Louloudi, J. J. Driscoll, and V. Sick, Finite Rate Chemsitry Effects in Turbulent Reacting Flows, *Flow, Turbulence and Combustion*, 72:407–426, 2004.

[10] Mohanmmad Irannezhad, A Numerical Study of Reacting Flows Using Finite Rate Chemistry, PhD Thesis, Chalmers University of Technology, Göteborg, Sweden, 2012.

[11] D. Healy, D. M. Kalitan, C. J. Aul, E. L. Petersen, G. Bourque, and H. J. Curran, Oxidation of C1-C5 Alkane Quinternary Natural Gas Mixtures at High Pressures. *Energy and Fuels*, 24:1521–1528, 2010.

[12] Q.–D. Wang, Skeletal Mechanism Generation for High-Temperature Combustion of H2/CO/C1-C4 Hydrocarbons, *Energy and Fuels*, 27:4021–4030, 2013.

[13] C. K. Westbrook, W. J. Pitza, O. Herbineta, H. J. Currana, and E. J. Silkea, A comprehensive detailed chemical kinetic reaction mechanism for combustion of n-alkane hydrocarbons from n-octane to n-hexadecane, *Combustion and Flame*, 156(2):181–199, 2009.

[14] G. Dixon-Lewis, F. A. Goldsworthy, and J. B. Greenberg, Flame Structure and Flame Reaction Kinetics IX. Calculation of Properties of Multi-Radical Premixed Flames, *Proceedings of the Royal Society of London*, 346:261–278, 1975.

[15] W. C. Gardner, Jr. (Ed.), *Gas-Phase Combustion Chemistry*, Springer-Verlag, New York, 1984.

[16] M. Frenklach, H. Wang, J. Rabinowitz, Optimization and analysis of large chemical kinetic mechanisms using the solution mapping method–combustion of methane, *Progress in Energy Combustion Science*, 18:47–73, 1992.

[17] P. Dagaut, M. Reuillon, J. C. Boetner, M. Cathonnet, Kerosene Combustion at Pressures up to 40 Atm: Experimental Study and Detailed Chemical Kinetic Modeling, *Proceedings of the Combustion Institute*, 25(1):919–926, 1994.

[18] N. M. Marinov, W. J. Pitz, C. K. Westbrook, M. J. Castaldi, S. M. Senkan, Modeling of aromatic and polycyclic aromatic hydrocarbon formation in premixed methane and ethane flames, *Combustion Science and Technology*, 116:211–287, 1996.

[19] H. J. Curran, P. Gaffuri, W. J. Pitz, C. K. Westbrook, A comprehensive modeling study of n-heptane oxidation, *Combust and Flame*, 114:149–177, 1998.

[20] E. Ranzi, M. Dente, A. Goldaniga, G. Bozzano, T. Faravelli, Lumping procedures in detailed kinetic modeling of gasification, pyrolysis, partial oxidation and combustion of hydrocarbon mixtures, *Progress in Energy and Combustion Science*, 27:99–139, 2001.

[21] J. M. Simmie, Detailed chemical kinetic models for the combustion of hydrocarbon fuels, *Progress in Energy and Combustion Science*, 29:599–634, 2003.

[22] M. V. Petrova and F. A. Williams, A small detailed chemical-kinetic mechanism for hydrocarbon combustion, *Combustion and Flame*, 144:526–544, 2006.

[23] Jürgen Warnatz , Ulrich. Maas, and R. W. Dibble, *Combustion: Physical and Chemical Fundamentals, Modeling and Simulation, Experiments, Pollutant Formation*, 3^{rd} Edition, 2001.

[24] P. Saxena, and F. A. Williams, Testing a small detailed chemical-kinetic mechanism for the combustion of hydrogen can carbon monoxide, *Combustion and Flame*, 145:316–323, 2006.

[25] F. A. Williams, Detailed and reduced chemistry for hydrogen autoignition, *Journal of Loss Prevention in the Process Industries*, 21:131–135, 2008.

[26] F. Mauss, N. Peters, B. Rogg, F. A. Willams, Reduced Kinetic Mechanism for Premixed Hydrogen Flames, in *Reduced Kinetic Mechanism for Application in Combustion Systems*, N. Peters, B. Rogg, (Eds.), *Springer-Verlag* , Berlin, Heidelberg, pages 29–43, 1993.

[27] K. Seshadri, N. Peters, F. A. Williams, Asymptotic analyses of stoichiometric and lean hydrogen-air flames, *Combustion and Flame*, 96:407–427, 1994.

[28] D. Fernández-Galisteo, A. L. Sánchez, A Liñán, F. A. Williams, One-step reduced kinetics for lean hydrogenair deflagration, *Combustion and Flame*, 156:985–996, 2009.

[29] D. Fernández-Galisteo, A. L. Sánchez, A Liñán, F. A. Williams, The hydrogenair burning rate near the lean flammability limit, *Combustion Theory and Modelling*, 13:741–761, 2009.

[30] E. Guthiel, G. Balarkrishnan, F. A. Williams, Structure and Extinction of Hydrogen-Air Diffusion Flams, in *Reduced Kinetic Mechanism for Application in Combustion Systems*, N. Peters, B. Rogg, (Eds.), *Springer-Verlag*, Berlin, Heidelberg, pages 177–195, 1993.

[31] G. Balarkrishnan, M. D. Smooke, F. A. Williams, A numerical investigation of extinction and ignition limits in laminar Nonpremixed Counterflowing hydrogen-air streams for both elementary and reduced chemistry, *Combustion and Flame*, 102:329–340, 1995.

[32] P. Boivin, C. Jiménez A. L. Sánchez, F. A. Williams, An explicit reduced mechanism for H_2-air combustion, *Proceedings of the Combustion Institute*, 30:517–523, 2010.

[33] Overview of Intro to MPI class, *Dartmouth College*, modified Feb 14, 2011, Retrieved from: http://www.dartmouth.edu/~rc/classes/intrompi/

[34] D. Carbonell, C. D. Perez-Segarra, P. J. Coelho, A. Oliva, Flamelet mathematical models for non-premixed laminar combustion, *Combustion and Flame*, 156:334–347, 2009.

[35] J. Boussinesq, Essai sur la théorie des eaux courantes, *Mém. prés. par div. savant à l'Acad. Sci.*, 1877.

[36] R. G. Lerner, G. L. Trigg, *Encyclopaedia of Physics (2nd Edition)*, VHC publishers, 1991, ISBN (Verlagsgesellschaft) 3-527-26954-1, ISBN (VHC Inc.) 0-89573-752-3.

[37] Millennium Prize Problems, Clay Mathematics Institute, retrieved 2014-01-14.

[38] Sir Horace Lamb, *Hydrodynamics 4th Edition*, University Press, Cambridge, 1916.

[39] Peter Bradshaw. "Turbulence: the chief outstanding difficulty of our subject." *Experiments in fluids* 16:203-216, 1994.

[40] J. M. McDonough, *Introductory Lectures on Turbulence Physics, Mathematics and Modeling*, Lecture notes, website of Advanced CFD Group. Retrieved from: http://www.engr.uky.edu/∼acfd/lecturenotes1.html

[41] Marcel Lesieur, and Oliver M´tais, New Trends in Large-eddy Simulation of Turbulence, *Annu. Rev. Fluid. Mech.*, 28:45–82, 1996.

[42] A. G. Kravchenko, and P. Moin, On the Effect of Numerical Errors in Large Eddy Simulations of Turbulent Flows, *Journal of Computational Physics*, 131:310–322, 1997.

[43] Charles G. Speziale, Analytical Methods for the Development of Reynolds-Stress Closures in Turbulence, *Annu. Rev. Fluid. Mech*, 23:107-157, 1991.

[44] Parviz Moin, and Krishnan Mahesh, Direct Numerical Simulation: a tool in turbulence research *Annu. Rev. Fluid Mech*, 30:539-578, 1998.

[45] Robert S. Rogallo, Numerical Experiments in Homogeneous Turbulence, *Volume 81315 of NASA technical memorandum*, National Aeronautics and Space Administration, 1981.

[46] Douglas G. Fox, and Douglas K. Lilly, Numerical Simulation of Turbulent Flows, *Reviews of Geophysics and Space Physics*, 10:51–72, 1972.

[47] Steven A. Orszag, and G. S. Patterson, Numerical Simulation of Turbulence, *Statistical Models and Turbulence Lecture Notes in Physicals*, 12:127–147, 1972.

[48] J. J. Riley, R. W. Metcalfe, Direct numerical simulations of a perturbed, turbulent mixing layer, *American Institute of Aeronautics and Astronautics*, 1980.

[49] Robert S. Rogallo, and Parviz Moin, Numerical Simulation of Turbulent Flows, *Annu. Rev. Fluid Mech.*, 16:99–137, 1984.

[50] John Kim, Parviz Moin, and Robert Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, *J. Fluid. Mech.*, 177:133–166, 1987.

[51] Robert D. Moser, and Parviz Moin, The effects of curvature in wall-bounded turbulent flows, *J. Fluid. Mech.*, 175:479–510, 1987.

[52] N. Kasagi, Y. Tomita, and A. Kuroda, Direct numerical simulation of passive scalar field in a turbulent channel flow, *J. Heat Transfer*, 114:598–606, 1992 .

[53] R. Kristoffersen, and H. I. Andersson, Direct simulations of low Reynolds number turbulent flow in a rotating channel, *J. Fluid. Mech.*, 256:163–197, 1993.

[54] João C. Neves, Parviz Moin, and Robert D. Moser, Effects of convex transverse curvature on wall-bounded turbulence. Part 1. The velocity and vorticity, *J. Fluid. Mech.*, 272:349–382, 1994.

[55] João C. Neves, Parviz Moin, and Robert D. Moser, Effects of convex transverse curvature on wall-bounded turbulence. Part 2. The pressure fluctuations, *J. Fluid. Mech.*, 272:383–406, 1994.

[56] Y. Sumitani, and N. Kasagi, Direct numerical simulation of turbulent transport with uniform wall injection and suction. *AIAA J.*, 33:1220–1228, 1995.

[57] Philippe R. Spalart, Numerical study of sink-flow boundary layers. *J. Fluid. Mech.*, 172:307–328, 1986.

[58] Philippe R. Spalart, Direct simulation of a turbulent boundary layer up to R_ϑ=1410, *J. Fluid. Mech.*, 187:61–98, 1988.

[59] W. J. Feiereisen, W. C. Reynolds, J. H. Ferziger, Numerical simulation of a compressible homogeneous, turbulent shear flow, Report TF-13, Standford University, Dept. of Mechanical Engineering, 1981.

[60] Gordon Erlebacher, M. Y. Hussaini, H. O. Kreiss, S. Sarkar, The analysis and simulation of compressible turbulence, *Theoret. Comput. Fluid Dynamics*, 2:73–95, 1990.

[61] S. Sarkar, G. Erlebacher, and M. Y. Hussaini, Direct Simulation of Compressible Turbulence in a Shear Flow, *Theoret. Comput. Fluid Dynamics*, 2:291–305, 1991.

[62] Sangsan Lee, Sanjiva K. Lele, and Parviz Moin, Eddy shocklets in decaying compressible turbulence, *Phys. Fluid A*, 3:657-664, 1991.

[63] G. A. Blaisdell, N. N. Mansour, and W. C. Reynolds, Compressibility effects on the growth and structure of homogeneous turbulent shear flow. *J. Fluid. Mech.*, 256:443–485, 1993.

[64] Gary N. Coleman, John Kim, and R. D. Moser, A numerical study of turbulent supersonic isothermal-wall channel flow, *J. Fluid. Mech.*, 305:159–183, 1995.

[65] Man Mohan Rai, Thomas B. Gatski, and Gordon Erlebacher, Direct Simulation of Spatially Evolving Compressible Turbulent Boundary Layers, *33^{rd} Aerospace Sciences Meeting and Exhibit*, AlAA 9590583, 1995.

[66] David C. Wilcox, Turbulence modeling for CFD, *DCW industries*, La Canada, CA, Vol. 3, 1993.

[67] C. J. Freitas, Perspective: Selected Benchmarks From Commercial CFD Codes, *J. Fluids Eng.*, 117(2):208–218, 1995.

[68] J. Boussinesq, Théorie de l'écoulement tourbillonnant et tumultueux des liquides dans les lits rectilignes a grande section, *Paris, Gauthier-Villars et fils*, 1897.

[69] Osborne Reynolds, On the dynamical theory of incompressible viscous fluids and the determination of the criterion, *Philosophical Transactions of the Royal Society of London. A*, 186:123–164, 1895.

[70] Ludwig Prandtl, Bericht Über die ausgebildete Turbulenz, *Z. agnew. Math. Mech.*, 5:136–139, 1925.

[71] Th. Von Kármán, Mechanical similitude and turbulence, *NACA-TM-611*, 1931 (1930).

[72] Th. Von Kármán, Progress in the statistical theory of turbulence, *Proceedings of the National Academy of Sciences of the United States of America 34*, 11:530–539, 1948.

[73] Ludwig Prandtl, Über ein neues Formelsystem für die ausgebildete Turbulenz, *Nachrichten der Akademie der Wissenschaften zu Güttingen, Mathematisch-physikalische Klasse*, S:6–19, 1945.

[74] J. C. Rota, Statistische Theorie nichthomogener Turbulenz, *Zeitschrift fürr Physik*, 129:547–572, 1951.

[75] Bart J. Daly, and Francis H. Harlow, Transport Equations in Turbulence, *Physics of Fluids*, 13:2634-2649, 1970.

[76] C. Dup. Donaldson, Calculation of Turbulent Shear Flows for Atmospheric and Vortex Motions, *AIAA Journal*, 10:4–12, 1972.

[77] B. E. Launder, G. J. Reece, and W. Rodi, Progress in the development of a Reynolds-stress turbulence closure, *Journal of Fluid Mechanics*, 68:537–566, 1975.

[78] John L. Lumley, Computational modeling of turbulent flows, *Advances in applied mechanics*,18:123–176, 1979.

[79] Charles G. Speziale, Second-order closure models for rotating turbulent flows, *NASA Report 1*, 1985.

[80] S. B. Pope, Consistent modeling of scalars in turbulent flows, *Physicals of Fluids*, 26:3448–3450, 1982.

[81] D. C. Haworth, and S. B. Pope, A generalized Langevin model for turbulent flows, *Physics of Fluids*, 29:387–405, 1986.

[82] Charles G. Speziale, Analytical Methods for the Development of Reynolds-Stress Closures in Turbulence, *Annual Review of Fluid Mechanics*, 23:107–157, 1991.

[83] V. Yakhot, S. A. Orszag, S. Thangam, T. B. Gatski, & C. G. Speziale, Development of turbulence models for shear flows by a double expansion technique, *Physics of Fluids A*, 4:1510–1520, 1992.

[84] B. Wang, P. Miles, R. Reitz, Z. Han, *et al.*, Assessment of RNG turbulence modeling and the development of a generalized RNG closure model, *SAE Technical Paper*, No. 2011-01-0829, 2011,

[85] Fang Wang, Rolf D. Reitz, Cecile Pera, Zhi Wang, and Jianxin Wang, Application of Generalized RNG Turbulence Model to Flow in Motored Single-Cyclinder PFI Engine, *Engineering Applications of Computational Fluid Mechanics*, 7:486–495, 2013.

[86] Fedina E. and Fureby C., A comparative study of flamelet and finite rate chemistry LES for an axisymmetric dump combustor, *Journal of Turbulence*, 12:1–20, 2011.

[87] Joseph Smagorinsky, General Circulation Experiments with the Primitive Equations, *Monthly Weather Review*, 91(3):99–164, March 1963.

[88] Parviz Moin, and John Kim, Numerical investigation of turbulent channel flow, *Journal of Fluid Mechanics*, 118:341–377, 1982.

[89] Charles Meneveau, and Joseph Katz, Scale-invariance and turbulence models for large-eddy simulation, *Annual Review of Fluid Mechanics*, 32:1–32, 2000.

[90] Pierre Sagaut, Large eddy simulation for incompressible flows, *Heidelberg: Springer-Verlag*, 2002.

[91] N. Branley, and W. P. Jones, Large eddy simulation of a turbulent non-premixed flame, *Combustion and Flame*, 127:1914–1934, 2001.

[92] J. Volavy, M. Forma, and M. Jicha, Large Eddy Simulation: subgrid-scale models, *Acta technica*. 56:271–280, 2011.

[93] I. Mahle, J. Sesterhenn, and R. Friedrich, Large eddy simulation of turbulent reacting shear layers including finite-rate chemistry and detailed diffusion processes, *Flow, Turbulence and Combustion*, 80:81–105, 2008.

[94] Heinz. Pitsch, Large-eddy simulation of turbulent combustion, *Annu. Rev. Fluid Mech*, 38:453–482, 2006.

[95] C. Fureby, A comparative study of flamelet and finite rate chemistry LES for a swirl stabilized flame, *Journal of Engineering for Gas Turbines and Power*, 134(4):1–12, No.041503, 2012.

[96] Joseph Mathew, Large Eddy Simulation, *Defense Science Journal*, 60(6):598–605, 2010.

[97] C. Moussaed, S. Wornom, M. V. Salvetti, B. Koobus and A. Dervieux, Impact of dynamic subgrid-scale modeling in variational multiscale large-eddy simulation of bluff-body flows, *Acta Mechanica*, 225(12):3309–3323, 2014.

[98] S. R. Gubba, S. S. Ibrahim, and W. MG Malalasekera, A dynamic SGS model for LES of turbulent premixed flames, *ICHMT DIGITAL LIBRARY ONLINE*, 13 (2008).

[99] M. Germano, U. Piomelli, P. Moin, and W. H. Cabot, A dynamic subgrid–scale eddy viscosity model. *Physics of Fluids A: Fluid Dynamics (1989-1993)*, 3(7):1760–1765, 1991.

[100] W. Zeng, R. Fu, and J. M. McDonough, Lower-order Discrete Dynamical System for H_2-air Finite-Rate Chemistry in 3D, *10th International Conference on Heat Transfer, Fluid Mechanics and Thermodynamics*, Jul 14–16, 2014.

[101] U. Frisch, TURBULENCE The Legacy of A. N. Kolmog- orov, *Cambridge University Press*, Cambridge, pages 89–92, 1995.

[102] McDonough J. M. and Zhang Sha, Discrete dynamical systems models of turbulence-chemical kinetics interactions, 37^{th} *Intersociety Energy Conversion Engineering Conference*, Washington, DC, July 28–31, 2002.

[103] J. M. McDonough and Sha Zhang, LES subgrid-scale models of turbulence-chemical kinetics interactions based on discrete dynamical systems, 32^{th} *AIAA Fluid Dynamics Conference*, St. Louis, June 24–27, 2002.

[104] May R. M., Simple mathematical models with very complicated dynamics, *Nature*, 261:459–467, 1976.

[105] J. M. McDonough and M. T. Huang, A 'poor man's Navier–Stokes equation': derivation and numerical experiments—the 2-D case, *Int. J. Numer. Meth. Fluids*, 44:545–578, 2004.

[106] Ch. Schneider, A. Dreizler, J. Janicka, and E. P. Hassel, Flow Field Measurements of Stable and Locally Extinguishing Hydrocarbon-Fuelled Jet Flames, *Combustion and Flame*, 135:85–190, 2003.

[107] J. B. Polly, *Numerical Experiment of the 3-D "Poor Man's Navier–Stokes Equations"*, MS Thesis, University of Kentucky, 2012.

[108] Anthony N. Michel, Ling Hou and Derong Liu, *Stability of Dynamical Systems–Continuous, Discontinuous, and Discrete Systems*, Boston: Birkhauser, pages 1–4 and 103–137, 2008.

[109] Oded Galor, Discrete Dynamical System, Springer-Verlag, Berlin, pages 92–107, 2007.

[110] J. M. McDonough Lecture in Elementary Fluid Dynamics–Physics, Mathematics and Applications, pages 47–100, Lecture notes, website of Advanced CFD Group. Retrieved from: http://www.engr.uky.edu/∼acfd/lecturenotes1.html

[111] J. Leray, Sur le mouvement d'un liquide visqueux emplissant l'espace, *Acta Mathematica*, 63:193–248, 1934.

[112] J. M. McDonough, *Lectures in Basic Computational Numerical Analysis*, Lecture notes. Retrieved from: http://www.engr.uky.edu/∼acfd/lecturenotes1.html

[113] S. A. Bible, Study of the "Poor Man's Navier-Stokes" Equation Turbulence Model, Master thesis, University of Kentucky, 2003.

[114] Juan Li, Zhenwei Zhao, Andrei Kazakov, and Frederick L. Dryer, An updated comprehensive kinetic model of hydrogen combustion, *Int. J. Chemical Kinetics*, 36:566–575, 2004.

[115] M. R. Beychok, NOX emission from fuel combustion controlled, *The Oil and Gas Journal*, pages 53–56, 1973.

[116] J. Janicka, and W. Kollmann, A two-variables formalism for the treatment of chemical reactions in turbulent H_2–Air diffusion flames *Seventeenth Symposium (International) on Combustion*, The combustion institute, Pittsburgh, 17:421-430, 1979.

Vita

Mr. Zeng got his bachelor degree in thermal energy and power engineering from Northeastern University, China, 2012. During 2012-2015, he conducted research on kinetic chemical reaction in the advanced Computational Fluid Dynamics (CFD) lab of the University of Kentucky, and his supervisor was Dr. McDonough. Specifically, he investigated a low-order discrete dynamical system for jet-flame finite rate chemistry combustion process, and he studied reduced mechanisms, bifurcation parameters, temperature-species phase portraits for this discrete dynamical system in the global flame field. He started to work as a combustion engineer in the PFMAN LLC since 2015.