UCRL-TH-209324

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Cubic Spline Collocation Method for the Simulation of Turbulent Thermal Convection in Compressible Fluids

V. M. Castillo

January 29, 1999

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Cubic Spline Collocation Method for the Simulation of Turbulent Thermal Convection in Compressible Fluids

by

Victor Manuel Castillo, Junior

B.S., Physics (San Jose State University) 1988
M.S., Computational Physics (San Jose State University) 1992
M.S., Engineering-Applied Science (University of California, Davis) 1997

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Engineering-Applied Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at DAVIS/LIVERMORE

Approved by the committee in charge:

Professor Emeritus William G. Hoover, Chair — Date

Professor Garry H. Rodrigue — Date

Professor Emeritus Richard M. Christensen — Date

1999

Cubic Spline Collocation Method for the Simulation of Turbulent Thermal Convection in Compressible Fluids

Copyright 1999

by

Victor Manuel Castillo, Junior

Abstract

Cubic Spline Collocation Method for the Simulation of Turbulent Thermal Convection in Compressible Fluids

by

Victor Manuel Castillo, Junior

Doctor of Philosophy in Engineering-Applied Science

University of California at Davis/Livermore

Professor Emeritus William G. Hoover, Chair

A collocation method using cubic splines is developed and applied to simulate steady and time-dependent, including turbulent, thermally convecting flows for two-dimensional compressible fluids. The state variables and the fluxes of the conserved quantities are approximated by cubic splines in both space direction. This method is shown to be numerically conservative and to have a local truncation error proportional to the fourth power of the grid spacing. A "dual-staggered" Cartesian grid, where energy and momentum are updated on one grid and mass density on the other, is used to discretize the flux form of the compressible Navier-Stokes equations. Each grid-line is staggered so that the fluxes, in each direction, are calculated at the grid midpoints. This numerical method is validated by simulating thermally convecting flows, from steady to turbulent, reproducing

known results. Once validated, the method is used to investigate many aspects of thermal convection with high numerical accuracy.

Simulations demonstrate that multiple steady solutions can coexist at the same Rayleigh number for compressible convection. As a system is driven further from equilibrium, a drop in the time-averaged dimensionless heat flux (and the dimensionless internal entropy production rate) occurs at the transition from laminar-periodic to chaotic flow. This observation is consistent with experiments of real convecting fluids. Near this transition, both harmonic and chaotic solutions may exist for the same Rayleigh number. The chaotic flow loses phase-space information at a greater rate, while the periodic flow transports heat (produces entropy) more effectively. A linear sum of the dimensionless forms of these rates connects the two flow morphologies over the entire range for which they coexist. For simulations of systems with higher Rayleigh numbers, a scaling relation exists relating the the dimensionless heat flux to the two-seventh's power of the Rayleigh number, suggesting the existence of "hard" turbulence in two-dimensional compressible convection.

Acknowledgements

The completion of this dissertation represents the end of a great part of my life – my "graduate experience". There are a number of people that I must thank.

First and foremost, I must thank my research advisor, Professor William G. Hoover, for his guidance through this time. His method of teaching by example and imparting intuition has made a lasting impression on me. He was never once too busy to discuss my work or offer counsel on any subject. Professor Hoover allowed me the freedom to choose my own path and was there to illuminate the way.

I must also thank Dr. Carol G. Hoover for her guidance and advice concerning the practical issue that each graduate student eventually must face – getting a real job. Carol introduced me to the Methods Development Group and some of the interesting computational engineering work done at LLNL. Carol graciously proofread this dissertation and provided valuable suggestions.

I thank the members of my Thesis Committee – Professors William G. Hoover, Garry H. Rodrigue, and Richard M. Christensen – each of whom taught some of the finest classes at DAS. I also thank the members of the Oral Committee – Garry H. Rodrigue (chair), Berni J. Alder, and Richard M. Christensen (Engineering Applied Science); Bruce R. White and Mohamed M. Hafez (Mechanical and Aeronautical Engineering); and Louise H. Kellogg (Department of Geology). The diverse background of this group helped to broaden my knowledge of this subject. I must thank the members of the LLNL Student Policy Committee for providing the financial support that made graduate school possible for me. I would especially like to thank William McCurdy for his help in gaining the SE position, and Ann Orel and Bill Chandler for giving a local guy a chance in this world-class program. I also thank the staff at DAS – Donna Clifford, Estelle Miller, and Archie Johnson – for making the administrative issues much less of a burden and for making DAS a fun place.

I really owe a debt of gratitude to my friends at DAS for dragging me away from the computer and convincing me that if I went away for a few days, my work would still be there when I returned. These guys made the graduate experience more exciting by introducing me to rock climbing, backpacking, mountain biking, snowboarding, etc. The liquid seminars and cigar nights allowed us to share our experience and bond as if we were *real* students.

My family has been very supportive during this time. I especially want to thank Mom for always supporting and encouraging my scientific curiosities since kindergarten. I always got a chemistry set on my birthday. Mom, you definitely raised a geek.

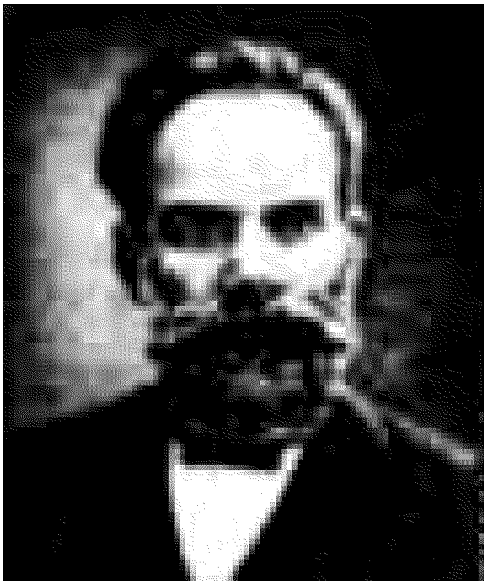
I would also like to thank individuals who have offered technical advise or in some way made my graduate experience more enjoyable. These include Oyeon Kum, Koichiro Shida, Alan Hindmarsh, and Kelly Barrett (LLNL); Fred Wolters, Keith Mainquist and David Sandoval (Clorox); and Lui Lam, Roger Dodd, Patrick Hamill, and Alex Garcia (San Jose State).

Last, but not least, I must thank a very special person – *thank you* Cecilia for your understanding, companionship, love, and encouragement.

Livermore, California V. M. C.
January, 1999

"If a body lighter than a fluid be forcibly immersed in it, the body will be driven upwards by a force equal to the difference between its weight and the weight of the fluid displaced."

The sixth proposition from
$\pi\epsilon\rho\iota\ \ o\chi o\nu\mu\epsilon\nu\omega\nu$ (On Floating Bodies),
Archimedes of Syracuse, circa $260\,\mathrm{B.C.}$ [1]



"When among the roots of the determinantal equation there are some for which the real parts are positive, the undisturbed motion is unstable."

Theorem III from
Общая задача об устойчивости движения
(On the General Problem of the Stability of Motion), Aleksandr M. Lyapunov, 1892. [2]

[1] English translation from "The Works of Archimedes" T. L. Heath, Dover, New York (1912).

[2] Translated from Russian into French by Édouard Davaux, in *Communications de la Société mathématique de Kharkow* (1893). Translated from French into English by A. T. Fuller in International Journal of Control, **55**(1992).

Contents

I Motivation and Definitions

1 Introduction **2**
- 1.1 Overview . . . 2
- 1.2 Review of previous work . . . 6
  - 1.2.1 Early work . . . 7
  - 1.2.2 Oberbeck-Boussinesq approximation . . . 9
  - 1.2.3 Modern work . . . 12

2 Thermal convection **19**
- 2.1 Heat transport in two dimensions . . . 19
  - 2.1.1 Fourier conduction . . . 20
  - 2.1.2 Rayleigh-Bénard instability . . . 20
  - 2.1.3 Harmonic convection . . . 22
  - 2.1.4 Chaotic and turbulent convection . . . 22
- 2.2 Examples of Convection . . . 24
  - 2.2.1 Atmospheric convection . . . 24
  - 2.2.2 Mantle convection . . . 25
  - 2.2.3 Stellar convection . . . 27
  - 2.2.4 Bénard's and other laboratory experiments . . . 28

3 Factors influencing convection **32**
- 3.1 Description of the fluid . . . 32
  - 3.1.1 Thermal and mechanical equation of state . . . 33
  - 3.1.2 Isotropy and transport coefficients . . . 34
- 3.2 Rayleigh number . . . 38
- 3.3 Nusselt number . . . 41

II Numerical Methods

4 Discretization **44**
4.1 Navier-Stokes equations . . . 44
4.1.1 Eulerian and Lagrangian forms . . . 45
4.1.2 Conservative (flux) form . . . 46
4.2 Staggered grid . . . 47
4.2.1 Example: One-dimensional heat equation . . . 47
4.3 Boundaries . . . 52
4.4 "Dual-staggered" grid . . . 53

5 Cubic spline collocation **58**
5.1 Background . . . 58
5.1.1 Finite difference methods . . . 59
5.1.2 Finite element methods . . . 60
5.1.3 Padé approximation . . . 61
5.1.4 Collocation . . . 62
5.2 Cubic spline approximations . . . 64
5.3 Midpoint interpolation and differentiation . . . 66
5.4 Accuracy – local truncation error . . . 67
5.5 Flux conservation . . . 72
5.6 Calculation of the spline function . . . 76
5.6.1 Fixed boundaries . . . 76
5.6.2 Periodic boundaries . . . 78
5.7 Time integration . . . 82

6 Validation of the method **86**
6.1 Description of system . . . 86
6.2 Extrapolation to continuum limit . . . 88
6.3 Validating results . . . 90
6.3.1 Steady flow . . . 90
6.3.2 Time-dependent flow . . . 92
6.3.3 Turbulent flow . . . 94

III Applications

7 Steady compressible convection **96**
7.1 Coexisting solutions for steady flow . . . 98
7.2 Min-max internal entropy production principle . . . 104

8 Harmonic and chaotic convection **108**
8.1 Transition to harmonic flow . . . 108
8.2 Transition to chaotic flow . . . 110
8.3 Coexisting flow morphologies . . . 114
8.3.1 Entropy production and the loss of phase space information . . . 116

9 Convective turbulence **123**
9.1 Non-Boussinesq (asymmetric) flow . . . 125
9.2 Scaling . . . 126

IV Conclusions

10 Conclusions and prospects **131**
10.1 Numerical method . . . 133
10.2 Two-dimensional compressible convection . . . 138
10.2.1 Coexisting attractors in compressible convection . . . 139
10.2.2 Heat flux at the onset of chaotic convection . . . 141
10.2.3 Coexistence of "harmonic" and chaotic flows . . . 141
10.2.4 Entropy production and Lyapunov instability at the onset of turbulent convection . . . 142
10.2.5 Turbulence in two-dimensional compressible convection . . . 143

V Appendix

A Convecting double pendulum **146**
A.1 Introduction . . . 146
A.2 Results and discussion . . . 150
A.3 Future work . . . 153
A.4 Listing . . . 154

B FORTRAN codes **155**
B.1 The Navier-Stokes solver . . . 155
B.1.1 Header file . . . 155
B.1.2 Setup and initialization . . . 156
B.1.3 Cubic spline subroutines . . . 157
B.1.4 Runge-Kutta subroutines . . . 160
B.2 Calculation of maximum Lyapunov exponent . . . 163
B.3 Calculation of internal entropy production . . . 169

C MATHEMATICA scripts **171**
C.1 Velocity and temperature field plots 171
C.2 Vorticity field plots 172

Bibliography **173**

List of Figures

1.1 Bénard's original experimental apparatus 8

2.1 Velocity field of steady thermal convection of a compressible fluid enclosed between two rigid thermal boundaries. 21
2.2 Dimensionless heat flux as a function of time for harmonic convection. . 23
2.3 Diagrams of ideal convection (left) and realistic moist-air convection (right) in the atmosphere. [*From Klemp (1987).*] 25
2.4 Cross-section diagram of the Earth showing the UM&C: upper mantle and crust $(0 - 670\,\mathrm{km})$, LM: convecting lower mantle $(670 - 2,890\,\mathrm{km})$, OC: liquid outer core $(2,890 - 5,150\,\mathrm{km})$, and IC: solid inner core $(5,150 - 6,370\,\mathrm{km})$. 27
2.5 Solar granulation showing very small-scale turbulent motions (from La Palma observatory) 28
2.6 Bénard cells in spermaceti. A reproduction from one of Bénard's original photographs. [*From Chandrasekhar (1961).*] 29
2.7 Shadowgraph of convecting glycerol at $Ra = 2.3 \times 10^8$. [*From Zhang (1997).*] 30

4.1 Convergence of the centered-difference approximation of the diffusion equation to the wrong solution using a simple grid. The temperatures are factors of A. 50
4.2 Convergence of the centered-difference approximation of the diffusion equation to the correct solution using a staggered grid. 51
4.3 Internal energy and momentum are updated on one grid of the "dual-staggered" mesh while the mass density is updated on the other. 53
4.4 Fluxes of conserved quantities are updated at the midpoints (staggered) along each grid line of the "dual-staggered" grid. 54

5.1 Cubic spline function and its spatial derivatives 66

6.1 Simple diagram of two-dimensional steady Rayleigh-Bénard convection. . 87

6.2 Results extrapolated to the continuum limit. 89
6.3 Global error in calculation of the horizontal kinetic energy. A least-squares fit through this data shows that the error scales linearly with the fourth power of the grid resolution ($1/N$). The data is for systems of height (left to right) 60,48,36,30,24,18,15,12,and 9. 90
6.4 Comparison of calculation of a global quantity at various grid sizes for the cubic spline (upper set) and centered difference method (lower set). The cubic spline method, for a given resolution has less error. 91
6.5 Comparison of calculation of a local quantity at various grid sizes for the cubic spline (lower set) and finite difference method (upper set). The cubic spline has a faster convergence rate than does the finite difference method. 92

7.1 Velocity, temperature, and density fields for steady compressible convection at $Ra = 20,000$. 97
7.2 Coexisting solutions (2-, 4-, and 6-roll) for compressible convection at $Ra = 40,000$. 99
7.3 Internal entropy production density for coexisting solutions (2-, 4-, and 6-roll) at $Ra = 40,000$. 105

8.1 Sinusoidal time dependence of the Nusselt number for harmonic convection. 109
8.2 Experimental results for convecting a. liquid mercury and b. gaseous helium. 111
8.3 Simulation results showing the drop in Nusselt number at the transition from harmonic to chaotic convection. 113
8.4 Nusselt number for harmonic and chaotic flows indicates a dual-morphology region where harmonic and chaotic flows may coexist. 114
8.5 Temperature contours for harmonic and chaotic flows at $Ra = 200,000$. The time sequence appearing on the left shows the flow for "harmonic" convection. 115
8.6 Dimensionless internal entropy production for the harmonic and chaotic flows. 117
8.7 Maximum Lyapunov exponent is a measure of the rate that a perturbed system diverges from a reference trajectory. 119
8.8 Dimensionless maximum Lyapunov exponent for chaotic and harmonic flow. Chaotic flows lose phase space information at a greater rate. 121
8.9 Linear sum of the dimensionless internal entropy production and dimensionless maximum Lyapunov exponent for the harmonic and chaotic flow morphologies. 122

9.1 Velocity, temperature, and vorticity fields for two-dimensional turbulent convection at $Ra = 1.5 \times 10^8$. 124
9.2 Diagram for the classic 1/3 scaling law theory 126

9.3 Dimensionless heat flux from turbulent convection as a function of the Rayleigh number. This suggests the existence of hard turbulence in two-dimensional compressible fluids. 127
9.4 Linear combination of the dimensionless entropy rates for turbulent convection as a function of the Rayleigh number. 128

A.1 Diagram of convecting double pendulum. The lower half of the cell is shown. A diagram of the thermally expanding mass is shown on the left. 147
A.2 Pendulum masses 1.) absorb heat at a higher background temperature 2.) expand, rise and 3.) release heat at a lower background temperature. . . 149
A.3 Entropy production as the convecting double pendulum is driven away from equilibrium. 150
A.4 Stroboscopic trajectory of the convecting double pendulum 152

List of Tables

2.1 Critical Rayleigh number and aspect ratio for the onset of convection for different boundary types. 21

7.1 Comparison of properties for 2-, 4-, and 6-roll patterns at $Ra = 40,000$. 102
7.2 Maximum internal entropy production density for the 2-, 4-, and 6-roll solutions for $Ra = 40,000$. 106

Part I

Motivation and Definitions

Chapter 1

Introduction

A numerical method for solving the full Navier-Stokes equations so as to simulate convection and convective turbulence in compressible fluids is described, analyzed, and validated by comparison with results from experiments and other computational work in this field. This method is then used to investigate some interesting aspects of convection such as coexisting steady states, a dual-morphology region where harmonic and chaotic flows coexist, and some properties of two-dimensional compressible turbulent convection.

1.1 Overview

This dissertation is divided into four sections – an introduction (Chapters 1-3) that provides a motivation for the work and defines terms used in later sections; a description of the cubic spline collocation method, along with analysis and validation (Chapters 4-6); a discussion of the application of the method to investigate convection and convective

turbulence (Chapters 7-9); and the conclusions (Chapter 10).

The remainder of this Chapter describes some historical work on convection, including a discussion of the Oberbeck-Boussinesq approximation. This is followed by a discussion of modern investigations, including those involving computer simulations.

Chapter 2 defines types of thermal convection, from steady to turbulent, and Fourier conduction – the simplest example of heat transport. Examples of convection in nature are discussed, from atmospheric to stellar, along with their physical parameters such as length scale, temperatures, and viscosities to show how convective processes are common among vastly different systems.

Chapter 3 develops the mathematical framework needed to describe convection in more detail. The properties of the fluid and its surroundings are discussed in order to define rigorously the Rayleigh number – the dimensionless parameter that quantifies how far from equilibrium a convecting system is. Other important dimensionless quantities are also defined.

Chapter 4 describes the "dual-staggered" discretization used for the method. The "dual"-part refers to the fact that the mass density is updated on one grid (zone-centered) while the energy and momentum are updated on another. The "staggered"-part refers to the fact that the flux of the conserved quantities are updated at the midpoints in each direction to avoid a decoupling of the even and odd-numbered nodes (the even/odd instability).

Chapter 5 describes the cubic spline collocation method. Similar methods, such as

centered finite difference, and Padé schemes are also described. The cubic spline method is analyzed to show that the truncation error of the midpoint interpolation *and* midpoint differentiation is of the order h^4, where h is the grid spacing. The method is also shown to be strictly conservative and to require relatively low computational effort (the computational effort scales linearly with the number of nodes used for the simulation).

Chapter 6 describes simulations of steady and time-dependent convection used to validate the numerical method. Results of these simulations agree with widely accepted results from experiments and other simulations reported in the literature.

Chapter 7 discusses properties of steady convection including an investigation of co-existing states in steady convection. Two-, four-, and six-roll solutions are generated and compared for the same Rayleigh number.

Chapter 8 details an investigation of intermediate-Rayleigh-number time-dependent convection including harmonic and chaotic flows. Here, a hysteresis exists that connects two different flow morphologies, possible for the same Rayleigh number. The simpler of the flows has two characteristic frequencies, the rotation frequency of the convecting rolls and the vertical oscillation frequency of the rolls. Observables, such as the heat flux, have a simple-periodic (harmonic) time dependence. The more complex (chaotic) flow has at least one additional characteristic frequency – the horizontal frequency of the cold, downward- and the warm, upward-flowing plumes. Observables of this latter flow have a broad-band frequency distribution.

The two flow morphologies, at the same Rayleigh number, have different rates of en-

tropy production and different Lyapunov exponents. The simpler "harmonic" flow transports more heat (produces entropy at a greater rate) whereas the more complex "chaotic" flow has a larger maximum Lyapunov exponent (corresponding to a larger rate of phase-space information loss). A linear combination of these two rates is invariant for the two flow morphologies over the entire range of Rayleigh numbers for which the flows coexist. This suggests a relation between the two rates near the onset of convective turbulence.

Chapter 9 discusses an investigation into convective turbulence. For these high-Rayleigh-number flows, a scaling relation exists relating the dimensionless heat flux to the Rayleigh number by the well-known two-seventh's power law. Other characteristics are observed such as vorticity on many length scales, lack of symmetry between the cold, downward- and the warm, upward-flowing thermal plumes, and an exponential-like temperature histogram of fixed points near the center of the cell.

Chapter 10 summarizes the conclusions about the numerical method and the investigation of convection.

Appendix A describes a simple model of a double pendulum, with thermally-conducting and thermally-expanding masses immersed in a constant-gradient temperature field, that transport heat in a way that is analogous to buoyancy-driven convection. A Rayleigh-number-like order parameter is defined for this system. The additional entropy produced by the convecting double pendulum is calculated as a function of the order parameter as the system is driven from regular to chaotic motion.

The intent in studying this model is to characterize the entropy production rate and the

Lyapunov spectrum as this simple, non-equilibrium system is driven into chaos. The results are compared to those of the more complex simulations of a compressible fluid, convecting heat in two dimensions. The simulations suggest that within a range of Rayleigh numbers time-averaged entropy production is related to the Kolmogorov-entropy (the rate at which phase-space information is lost) for this convecting system.

Appendix B provides the FORTRAN code for the Navier-Stokes solvers and utilities that are used to determine the maximum Lyapunov exponent and the internal entropy production.

Appendix C provides MATHEMATICA scripts for plotting and analysis.

1.2 Review of previous work

A discussion is given here of some historical highlights in the investigation of convective processes. The work of Bénard and Rayleigh, who laid the foundation of what is now known as the "Rayleigh-Bénard" problem, is outlined. The Oberbeck-Boussinesq approximation was used by Rayleigh and is still used by some researchers today to simplify the equations governing thermal convection. Its derivation and assumptions are discussed as historical background. Some current topics of investigation and their history is then discussed and related to the work in this dissertation.

1.2.1 Early work

Henri Bénard, for his PhD thesis work, did the first careful experiments with convecting fluids around the turn of the century. This work, appearing also in two subsequent publications in 1900 [1] and 1901 [2] describes an apparatus (see Fig.1.1) in which a thin layer of fluid is situated on a level metallic plate maintained at a constant temperature. The fluid is in free contact with the air above which is at a temperature lower than the metallic plate. He used a variety of liquids with different properties to investigate the rôle of viscosity on the flow characteristics of the fluid. Bénard's most important contribution, however, was the observation that at a critical plate temperature, for each fluid, a cellular structure (see Section 2.2.4) spontaneously appears. He noticed that this cellular structure was made up of small steady fluid currents, "*le régime permanent*", where the fluid ascended in the center and descended at the boundaries of adjoining cells.

Lord Rayleigh (John William Strutt), 1904 Nobel Laureate [1], laid out the groundwork for theoretical investigations into the onset of thermal convection. In his seminal paper in 1916 [3], he treated the "Bénard problem" as a linear stability problem. He used a linear approximation of the Navier-Stokes equations, originally developed by Oberbeck in 1879 [4], but attributed to Boussinesq's 1903 work [5]. Rayleigh, for practical reasons, considered a convecting system with two free boundaries at constant temperature. Al-

[1] 1904 Nobel Laureate in Physics "for his investigations of the densities of the most important gases and for his discovery of argon in connection with these studies"

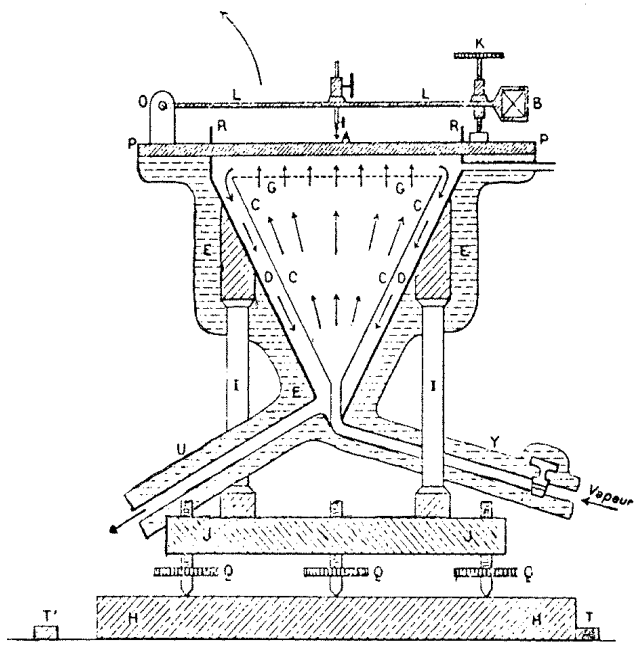


Figure 1.1: Bénard's original experimental apparatus

though these conditions do not model a true experimental system, the assumptions did allow Rayleigh to make a reasonable estimate of the conditions for the onset of convective instability.

Rayleigh's work was continued by Jeffreys in 1926 [6] and 1928 [7] who solved the problem for rigid-rigid and rigid-free boundaries and determined the critical "Rayleigh number" for each. The Rayleigh number, Ra, is a dimensionless quantity that depends on the temperature difference, length scale, viscosity, conductivity, and expansivity. (The derivation of the Rayleigh number is presented in Chapter 3). This dimensionless parameter, used by Rayleigh and Jeffreys, was first *called* the "Rayleigh number" by Sutton in 1961 [8]

Chandrasekhar (1961) [9] completed the analysis of the linear theory of thermal con-

vective instability. Again, the Oberbeck-Boussinesq approximation was employed with various boundary conditions. He analyzed the problem of a fluid heated from below in the presence of a shear, in a rotating system, and in the presence of a magnetic field – all using the linear, incompressible approximation of the Navier-Stokes equations.

1.2.2 Oberbeck-Boussinesq approximation

It may seem somewhat odd to model buoyancy-driven convection with a set of equations that assumes constant density, but the Oberbeck-Boussinesq approximation has afforded analysis of a complex problem and has allowed useful insight. Because of its historical significance, a derivation of the Oberbeck-Boussinesq equations is outlined here.

The derivation of the Oberbeck-Boussinesq approximation starts with the incompressible Navier-Stokes equations. The variations in the density ρ are neglected except insofar as they are used in a buoyancy term,

$$\begin{aligned} \nabla \cdot \vec{u} &= 0 \\ \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} &= \nu \nabla^2 \vec{u} - \frac{1}{\rho_0} \nabla p + \frac{\rho}{\rho_0} \vec{g} \\ \frac{\partial T}{\partial t} + \vec{u} \cdot \nabla T &= D_T \nabla^2 T, \end{aligned} \tag{1.1}$$

where the transport coefficients and thermal expansion coefficient are defined in terms of

this constant density

$$\nu \equiv \frac{\eta}{\rho_0}; \quad D_T \equiv \frac{\kappa}{c\rho_0}. \tag{1.2}$$

The variation in the density is linearized with the temperature variation $(T - T_0)$ by the thermal expansion coefficient,

$$\rho = \rho_0(1 - \alpha(T - T_0)); \qquad \alpha \equiv -\frac{1}{\rho_0}\left(\frac{\partial \rho}{\partial T}\right)_p, \tag{1.3}$$

where ρ_0 and T_0 are the density and temperature at $z = 0$.

Now, consider a static (quiescent) state with a constant temperature gradient, $\beta \equiv -\frac{\partial T}{\partial z}$,

$$\vec{u} = 0; \quad T = T(z) = T_0 - \beta z; \quad \rho = \rho_0(1 - \alpha(T - T_0)) = \rho_0(1 + \alpha\beta z). \tag{1.4}$$

The scalar pressure is related only to the gravitational force, which acts in the direction opposite to the heat flow,

$$\frac{\partial p}{\partial z} = -g\rho = -g\rho_0(1 + \alpha\beta z) \quad \Rightarrow p = p_0 - g\rho_0(z + \frac{1}{2}\alpha\beta z^2). \tag{1.5}$$

The thermal energy is transported solely by diffusion

$$\nabla^2 T = 0. \tag{1.6}$$

A new state with temperature, pressure, and density $\{T', p', \rho'\}$ and a non-zero velocity represents a perturbation of the static state,

$$\begin{aligned} T' &= T + \theta, \\ \rho' &= \rho_0(1 - \alpha(T' - T_0)) = \rho_0(1 + \alpha(\beta z - \theta)) \\ p' &= -g\rho_0(z + \alpha(\frac{1}{2}\beta z^2 - \theta z)) = p + g\rho_0\theta\alpha z, \end{aligned} \tag{1.7}$$

where θ is the temperature perturbation.

We can now solve the incompressible Navier-Stokes equations (1.1) for the perturbed state. The continuity equation, in terms of velocity, u, of the perturbed system also assumes incompressible flow,

$$\nabla \cdot \vec{u} = 0 \tag{1.8}$$

The terms in the momentum equation with density and pressure are expressed in terms of the perturbation using 1.5,

$$\begin{aligned} \frac{1}{\rho_0}\left[\rho'\vec{g} - \nabla p'\right] &= \frac{1}{\rho_0}\left[\vec{g}\rho_0(1 + \alpha(\beta z - \theta) - \nabla p - \nabla \delta p\right] \\ &= \frac{1}{\rho_0}\left[(\vec{g}\rho - \nabla p) - \alpha\vec{g}\rho_0\theta - \nabla\delta p\right] \\ &= -\alpha\vec{g}\theta - \nabla\frac{\delta p}{\rho_0}. \end{aligned} \tag{1.9}$$

The thermal energy equation for the temperature of the perturbed system reduces to an

equation for the perturbation temperature using 1.6,

$$\begin{aligned} \frac{\partial T'}{\partial t} + \vec{u} \cdot \nabla T' &= D_T \nabla^2 T' \\ \Rightarrow \frac{\partial T}{\partial t} + \frac{\partial \theta}{\partial t} + \vec{u} \cdot \nabla (T + \theta) &= D_T \nabla^2 (T + \theta) \\ \Rightarrow \frac{\partial \theta}{\partial t} + \vec{u} \cdot \nabla T + \vec{u} \cdot \nabla (T_0 - \beta z) &= D_T \nabla^2 \theta \end{aligned} \tag{1.10}$$

Using 1.9, 1.8, and 1.10, we arrive at the Oberbeck-Boussinesq equations in terms of the temperature perturbation,

$$\begin{aligned} \nabla \cdot \vec{u} &= 0 \\ \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} &= \nu \nabla^2 \vec{u} - \nabla \frac{\delta P}{\rho_0} - \alpha \theta \vec{g} \\ \frac{\partial \theta}{\partial t} + \vec{u} \cdot \nabla \theta &= D_T \nabla^2 \theta + \vec{u} \cdot \vec{\beta} \end{aligned} \tag{1.11}$$

1.2.3 Modern work

The term "modern" is used here to describe work on open questions that are currently under investigation. Most of this work involves the use of or is somehow influenced by computer simulation. Computers allow sufficiently accurate "thought" experiments of systems that are too complex for conventional analysis. In general, simulations of convective processes are based on either Oberbeck-Boussinesq approximation, Navier-Stokes equations, or *ab initio* methods including molecular dynamics [10–12], lattice gases [13], and smooth particle applied mechanics [14, 15]. Two areas of current research in ther-

mal convection are 1.) understanding the transitions to different convecting morphologies and 2.) understanding the heat transport in turbulent convection. Additionally, the need for accurate, efficient, and scalable numerical methods for the simulation of convective processes is motivating much research.

Transitions in convective flows

The nature of flow transitions, such as the transition from laminar to turbulent flow, has been questioned for ages. Benard's experiments to study the transition to cellular convection only mark the beginning of careful and controlled observation of convective processes. Experimental observations of the discrete transitions in convecting systems leading to turbulence were reported by Malkus (1954) [16]. Distilled water and acetone were allowed to convect with very high Rayleigh numbers (up to 10^{10}, well in the turbulent regime) while the total amount of transported heat was measured. Six discrete transitions were observed, each indicated by a change in the slope of the heat transport - Rayleigh number relation. This was later confirmed and renumbered by Willis and Deardorff (1967) [17]. Krishnamurti (1970) did very detailed experiments to characterize the transition from two-dimensional to three-dimensional steady flow [18] and to characterize the onset of time-dependent flow [19]. He showed that at a fixed point a time periodic oscillation develops. A discrete change in slope of the heat flux curve accompanies this transition. A hysteresis is also shown for the heat flux near this transition. Heslot (1987) [20] did experiments to map out transitions to turbulence in helium gas. His-

tograms of the temperature a points in the cell were used to define the regions of steady, oscillatory, chaotic, and soft and hard turbulent convection.

Clever and Busse [21–23] extended Chandrasekhar's work [9] with the non-linear Oberbeck-Boussinesq approximation and compared results to experimental results. They explored transitions between steady states and the onset of time-dependent convection.

For more complicated flows, computer simulations are needed. Goldhirsch (1989) [24] used simulations based on the Oberbeck-Boussinesq approximation to investigate steady and time-dependent flows. Although this method agrees well with experiments for steady and periodic flows, there is a surprising absence of chaotic or even quasi-periodic flows for low-Prandtl-numbered fluids. Rapaport (1991) [12] employed molecular dynamics methods to show that the onset of time-dependent convection results from these *ab initio* assumptions. The periodic flow that develops has a frequency close to that predicted by Goldhirsch. Deluca (1990) [25] and Werne (1991 and 1993) [26, 27] were able to successfully employ the Oberbeck-Boussinesq approximation to simulate and study the transition to two-dimensional convective turbulence.

Simulations that reveal coexisting flow morphologies for steady convection in compressible fluids have been reported by this author [15] and are discussed in Chapter 7. A region in which periodic and chaotic flows may coexist is observed near the transition for compressible fluids. This is reported by this author [28, 29] and is discussed in Chapter 8. A characterization of turbulent convection for a two-dimensional compressible fluid is also given by this author [29] and discussed in Chapter 9.

Heat transport in turbulent convection

Convective turbulence is an area of research that has many unanswered questions. The scaling relation between the Rayleigh number and the dimensionless heat flux has been the subject of much attention. Prandtl (1932) [30] used similarity arguments to estimate the velocity of a convecting fluid allowing to free fall. Priestley (1954) [31] used these arguments to estimate that the dimensionless heat flux for an equivalent system scales as the $\frac{1}{3}$ power of the Rayleigh number. Malkus (1954) [16] gave an analysis of the turbulent regime in terms of marginal stability of the mean flow to improve the estimates of the temperature dependence of the interior. This approach allows the derivation of the $\frac{1}{3}$ power law without similarity assumptions. Building on this approach, Howard (1963) [32] used similarity arguments based on a marginal stability of the boundary layers (see Chapter 9) to show this same power-law relation. Although these independent approaches predict a $\frac{1}{3}$ power law relation, experimental results do not agree. Heslot, Castaing and Libchaber (1987) [20] analyzed experimental data for convecting helium gas and observed that within the region of well-developed convective turbulence, the dimensionless heat flux scales as the $\frac{2}{7}$ power of the Rayleigh number. They referred to this region as "hard turbulence". They later [33] went on to argue, based on observations, that the shear wind induced by the large scale convection current stabilizes the viscous boundary layer. This allows for a slightly larger boundary layer, reducing the exponent in the power-law derivation of Howard.

She (1989) [34] suggested that thermal turbulence may have an inertial subrange quite

different from a Kolmogorovian freely cascading inertial range, leading to a departure from the classical $\frac{1}{3}$ law. This, however, has not been supported by data from experiment or simulation.

Shraiman and Siggia (1990) [35] deduced the $\frac{2}{7}$ power law from the presence of a mean flow and the nesting of a thermal boundary layer within a viscous one. Their assumption is that the heat flux into the boundary layer is carried completely by a mean flow. Xin (1997) [36] used a light scattering technique to measure the velocity profiles in convecting water to show that the thermal boundary layer is nested entirely within the viscous layer for turbulent convection. This confirms the theoretical predictions of Shraiman and Siggia. Naert *et alii* (1997) [37], however, did high-Rayleigh-number experiments using Mercury, a low-Prandtl-number fluid showing that the $\frac{2}{7}$ power law persists after the inversion of the thermal and viscous boundary layers. This disputes a basic assumption of Shraiman and Siggia. Werne *et alii* (1991) [26] used simulations based on the Oberbeck-Boussinesq approximation to show that "hard turbulence" exists in two dimensions. They reported that the convective part of the dimensionless heat flux $(Nu - 1)$ develops a $\frac{2}{7}$ power-law dependence on the Rayleigh number for two-dimensional convective turbulence. In a later paper, DeLuca, Werne, Rosner, and Cattaneo (1990) [25] reported that simulations of two-dimensional turbulence, based on Oberbeck-Boussinesq approximation, suggest that a substantial part of the heat is transported by hot and cold plumes. This disputes the assumption of Shraiman and Siggia that the heat flux into the boundary layer is carried completely by a mean flow. Ching (1997) [38] used analysis based on the Oberbeck-

Boussinesq approximation to show that the $\frac{2}{7}$ power-law is a result of the nonuniform shear from these plumes.

Wu and Libchaber (1991) [39] proposed that in a "non-Boussinesq" regime of turbulent convection, the boundary layers on opposite sides of a cell adjust their length scales so that the mean hot and cold temperature fluctuations are equal in the center of the cell. Zhang *et alii* (1997) [40] confirmed this equality experimentally with glycerol. This experiment also reveals a systematic deviation in the $\frac{2}{7}$ power law relation (as seen in Chapter 9).

Most simulation and theory of this power-law relation are based on the Oberbeck-Boussinesq approximation. In Chapter 9, convective turbulence is discussed for a two-dimensional *compressible* fluid. The $\frac{2}{7}$ power-law is recovered in these simulations. A discussion of the dimensionless internal entropy production and a dimensionless measure of the rate at which phase space information is lost is presented for this hard turbulence regime.

Simulating compressible convection

Some of the earliest models of compressible convection came from the astrophysical community. The Oberbeck-Boussinesq approximation is sufficient for thin layers, where the variation in density can be neglected. The compressible effects in stars, however, are significant. Spiegel [41] used a first-order perturbation of the linear OB equations to determine the marginal stability for polytropic atmospheres. Advances in computational approaches for modeling compressible convection were made by Fröhlich and Gauthier

for two-dimensional systems. A Fourier-Chebyshev spectral method is used for a system with both slippery and rigid boundary conditions [42]. A Fourier expansion is used in the horizontal direction while Chebyshev polynomials are used in the vertical. An explicit Adams-Bashforth predictor-corrector numerical scheme is used for the time domain. Steady compressible flows were generated and analyzed [43, 42, 44] for both boundary types. With advances in large super computers, three-dimensional compressible systems could be modeled with sufficient resolution. Toomre, Cattaneo, Brummell, and Hurlburt used a hybrid finite-difference-pseudospectral method with a resolution of 96^3 grid points to model compressible convection. A set of four computer runs with varying Rayleigh and Prandtl numbers is reported in references [45–47].

The cubic spline collocation method used in this dissertation to simulate convective processes in compressible fluids is developed, analyzed, and validated in Chapters 4-6.

Chapter 2

Thermal convection

Thermal convection is the transport of thermal energy by fluid currents. These currents are induced by buoyancy forces arising from a temperature difference. The morphology of a convecting system may vary from simple, time-independent convecting rolls to highly irregular, both in space and time, turbulent convection. In a fluid, heat is transported by conduction (thermal diffusion) and by convection if the conditions are right. Convection is a naturally occurring process – evident in vastly different systems.

2.1 Heat transport in two dimensions

Thermal energy is transported in two dimensional fluids in different amounts and by different mechanisms, depending on the fluid properties and how far from equilibrium the system is. Slightly away from equilibrium, heat is typically transported by linear conduction. A highly nonequilibrium system, on the other hand, may transport heat in a

turbulent fashion. The types of thermal transport, in two-dimensional fluids, are outlined in the remainder of this Section.

2.1.1 Fourier conduction

If viscous forces are strong enough to deny fluid motion, thermal energy is transported from a heat source to a heat sink solely by thermal diffusion. This process is known as Fourier conduction as it is described by Fourier's law of heat conduction, $q_F = \kappa \nabla T$. The heat flux, q_F, is related to the local temperature gradient by the thermal conductivity coefficient, κ.

2.1.2 Rayleigh-Bénard instability

A fluid, like air, in the presence of a heat source tends to expand as it heats, thus lowering its density. This warmer, lighter fluid spontaneously starts to rise if the buoyant forces overcome the viscous constraints in a time shorter than the time it takes for the heat to diffuse through the surrounding fluid. As the fluid rises, it carries thermal energy away from the heat source faster than it diffuses away. Conversely, a fluid in the presence of a heat sink cools, condenses, and submerges if the buoyancy forces (acting downward in this case) overcome the viscous grip.

The instability of the quiescent state leading to a convecting state occurs at a critical value of the Rayleigh number. This is the well-known "Rayleigh-Bénard" instability. In the convective state, heat is transported more effectively. The Rayleigh number is

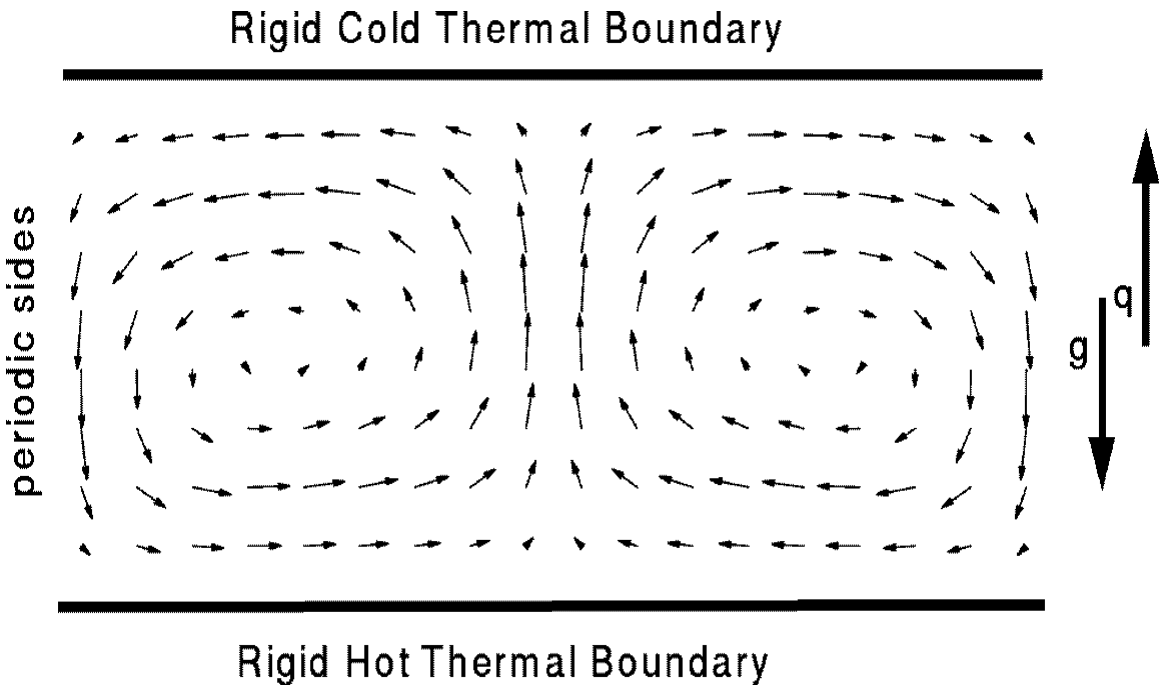


Figure 2.1: Velocity field of steady thermal convection of a compressible fluid enclosed between two rigid thermal boundaries.

a dimensionless temperature gradient that depends on the properties of the fluid, the magnitude of the body force (gravity), the length scale, and the temperature difference. The Rayleigh number is developed in Chapter 3 from dimensional analysis. For now, the Rayleigh number can be thought of as a parameter that quantifies how far a system is from equilibrium, much like the Reynolds number. It was shown by linear analysis that the critical Rayleigh number and critical cell aspect ratio *a* (ratio of convection cell width

Type of boundaries	Ra_c	a
Both free	657.5	1.41
Both rigid	1707.8	1.99
One rigid and one free	1100.7	1.71

Table 2.1: Critical Rayleigh number and aspect ratio for the onset of convection for different boundary types.

to height) depends on the nature of the upper and lower boundaries.

In simple systems, the convection state that results at this critical value is steady and has simple rolls. For a system with periodic side boundaries (representing an infinite width enclosure), the number of rolls is even to conserve angular momentum. For systems with rigid sides, an odd number of rolls is possible. The diameter of the rolls becomes equal to the height of the system, if the material is uniform and the system is isobaric. Steady convection is discussed further in Chapter 7.

2.1.3 Harmonic convection

At a much higher Rayleigh number, the system makes another transition from the steady convection to unsteady (time-dependent) convection. This transition was noted by Clever and Busse [22] for an incompressible fluid and later simulated by Rapaport [12] using molecular dynamics. This transition occurs at $Ra \approx 8 \times 10^4$ for our ideal gas fluid and marks the beginning of a periodic motion where the rolls oscillate up and down.

2.1.4 Chaotic and turbulent convection

Eventually, this periodic motion gives way to aperiodic convection as plumes start to mix the flow. It was pointed out by Heslot [20] that the chaotic flow at this point is distinctly different from fully-developed turbulence. This is usually referred to as "soft turbulence". Soft turbulence is characterized by a loss of time coherence while the space coherence persists. As the Rayleigh number is increased further, $Ra \approx 4 \times 10^7$, the

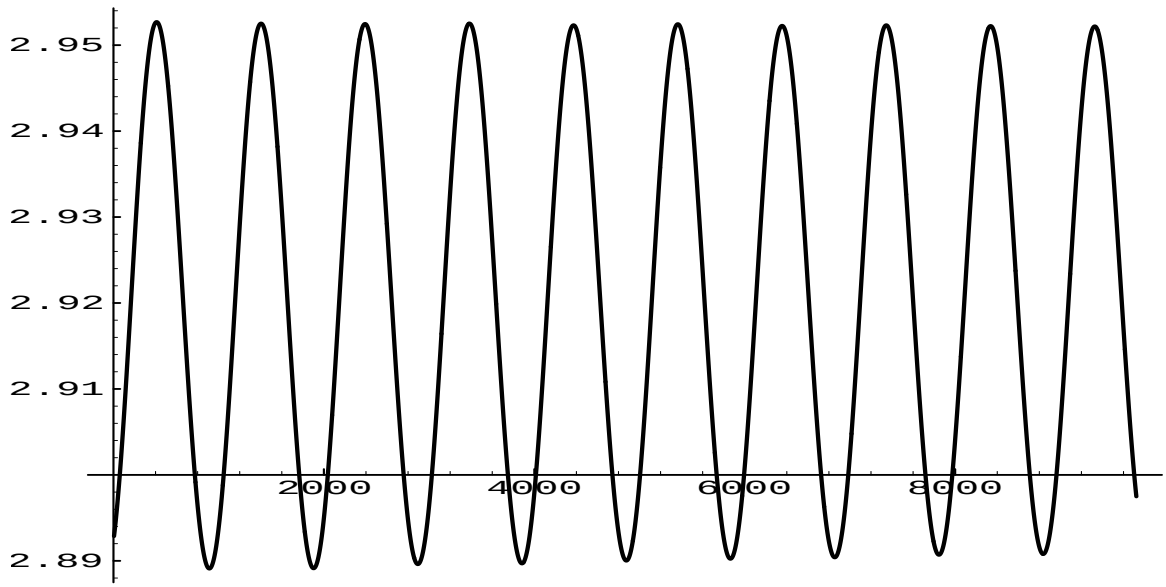


Figure 2.2: Dimensionless heat flux as a function of time for harmonic convection.

system finally makes a transition to "hard turbulence" where time and space coherence is lost. An interesting feature of this "hard turbulence" region that has been observed in experimental studies of convection using helium gas [33] is the existence of a scaling region where the convective part of the heat flux is related to the Rayleigh number by a two-seventh's power-law. This is different from the "classical" result [32] relating the heat flux to $Ra^{\frac{1}{3}}$ which is based on the assumption of marginal stability at a thermal boundary layer. Numerical simulations of incompressible Boussinesq fluids reveal a scaling relation close to the experimental one for Rayleigh numbers between 10^8 and 10^{15} [25, 27]. Harmonic and chaotic convection are discussed in Chapter 8. Turbulent convection is discussed in Chapter 9.

2.2 Examples of Convection

Many examples of convection, both laminar and turbulent, exist in nature and are part of our everyday lives. They range greatly in temperature scale, length scale, and scale of the various material properties. Examples of convection in Nature and laboratories are discussed in the remainder of this Section.

2.2.1 Atmospheric convection

Convective weather patterns affect us daily. As solar radiation heats the Earth's surface, the air near the surface heats, expands, and rises. Similarly, air in the upper atmosphere losses heat, condenses, and falls – completing the convective cycle. This extremely simplified view does not take into account moisture, photochemistry, rotation, diurnal effects, or jet streams. This simplified view of "dry" convection, however, does give a rough model of the dynamics of planetary atmospheres. The fact that the atmosphere has a very low viscosity ($\nu \sim 1.5 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$), a low heat transport coefficient, and a large characteristic length (the troposphere starts about $10\,\mathrm{km}$ above the surface) leads to a system that is quite convectively unstable. In fact, given that the gravitational acceleration is $9.8\,\mathrm{m\,s^{-2}}$, the mean temperature is $300\,\mathrm{K}$, and assuming that air has a thermal expansion similar to an ideal gas, it can be shown that if the temperature difference between the surface and the air, say at $1\,\mathrm{km}$, is on the order of $10^{-15}\,\mathrm{K}$, convective instability will dominate. The Rayleigh number for atmospheric convection can be as high as 10^{17} to 10^{20}.

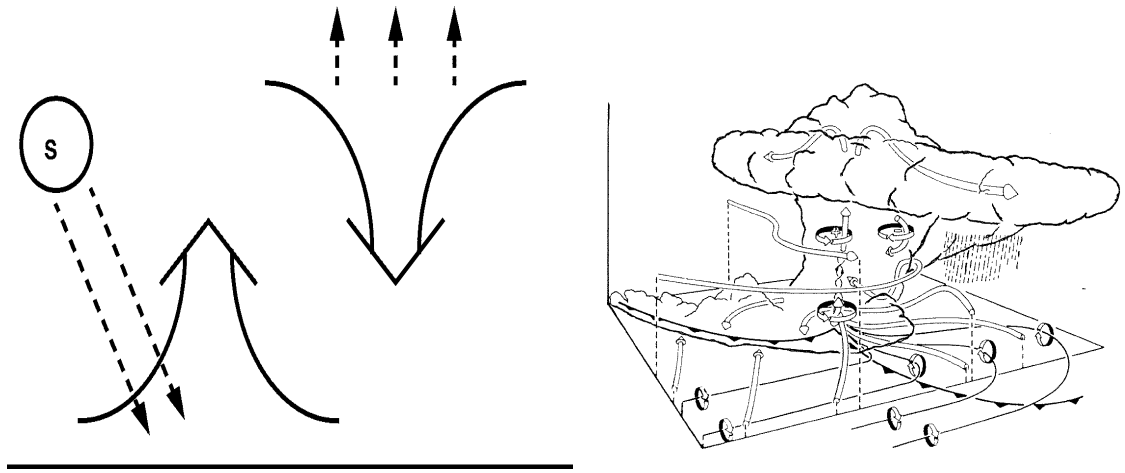


Figure 2.3: Diagrams of ideal convection (left) and realistic moist-air convection (right) in the atmosphere. [*From Klemp (1987).*]

In reality, moisture adds thermodynamic complexity to the convection dynamics by introducing a variety of new dynamical processes that have no analog in dry convection [48, 49]. Not only has moist, unsaturated air a greater heat capacity than dry air, but absorbed water can go through phase changes as temperature and pressure change. Condensation or "deposition" (transformation of water vapor to ice) releases heat locally at a greater rate than molecular diffusion would allow without the phase change. The cloud droplets may coalesce and grow to precipitation-sized particles and fall. The falling of water droplets (rain) forces a convective-like flow as the air below is displaced.

2.2.2 Mantle convection

Mantle convection is a process that has a great influence on the shape of the earth, yet goes mostly unnoticed – that is, except for an occasional earthquake or volcanic eruption. Convection dynamics are responsible for mountain building, plate tectonics, fluctuations

in the Earth's magnetic field, and most of the global heat loss. The global heat loss, due to mantle convection, is estimated to be $44.2 \times 10^{12}\,\mathrm{Watts}$ [50].

The mantle is made up of molten material such as magnesium-iron silicates. At high temperatures and pressures these act as a high-viscosity plastic. The reference viscosity is this molten material is estimated at $10^{21}\,\mathrm{Pa} \cdot \mathrm{s}$ [51, 52] (the actual viscosity varies greatly with temperature and pressure). The main characteristic that distinguishes mantle convection from the convection one sees in the laboratory is the time scale. The speed of the creeping flow of the mantle is inferred from the speed of the Earth's tectonic plates, since these are linked to the down-welling cold boundary layer. Typical plate speeds are about $10\,\mathrm{cm/year}$ and the mantle depth is about $2900\,\mathrm{km}$ giving the estimated "transit" time for a plume of about 29 million years. For reference, the age of the Earth is 4.5 billion years. Since the rates were probably higher in the past when the Earth's interior was hotter, it has been estimated that there have been hundreds of overturns during Earth's history [53].

Mantle convection serves to transport heat from the outer core (which is primarily molten iron) to the Earth's surface. The temperature difference, estimated from the high-pressure melting temperature of FeO and FeS, is around 750 K. The thermal expansivity, thermal diffusivity, and density estimated from laboratory measurements at high temperature and pressure are $3 \times 10^{-5}\,\mathrm{K}^{-1}$, $10^{-6}\,\mathrm{m}^2\,\mathrm{s}^{-1}$, and $4 \times 10^3\,\mathrm{kg} \cdot \mathrm{m}^{-3}$ [51]. This leads to a Rayleigh number on the order of 10^7. This is within the chaotic or turbulent range for most fluids. Mantle convection is assumed to be chaotic given on the distribution of

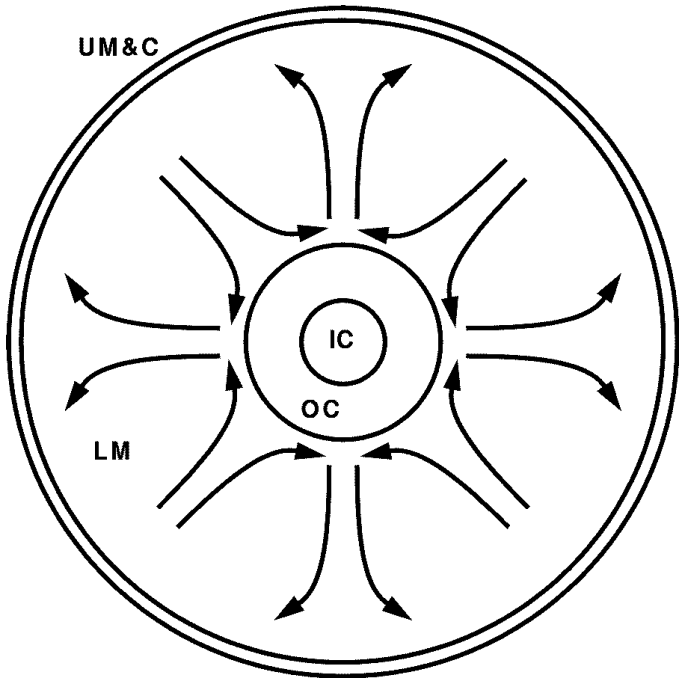


Figure 2.4: Cross-section diagram of the Earth showing the UM&C: upper mantle and crust ($0-670\,\mathrm{km}$), LM: convecting lower mantle ($670-2,890\,\mathrm{km}$), OC: liquid outer core ($2,890-5,150\,\mathrm{km}$), and IC: solid inner core ($5,150-6,370\,\mathrm{km}$).

trace elements in the upper mantle [54].

2.2.3 Stellar convection

Convection in stars, like the Sun, is the main mechanism through which heat, generated by nuclear reactions at the core, is transported to the surface. Convection occurs in the outer 30% of the solar radius (roughly $200,000\,\mathrm{km}$) across which the temperature drops from $2,000,000\,\mathrm{K}$ to about $6,000\,\mathrm{K}$ at the solar surface. The density ratio of the compressed material near the bottom of the convection rolls to the outer surface is 10^7. The Rayleigh number associated with solar convection is near 10^{24}.

Solar granulation is a term used to refer to the ever-present convection rolls observed

on the Sun's surface (see Figure 2.5). They are grain-like in shape and about $5,000\,\mathrm{km}$ across. The "turn-over" time for these cells is about ten minutes.

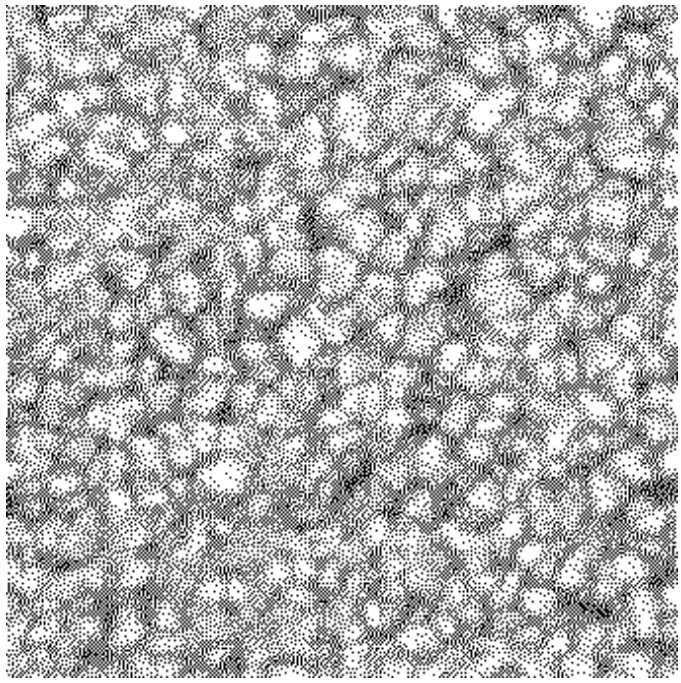


Figure 2.5: Solar granulation showing very small-scale turbulent motions (from La Palma observatory)

2.2.4 Bénard's and other laboratory experiments

Laboratory experiments, including those of Bénard, are carried out to study thermal convection in a controlled environment. Obviously, the extreme conditions (temperatures, pressures, viscosities, and length scales) that exist in atmospheric, mantle, and stellar convection would be difficult to reproduce in the laboratory. However, many physical aspects of convection may be studied in a more moderate environment with an appropriate

choice of materials and laboratory conditions.

Bénard, for example, studied convection in a very thin layer of fluid (about $1\,\mathrm{mm}$) on a metallic plate which was heated uniformly by steam ($T_H \leq 100^\circ\,\mathrm{C}$) while the upper free surface was cooled by ambient air ($T_L \approx 20^\circ\,\mathrm{C}$). Some of the higher-viscosity fluids that

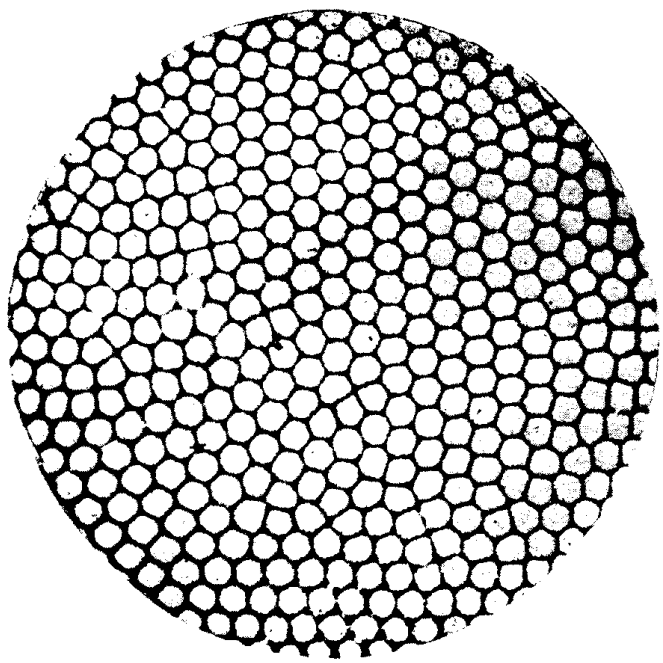


Figure 2.6: Bénard cells in spermaceti. A reproduction from one of Bénard's original photographs. [*From Chandrasekhar (1961).*]

he used included spermaceti (oil from a sperm whale) and melted paraffin. Spermaceti is rigid at room temperature, but melts at $46^\circ\,\mathrm{C}$, and has a low thermal conductivity. The result was the formation of an array of steady hexagonal-shaped convection cells with a cross section of about $1 - 1.5\,\mathrm{mm}$ with a turnover time (the time for a fluid volume to travel to the top and back to the bottom of a convecting cell) of about ten seconds. The

cellular structure resembles that of the small-scale stellar convection (compare figures 2.5 and 2.6), despite great differences in length scale, temperature, and viscosity.

Many advances in experimental technique have allowed more precise measurement and characterization of convective processes. The bolometer is the most sensitive detector of broad-band electromagnetic radiation at wavelengths between 250 m and 5 mm. The responsiveness of this detector allows the measurement of the temperature histogram of the convecting fluid at a point in the cell. The temperature histogram of a point in the center and near the bottom of a container of gaseous helium allowed Heslot *et alii* [20] to characterize the domains of the various convection types – laminar, oscillatory, chaotic, and turbulent. To visualize the flow, a shadowgraph technique was used by Zhang and coworkers [40]. Using a collimating lens, a nearly parallel light source was sent through

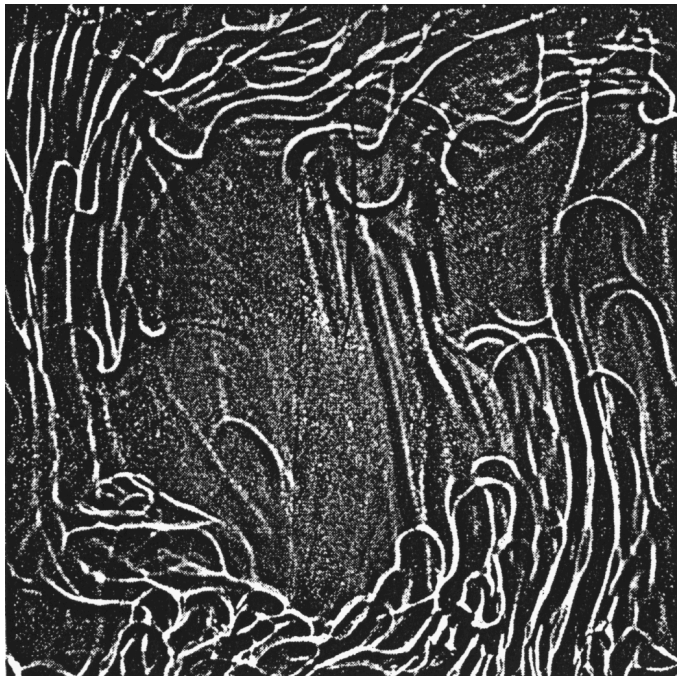


Figure 2.7: Shadowgraph of convecting glycerol at $Ra = 2.3 \times 10^8$. [*From Zhang (1997).*]

the convection cell. Shadows, resulting from refraction of the fluid, outline the thermal plumes. See Figure 2.7. The need to examine the convective properties of a low-Prandtl-number fluid inspired Naert and his colleagues to use liquid mercury. The objective of this study was to reach a high-Rayleigh-number point at which the viscous and thermal boundaries invert. Rayleigh numbers as high as 10^9 were obtained in this study.

Chapter 3

Factors influencing convection

The properties of the fluid transporting heat along with its boundary conditions determine whether convection will occur. The rate a which heat is transported and the type of convection that develops depends on how far the system is from equilibrium. Whether or not convection occurs and the nature of the flow if it does depends on the properties of the fluid and how far from equilibrium the system is. The amount of heat that is transported is also dependent on these factors.

3.1 Description of the fluid

Many of the physical properties of a fluid can be determined by the thermal and mechanical equation of state. These include its response to a change in the environment. The transport coefficients describe the local transport of energy and momentum. The nature of the simple two-dimensional fluid used here to study convective processes is

described in the remainder of this section.

3.1.1 Thermal and mechanical equation of state

To study natural convection and the transitions leading to turbulence, we establish a model of a simple, heat-conducting, viscous fluid enclosed between two thermal boundaries and in the presence of a body force. For simplicity, we assume a fluid with an ideal equation of state, where the equilibrium hydrodynamic scalar pressure is related to the density and temperature through the Boltzmann constant (per unit mass), $P_{eq} = \rho k_B T$. The thermal equation of state, $e = k_B T$, defines the internal energy per unit mass. With these equations, we can define some of the properties of the material. The sound speed for the fluid depends on the relationship between the pressure and the density. For the ideal gas equation of state,

$$c = \left(\frac{\partial P}{\partial \rho}\right)^{1/2} = (k_B T)^{1/2}. \tag{3.1}$$

The thermal expansion coefficient, or expansivity, is the rate a volume changes with the temperature,

$$\alpha \equiv \frac{1}{V}\left(\frac{\partial V}{\partial T}\right)_P = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_P = T^{-1}. \tag{3.2}$$

Also, the isothermal mechanical compressibility can be defined,

$$\beta \equiv -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_T = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial P}\right)_T = P^{-1}. \tag{3.3}$$

For convenience we also assume that transport coefficients – the heat conductivity (κ) and the shear viscosity (η) – are constant.

3.1.2 Isotropy and transport coefficients

Mechanical and thermal isotropy is a common simplifying assumption used in modeling fluids. It assumes that the mechanical and thermal properties of the fluid do not depend on the direction and requires only two parameters – one for shear deformations and one for volumetric deformations. For an isotropic fluid, the stress tensor can be written as

$$\overline{\overline{\sigma}} \equiv \lambda(\nabla \cdot \vec{u})\overline{\overline{I}} + \eta(\overline{\overline{\nabla u}} + \overline{\overline{\nabla u}}^t) - P_{eq}\overline{\overline{I}}, \tag{3.4}$$

where P_{eq} is the equilibrium pressure scalar, η and η_v are the shear and bulk viscosity. The term $\overline{\overline{\nabla u}}$ is the velocity gradient (a tensor) and $\overline{\overline{\nabla u}}^t$ is its transpose. In three dimensions, the velocity, its divergence and gradient are

$$\vec{u} \equiv \begin{pmatrix} u \\ v \\ w \end{pmatrix}, \qquad \nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}, \qquad \overline{\overline{\nabla u}} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{bmatrix}. \tag{3.5}$$

Considering the trace of the stress tensor,

$$\begin{aligned} Tr(\sigma) &= 3\lambda(\nabla \cdot \vec{u}) + 2\eta(\nabla \cdot \vec{u}) - 3P_{eq} \\ \Rightarrow \frac{1}{3}Tr(\sigma) + P_{eq} &= (\lambda + \frac{2}{3}\eta)\nabla \cdot \vec{u} \\ &\equiv \eta_{v(3d)}\nabla \cdot \vec{u} \end{aligned} \tag{3.6}$$

we can relate the equilibrium pressure to a contribution from the volumetric deformation. This defines λ in terms of the shear and bulk viscosities in three dimensions,

$$\lambda = \eta_{v(3d)} - \frac{2}{3}\eta \tag{3.7}$$

In two dimensions, the velocity, its divergence and gradient are

$$\vec{u} \equiv \begin{pmatrix} u \\ \\ v \end{pmatrix}, \qquad \nabla \cdot \vec{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}, \qquad \overline{\overline{\nabla u}} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}. \tag{3.8}$$

Therefore, the trace of the stress tensor is

$$\begin{aligned} Tr(\sigma) &= 2\lambda(\nabla \cdot \vec{u}) + 2\eta(\nabla \cdot \vec{u}) - 2P_{eq} \\ \Rightarrow \frac{1}{2}Tr(\sigma) + P_{eq} &= (\lambda + \eta)\nabla \cdot \vec{u} \\ &\equiv \eta_{v(2d)}\nabla \cdot \vec{u}, \end{aligned} \tag{3.9}$$

which relates λ to the shear and bulk viscosities in two dimensions,

$$\lambda = \eta_{v(2d)} - \eta. \tag{3.10}$$

At this point, we can relate the shear viscosity with the viscosities associated with volumetric deformation in two and three dimensions,

$$\begin{aligned} \eta_{v(3d)} - \frac{2}{3}\eta &= \eta_{v(2d)} - \eta \\ \Rightarrow \eta_{v(2d)} &= \eta_{v(3d)} + \frac{1}{3}\eta \end{aligned} \tag{3.11}$$

Stoke's assumption further simplifies the isotropic assumption by setting the bulk viscosity to zero. If we are to use Stoke's assumption for simulations in two dimensions, we must decide whether the fluid is truly two-dimensional or just a slice of a three-dimensional fluid. For this study, we choose to represent a three-dimensional Stoke's fluid and do so by setting $\eta_{v(3d)} = 0$. This determines the value of the bulk viscosity for the two-dimensional constitutive equations,

$$\boxed{\eta_{v(2d)} = \frac{1}{3}\eta} \tag{3.12}$$

This assumption is quite reasonable for simulations of steady convection where the properties of the rolls, in three dimensions, are independent of the position along the axis of the rolls. Put another way, the two-dimensional simulations represent a slice of three-dimensional steady convection. For time-dependent convection, particularly for chaotic

and turbulent convection, the flow is not independent of the third direction. In this case, the choice of the bulk viscosity is not based on axial symmetry, but is still valid for the expansion of a small volume.

The numerical effect of adding the bulk viscosity in the two-dimensional simulations is typically quite small. In an ideal-gas simulation of Rayleigh-Bénard flow, at a Rayleigh number of 3600, the flow velocity increased by about one part in 1000 when the two-dimensional bulk viscosity was set equal to zero rather than to $\eta/3$. It is interesting, and possibly significant, that taking a two-dimensional reference system with vanishing bulk viscosity, would require a *negative* bulk viscosity in three dimensions, leading to catastrophic instabilities [15].

Fourier's law of heat conduction describes the linear transport of thermal energy where the heat flux is related to the temperature gradient by the heat conductivity coefficient,

$$\vec{q}_F = -\kappa \nabla T. \tag{3.13}$$

The conservation of thermal energy for a material of mass density ρ and heat capacity c relates the time rate of change for the temperature of a small volume to the divergence of the heat flux. For simple conduction,

$$\rho c \frac{\partial T}{\partial t} = -\nabla \cdot q_F = \nabla \cdot \kappa \nabla T. \tag{3.14}$$

For constant density, heat capacity and conductivity,

$$\frac{\partial T}{\partial t} = \frac{\kappa}{\rho c}\nabla^2 T = D_T \nabla^2 T \tag{3.15}$$

relating the thermal diffusion and conduction coefficients.

3.2 Rayleigh number

This system may be driven away from equilibrium by increasing the temperature gradient, decreasing the transport coefficients, or increasing the gravitational field. A dimensionless parameter that incorporates all of the possibilities was first derived by Rayleigh [3] and allows us to compare the convection of very different systems such as stellar and mantle convection. The Rayleigh number is defined as the ratio of the power per unit area from conduction to convection for a simple model of the system,

$$Ra \equiv \frac{|u|\rho c \Delta T}{\kappa \Delta T / L}. \tag{3.16}$$

The characteristic velocity is determined by balancing the viscous drag force with the buoyancy force per unit area for a volume element moving with the flow,

$$\eta \frac{|u|}{L} = g\rho\alpha\Delta T L. \tag{3.17}$$

By equating the thermal and potential energy change for a volume of fluid moving from the one boundary to the other, the magnitude of the body force can be defined by

$$\rho g L = k_B \Delta T, \tag{3.18}$$

thus simplifying the Rayleigh number,

$$\boxed{Ra = (\rho_0 c k_B) \frac{\alpha \Delta T^2 L^2}{\kappa \eta}.} \tag{3.19}$$

The convecting system has a characteristic length scale, velocity, and shear viscosity for which a Reynolds number may be estimated. The Reynolds number, like the Rayleigh number, is a dimensionless order parameter that describes how far a dynamical system is from equilibrium. The Reynolds number is a ratio of inertial to momentum diffusion rates,

$$Re \equiv \frac{|u| L}{\nu} = \frac{|u| L \rho_o}{\eta}. \tag{3.20}$$

For steady flows, the viscous drag force is balanced by the buoyancy force (Eq. 3.17). By balancing the potential and thermal energies (Eq. 3.18), we can express the characteristic

velocity in terms of the Reynolds number,

$$\begin{aligned}
\frac{\eta^2 Re}{\rho_0 L^2} &= k_B \alpha \Delta T^2 \\
\Rightarrow Re &= (\rho_0 k_B) \frac{\alpha \Delta T^2 L^2}{\eta^2} \\
&= (\rho_0 c k_B) \frac{\alpha \Delta T^2 L^2}{\eta \kappa} Pr^{-1} \\
&= Ra\ Pr^{-1} \qquad \text{(steady)},
\end{aligned} \tag{3.21}$$

where the Prandtl number is the ratio of the momentum to thermal diffusion times.

$$Pr \equiv \frac{\nu}{D_T} \equiv \frac{c\eta}{\kappa}. \tag{3.22}$$

For turbulent flows, the buoyancy force must be balanced by the fluid acceleration. The Froude number is the ratio of inertial to potential energies,

$$Fr \equiv \frac{u^2}{gL}. \tag{3.23}$$

Again, using Eq. 3.18 for g and Eq. 3.20 to relate the Reynolds number to the characteristic velocity,

$$\begin{aligned}
\frac{Re^2 \eta^2}{L^2 \rho_0^2} &= Fr\ \frac{k_B \Delta T}{\rho_0} \\
\Rightarrow Re^2 &= Fr\ (k_B \rho_0) \frac{L^2 \Delta T}{\eta^2} \\
&= Fr\ Pr^{-1} (\rho_0 c k_B) \frac{L^2 \Delta T}{\kappa \eta}
\end{aligned} \quad . \tag{3.24}$$

For an ideal gas, $\alpha = T^{-1}$. If the mean temperature is of the order of the temperature difference, then,

$$Re^2 \sim Fr\ Pr^{-1} Ra \qquad \text{(turbulent)}. \tag{3.25}$$

3.3 Nusselt number

The Nusselt number is the ratio of the total heat flux to the purely-conductive heat flux for an equivalent system (length scale, temperature difference, and conductivity).

$$Nu \equiv \frac{q_{tot}}{q_F} = \frac{q_{tot}}{-\kappa \nabla T}. \tag{3.26}$$

The total heat flux across a system is equal to the heat flux across its boundary wall,

$$\vec{q}_{tot} = \vec{q}_{wall} = -\kappa \lim_{x \to wall} \nabla T. \tag{3.27}$$

By calculating the heat flux near the wall, thermal energy entrained by the fluid flows may be neglected. Within the bulk fluid, the more complicated energy flux vector must

be calculated,

$$\begin{aligned}\vec{q}_{tot} &= -\kappa\nabla T - \vec{u}\cdot\overline{\overline{\sigma}} + \rho\vec{u}(e + \frac{1}{2}\vec{u}\cdot\vec{u}) \\ &= \left[-\kappa\frac{\partial T}{\partial x} - u\sigma_{xx} - v\sigma_{xy} + \rho u e + \frac{\rho u}{2}(u^2+v^2)\right]\hat{i} \\ &+ \left[-\kappa\frac{\partial T}{\partial y} - u\sigma_{xy} - v\sigma_{yy} + \rho v e + \frac{\rho v}{2}(u^2+v^2)\right]\hat{j}\end{aligned} \tag{3.28}$$

for a two-dimensional flow with velocity $\vec{u} \equiv u\hat{i} + v\hat{j}$. This expression for the heat flux approaches $\vec{q}_{wall}$ (3.27) as the velocity vanishes at the wall. The total flux across a line, parallel to the boundary is

$$Q_{tot} \equiv \oint_A \vec{q}_{tot}\cdot d\vec{A}. \tag{3.29}$$

Part II

Numerical Methods

Chapter 4

Discretization

Simulation of a physical system by digital computers requires the discretization of the appropriate equations. The equations, although based on some simplifying assumptions, must represent the essential underlying physics of the problem. Once the equations are determined, the system must be divided into smaller discrete parts (discretized) so that the computer, with its finite capacities, can store and operate on them. Improper discretization may lead to spurious answers or catastrophic instabilities.

4.1 Navier-Stokes equations

The Navier-Stokes equations describe the conservation of mass, momentum, and energy for a continuous medium that is either compressible or incompressible. The incompressibility assumption has been used by many researchers, as in the case of the Oberbeck-Boussinesq approach, because it greatly simplifies the equations and reduces the

computational effort needed. This assumption, however, fails to correctly model all physical responses of the system such as compression (sound) waves. The fully-compressible Navier-Stokes equations are used here to simulate buoyancy-driven thermal convection.

4.1.1 Eulerian and Lagrangian forms

These conservation laws may be expressed in either the "Lagrangian" frame (following the motion of the fluid) or the "Eulerian" frame (fixed or "lab"). By considering the definition of the "convective" or "material" time derivative,

$$\frac{Ds}{Dt} \equiv \dot{s} = \frac{\partial s}{\partial t} + \vec{u} \cdot \nabla s, \tag{4.1}$$

where $\vec{u}$ is the material flow velocity and s is an observed quantity, we can express the Navier-Stokes equations in both frames,

$$\begin{aligned} \dot{\rho} &= -\rho \nabla \cdot \vec{u} = \frac{\partial \rho}{\partial t} + \vec{u} \cdot \nabla \rho \\ \rho \dot{\vec{u}} &= \nabla \cdot \overline{\overline{\sigma}} + \rho \vec{g} = \rho \frac{\partial \vec{u}}{\partial t} + \rho \vec{u} \cdot \nabla \vec{u} \\ \rho \dot{e} &= \overline{\overline{\sigma}} : \nabla \vec{u} - \nabla \cdot \vec{q} = \rho \frac{\partial e}{\partial t} + \rho \vec{u} \cdot \nabla e. \end{aligned} \tag{4.2}$$

The constitutive equations describe the local comoving linear transport,

$$\vec{q} = -\kappa \nabla T; \qquad \overline{\overline{\sigma}} = \eta_v (\nabla \cdot \vec{u}) \overline{\overline{I}} + \eta (\nabla \vec{u} + \nabla \vec{u}^t) - P_{eq} \overline{\overline{I}} \tag{4.3}$$

where,

$\rho \equiv$ density	$\vec{u} \equiv$ velocity	$e \equiv$ internal energy per unit mass
$\vec{q} \equiv$ heat flux	$\vec{g} \equiv$ body force	$P_{eq} \equiv$ equilibrium pressure $= \rho e = \rho k_B T$
$\overline{\overline{\sigma}} \equiv$ stress tensor	$\eta \equiv$ shear viscosity	$\eta_v \equiv$ bulk viscosity .

4.1.2 Conservative (flux) form

The Navier-Stokes equations can be put into "conservative form":

$$\frac{\partial Q}{\partial t} + \nabla \cdot \vec{F} = S \tag{4.4}$$

where the time-rate-of-change of the conserved quantity Q is equal to the negative divergence of its flux $\vec{F}$ plus the source S. When put into discrete form, the difference equations conserve the quantities to machine accuracy by explicitly accounting for the change due to the fluxes.

In two dimensions, the conservative form of the Navier-Stokes equations is,

$$\boxed{\begin{aligned}
&\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \\
&\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2 - \sigma_{xx}) + \frac{\partial}{\partial y}(\rho uv - \sigma_{yx}) = \rho g_x, \\
&\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho uv - \sigma_{xy}) + \frac{\partial}{\partial y}(\rho v^2 - \sigma_{yy}) = \rho g_y, \\
&\frac{\partial}{\partial t}\left[\rho e + \frac{\rho}{2}(u^2+v^2)\right] + \frac{\partial}{\partial x}\left[q_x - u\sigma_{xx} - v\sigma_{xy} + \rho ue + \frac{\rho u}{2}(u^2+v^2)\right] + \\
&\qquad \frac{\partial}{\partial y}\left[q_y - u\sigma_{xy} - v\sigma_{yy} + \rho ve + \frac{\rho v}{2}(u^2+v^2)\right] = \rho(ug_x + vg_y),
\end{aligned}} \tag{4.5}$$

with,

$$\vec{q} \equiv -\kappa \nabla e; \quad \overline{\overline{\sigma}} \equiv [(\eta_v - \eta)\nabla \cdot \vec{u} - P_{eq}]\,\overline{\overline{I}} + \eta(\overline{\overline{\nabla u}} + \overline{\overline{\nabla u}}^t); \quad \vec{u} = u\hat{i} + v\hat{j}.$$

Again, the equilibrium pressure is determined by the equation of state

$$P_{eq} \equiv \rho k_B T.$$

4.2 Staggered grid

The solution of numerical approximation of differential equations depends on the way that the problem is discretized. Discretization is the representation of the continuous variables as a finite number of values, each with finite accuracy. For an Eulerian representation, these values are fixed in space and may sit on the nodes of a regularly-spaced grid. Improper discretization, however, may lead to spurious solutions.

4.2.1 Example: One-dimensional heat equation

The one-dimensional diffusion equation, expressed in flux form, is used as a simple example that demonstrates how improper discretization may lead to a spurious solution.

$$\frac{\partial T}{\partial t} = -\nabla \cdot \vec{q} = -\frac{\partial \vec{q}}{\partial x}; \qquad \vec{q} = -\nabla T = -\frac{\partial T}{\partial x}, \tag{4.6}$$

where k_B, ρ, and c_v are set equal to one for simplicity.

First, the continuous variables T and q are represented as discrete values on the nodes of a regular (evenly-spaced) one-dimensional grid. Next, the spatial derivatives at the nodes are approximated using a centered-difference,

$$\frac{\partial q}{\partial x}\Big|_0 \approx \frac{q_+ - q_-}{2\Delta x}; \qquad \frac{\partial T}{\partial x}\Big|_0 \approx \frac{T_+ - T_-}{2\Delta x}, \tag{4.7}$$

where the derivative is approximated by the difference of the neighboring values divided by the space between them. The grid spacing is set equal to one for simplicity. The boundaries are made periodic by allowing the end points to be neighbors. Starting with an initial condition $T = \{0, 0, A, 0\}$, the heat flux q is determined to be $\frac{A}{2}\{0, -1, 0, 1\}$. Next the time derivative of the temperature is determined to be $\frac{A}{2}\{1, 0, -1, 0\}$. This shows that the temperature of nodes 2 and 4 (the even nodes) will not change.

#	1	2	3	4
	—O————	O————	O————	O————
T:	0	0	A	0
q:	0	$-\frac{A}{2}$	0	$\frac{A}{2}$
$\dot{\mathrm{T}}$:	$\frac{A}{2}$	0	$-\frac{A}{2}$	0

Because the flux depends on the gradient of the conserved quantity and the centered-difference approximation of the gradient depends only on the value of the immediate neighbors, the time evolution of the even nodes becomes decoupled from that of the odd nodes. This is the "even/odd" numerical instability. It should be noted, however, that if

the total number of nodes is odd for a periodic grid, the even and odd nodes are coupled and the instability is avoided.

The following MATHEMATICA program iterates equations from an initial condition, $T = \{0, 0, A, 0\}$ with a time step $dt = 0.1$ on a simple periodic grid.

```
(* Simple grid for diffusion equation *)
nn=4; dt=0.1;
T=Table[0,{nn}]; T[[3]]=1; q=Table[0,{nn}];ntime=50;
pdat1=Table[0,{ntime}]; pdat2=Table[0,{ntime}];
pdat3=Table[0,{ntime}]; pdat4=Table[0,{ntime}];
pdat1[[1]]=T[[1]]; pdat2[[1]]=T[[2]];
pdat3[[1]]=T[[3]]; pdat4[[1]]=T[[4]];
For[j=2,j<=ntime,j++,
        q[[1]]= -(T[[2]]-T[[nn]])/2;
        For[i=2,i<=nn-1,i++,
                q[[i]]= -(T[[i+1]]-T[[i-1]])/2;
        ];
        q[[nn]]= -(T[[1]]-T[[nn-1]])/2;
        T[[1]]= T[[1]]-dt*(q[[2]]-q[[nn]])/2;
        For[i=2,i<=nn-1,i++,
                T[[i]]= T[[i]]-dt*(q[[i+1]]-q[[i-1]])/2;
        ];
        T[[nn]]= T[[nn]]-dt*(q[[1]]-q[[nn-1]])/2;
        Print[T];
        pdat1[[j]]=T[[1]]; pdat2[[j]]=T[[2]];
        pdat3[[j]]=T[[3]]; pdat4[[j]]=T[[4]];
];
```

The results are shown in Figure 4.1. The value of the odd nodes (T_1, T_3) converge to $A/2$, while the even node temperatures remain unchanged. The time dependence of the odd nodes can be described by

$$T_1(t) = \frac{A}{2}\left[1 - e^{-t}\right]; \quad T_3(t) = \frac{A}{2}\left[1 + e^{-t}\right]; \quad T_2(t) = T_4(t) = 0.0 \tag{4.8}$$

This is obviously a non-physical solution for the diffusion equation.

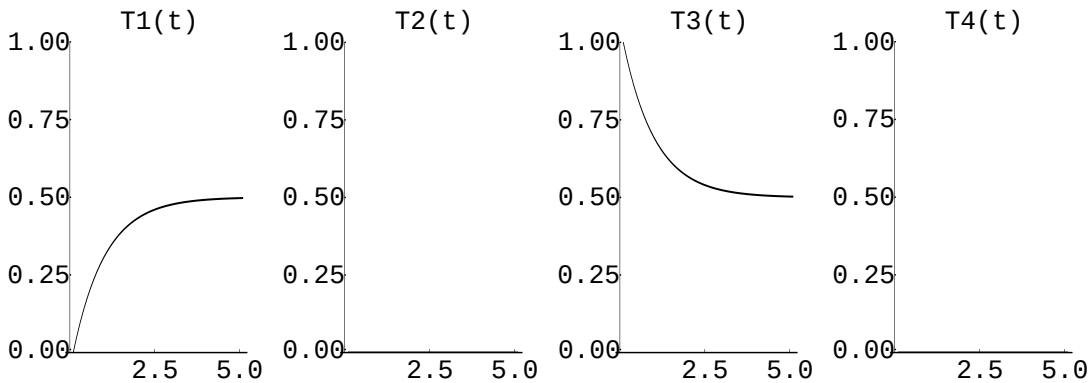


Figure 4.1: Convergence of the centered-difference approximation of the diffusion equation to the wrong solution using a simple grid. The temperatures are factors of A.

To avoid this even/odd numerical instability, a staggered grid is used where the flux (q) is calculated at the cell centers. Again the initial condition is used for the temperature is $T = \{0, 0, A, 0\}$. The heat flux, at the midpoints, is calculated by differencing the values at the nearby nodes. This time, the temperature time-rate-of-change is nonzero on both even and odd numbered nodes and have the correct direction.

```
#      1          2          3          4
     —O——x——O——x——O——x——O——x—
T:     0          0          A          0
q:          0          -A         A          0
Ṫ :    0          A          -2A        A
```

The following MATHEMATICA program iterates equations from an initial condition, $T = \{0, 0, A, 0\}$ with a time step $dt = 0.1$ on a periodic *staggered* grid.

```
(* staggered grid for diffusion equation *)
nn=4; dt=0.1;
T=Table[0,{nn}]; T[[3]]=1; q=Table[0,{nn}];ntime=50;
Print[T];
pdat1=Table[0,{ntime}]; pdat2=Table[0,{ntime}];
```

```
pdat3=Table[0,{ntime}]; pdat4=Table[0,{ntime}];
pdat1[[1]]=T[[1]]; pdat2[[1]]=T[[2]];
pdat3[[1]]=T[[3]]; pdat4[[1]]=T[[4]];
For[j=2,j<=ntime,j++,
        For[i=1,i<=nn-1,i++,
                q[[i]]= -(T[[i+1]]-T[[i]]);
        ];
        q[[nn]]= -(T[[1]]-T[[nn]]);
        T[[1]]= T[[1]]-dt*(q[[1]]-q[[nn]]);
        For[i=2,i<=nn,i++,
                T[[i]]= T[[i]]-dt*(q[[i]]-q[[i-1]]);
        ];
        Print[T];
        pdat1[[j]]=T[[1]]; pdat2[[j]]=T[[2]];
        pdat3[[j]]=T[[3]]; pdat4[[j]]=T[[4]];
];
```

In this case, the system converges to the correct solution $T = \frac{A}{4}\{1,1,1,1\}$. The time dependence of the nodal temperatures can be described by

$$\begin{aligned} T_1(t) =& \frac{A}{4}(1+e^{-4t}) - \frac{A}{2}e^{-2t}; \quad T_2(t) = T_4(t) = \frac{A}{4}(1-e^{-4t}); \\ T_3(t) =& \frac{A}{4}(1+e^{-4t}) + \frac{A}{2}e^{-2t}. \end{aligned} \tag{4.9}$$

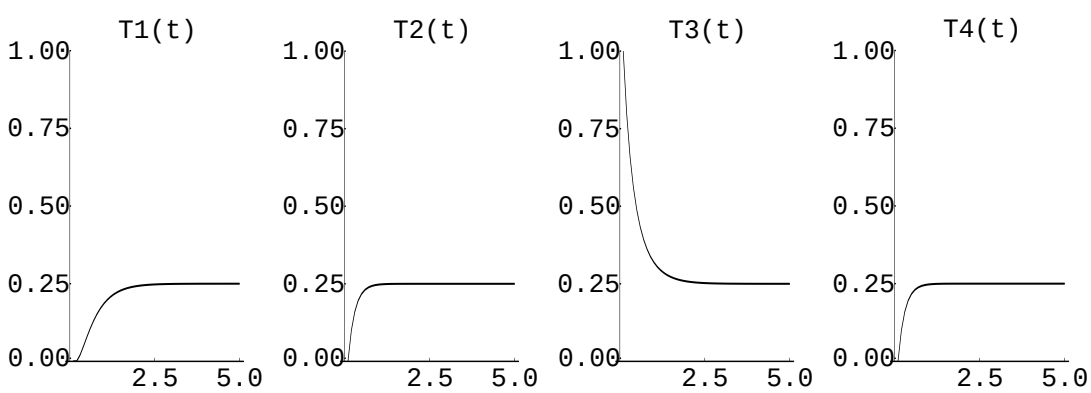


Figure 4.2: Convergence of the centered-difference approximation of the diffusion equation to the correct solution using a staggered grid.

4.3 Boundaries

The choice of boundaries has an influence on the nature of the solution. The boundaries of the system must be selected to represent those of the physical system that is to be modeled. In some cases, it is desirable to study a system in which the effects of physical boundaries are minimized. Periodic boundaries approximate an infinite domain without great computational cost. The computational domain wraps around so that end points become neighbors. The result is an infinitely periodic simulation, of which only one period is actually calculated. For simulating convection, in particular steady convection, periodic boundaries are a reasonable choice for representing an infinite set of pairs of convecting rolls. The symmetry of the problem requires an even number of convecting rolls (counter-rotating pairs) to avoid high shears. The aspect ratio of the simulation is the length of the periodic part (which may be more than one multiple of the wave length of a roll pair) divided by vertical length scale.

The horizontal thermal boundaries are usually modeled as free or rigid, giving three possibilities – both free, both rigid, or one free and one rigid. The rigid boundary differs from a free boundary in that the normal component of the velocity vanishes. The rigid boundary can be made slip or no-slip, where the latter requires the tangential velocity of the fluid, relative to the velocity of the boundary, to vanish. A stress-free boundary requires that the stress, at the boundary, vanish. In all cases, there exists a certain ambiguity concerning the density of the fluid at the boundary for compressible flows.

4.4 "Dual-staggered" grid

A cartesian grid is the natural choice for the discretization of this problem because the system is rectangular. For this problem, we use a cartesian "dual-staggered" grid. This

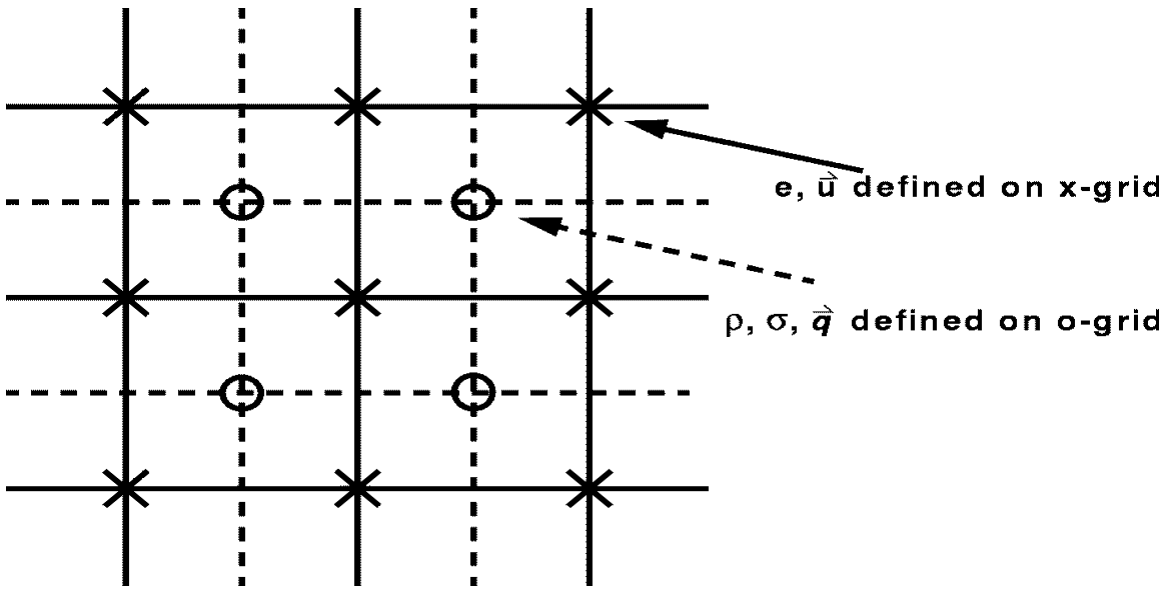


Figure 4.3: Internal energy and momentum are updated on one grid of the "dual-staggered" mesh while the mass density is updated on the other.

means that the state variables are updated on two different grids (the "dual" part) and the fluxes are calculated at the grid midpoints (the "staggered" part).

At the rigid, thermal boundaries, the temperature and velocities are specified, but the mass density is ambiguous. To deal with this ambiguity, a "dual" grid is used where the temperature and velocity are updated on one grid (the grid that includes the rigid boundaries) and the mass density is updated on the other. The values of the state variables are interpolated to intermediate (staggered) grids where the fluxes of the conserved quantities

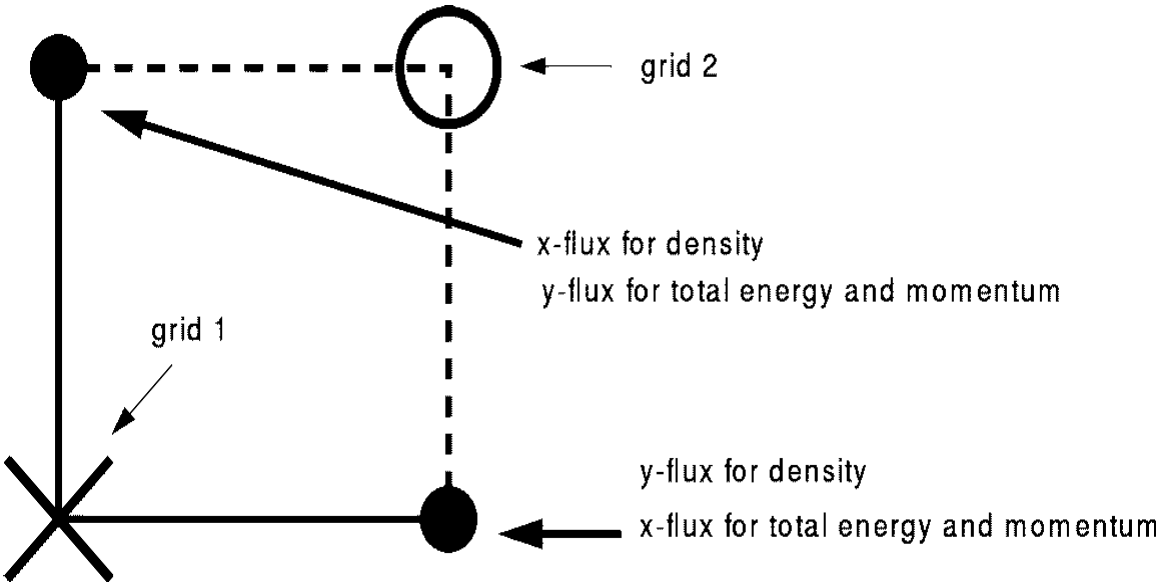


Figure 4.4: Fluxes of conserved quantities are updated at the midpoints (staggered) along each grid line of the "dual-staggered" grid.

are calculated. The staggered grid scenario is not uncommon – it is used to avoid the "even/odd" instability, where the even numbered nodes become decoupled from the odd numbered nodes.

Figure 4.3 shows the cartesian grid on which the energy and momentum are defined (nodes are indicated by an "X") and the grid on which the mass density (nodes are indicated by an "O") are defined. This configuration is also known as a density-centered grid because the density nodes are in the center of zones whose corners are the X-nodes. Figure 4.4 shows the two grids in more detail. The solid circles at the intersection of the grid indicates the place at which the fluxes are calculated. These points can also be thought of as the midpoints between nodes along lines. The x-flux for the density and the y-flux for the energy and momentum are calculated on one flux point while the y-flux

for the density and the x-flux for the energy and momentum are calculated on the other.

The updating of the conserved quantities at each time-step requires a sequence of steps that involve interpolation and differentiation of quantities. The steps are outlined here.

1. Interpolate new density to other grids
    - interpolate density to midpoint in x-direction
    - interpolate density to midpoint in y-direction
    - interpolate midpoint density to Grid 1
2. Determine state variables u,v,e from quantities on Grid 1
    - x-velocity equals x-momentum divided by density
    - y-velocity equals y-momentum divided by density
    - internal energy is determined from density and velocities
3. Interpolate state variables
    - interpolate x-velocity to midpoint in y-direction
    - interpolate y-velocity to midpoint in y-direction
    - interpolate internal energy to midpoint in y-direction
    - interpolate x-velocity to midpoint in x-direction
    - interpolate y-velocity to midpoint in x-direction
    - interpolate internal energy to midpoint in x-direction
4. Determine gradients of state variables
    - determine midpoint gradient of x-velocity in x-direction (u_x)
    - determine midpoint gradient of y-velocity in x-direction (v_x)
    - determine midpoint gradient of internal energy in x-direction (e_x)
    - determine midpoint gradient of x-velocity in y-direction (u_y)
    - determine midpoint gradient of y-velocity in y-direction (v_y)
    - determine midpoint gradient of internal energy in y-direction (e_y)
5. Interpolate velocity gradients to opposite grid

- interpolate u_x to Grid 2
- interpolate v_x to Grid 2
- interpolate u_y to Grid 2
- interpolate v_y to Grid 2
- interpolate Grid 2 u_x to midpoint in y-direction
- interpolate Grid 2 v_x to midpoint in y-direction
- interpolate Grid 2 u_y to midpoint in x-direction
- interpolate Grid 2 v_y to midpoint in x-direction

6. Determine flux values

   - calculate q_x at midpoint
   - calculate σ_{xx} at midpoint
   - calculate σ_{xy} at midpoint
   - calculate midpoint total energy
   - calculate y-flux of density at midpoint
   - calculate x-flux of x-momentum at midpoint
   - calculate x-flux of y-momentum at midpoint
   - calculate x-flux of total energy at midpoint
   - calculate q_y at midpoint
   - calculate σ_{yy} at midpoint
   - calculate σ_{xy} at midpoint
   - calculate midpoint total energy
   - calculate x-flux of density at midpoint
   - calculate y-flux of x-momentum at midpoint
   - calculate y-flux of y-momentum at midpoint
   - calculate y-flux of total energy at midpoint

7. Compute sources (mass density has no source)

   - calculate y-momentum source
   - total energy source

8. Compute divergence of the fluxes

   - calculate divergence of density flux in x-direction

- calculate divergence of density flux in y-direction
- calculate divergence of x-momentum flux in x-direction
- calculate divergence of x-momentum flux in y-direction
- calculate divergence of y-momentum flux in x-direction
- calculate divergence of y-momentum flux in y-direction
- calculate divergence of total energy flux in x-direction
- calculate divergence of total energy flux in y-direction

9. return to time integrator

Obviously, there is a need to employ a method that calculates the midpoint interpolant and spatial derivative with high accuracy, but with relatively-low computational cost. The methods used are discussed in the next chapter.

Chapter 5

Cubic spline collocation

5.1 Background

The fully compressible, viscous, time-dependent Navier-Stokes equations are rich in possible solutions. The compressibility allows for sound (compression) waves in addition to buoyancy-driven convection. Many approaches exist for solving the time-dependent Navier-Stokes equations with specified initial and boundary conditions. Each has its advantages and disadvantages. Approaches for solving fluid flow problems generally fall into one of two groups – Lagrangian or Eulerian. Lagrangian approaches follow the flow of the fluid and allow the grid to track the fluid interface or other regions of interest. Often these moving meshes become tangled and produce difficulties such as "hour-glassing". Grid-free (free-Lagrangian) approaches avoid these problems by tracking particles that may represent fluid molecules (such as the Direct Simulation Monté

Carlo (DSMC) method [55, 56, 11]), a group of particles (such as in the case of Cellular Automata (CA) [57, 58], Smooth Particle Applied Mechanics (SPAM) [14, 59, 60], and Particle in Cell (PIC) [61]) or statistical distributions (such as Grid and Particle Hydrodynamics (GaPH) [62]). Hybrid methods have also been developed such as the Arbitrary Lagrangian-Eulerian (ALE) method to take advantage of the Lagrange method while avoiding mesh entanglement.

In the case of Eulerian methods, the grid is fixed in space and the fluid moves through it. These methods are often easier to implement and require less computational effort to solve. The disadvantage with these methods is that often the phenomena under study may be at a length scale much less that the length scale of the problem – requiring an unreasonable number of grid points. One can use multigrid methods which refine the grid locally around the region of interest. If the region is moving, as in the case of a moving interface, adaptive mesh refinement methods are used.

5.1.1 Finite difference methods

Once a discrete grid has been constructed on the domain of the problem, the continuous variables and their derivatives can be approximated to the desired accuracy with truncated Taylor series at the nodal points. An example of this is the second order central difference approximation of the first derivative, in one dimension, of a function f,

$$f'_j \approx \frac{f_{j+1} - f_{j-1}}{2h}, \tag{5.1}$$

which has an error, due to truncation, of order $f'''h^2$, where h is the grid spacing.

5.1.2 Finite element methods

The idea behind the finite element method is to break up the domain of the problem into small elements, over which the continuous variables are interpolated. This is accomplished by first discretizing the spatial domain into subdomains of arbitrary shape and size. These elements are further reduced to triangular and quadrilateral elements that do not overlap and that provide complete coverage of the computational domain. The sides of the elements are referred to as the edges whose intersections define the nodal points of the domain. The objective is to solve for the unknowns (usually the state variables) at the nodes for a particular time. This is usually accomplished by approximating the field variables by a linear combination of basis functions $N_i(x)$ (also referred to as shape, interpolation, or trial functions),

$$\tilde{f} \approx \sum_i \alpha_i(t) N_i(x) \tag{5.2}$$

where the sum is over all nodes.

5.1.3 Padé approximation

To solve the continuum equations on a uniformly spaced mesh, the derivative of the flux terms can be determined by the application of a Padé-like [63] method where

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = \frac{c}{6h}(f_{i+3} - f_{i-3}) + \frac{b}{4h}(f_{i+2} - f_{i-2}) + \frac{a}{2h}(f_{i+1} - f_{i-1}) \tag{5.3}$$

The values of the coefficients are related by matching the Taylor series coefficients to get the desired truncation error. For fourth order accuracy in calculating the first derivative, β is set to zero, leaving a tridiagonal system to be solved. Choosing $\alpha = \frac{1}{4}$, $b = c = 0$, and $a = \frac{3}{2}$ the classical Padé scheme is recovered with a truncation error of $\frac{1}{120}h^4 f^{(5)}$,

$$f'_{i-1} + 4f'_i + f'_{i+1} = \frac{3}{h}(f_{i+1} - f_{i-1}). \tag{5.4}$$

At fixed boundaries, a one-sided form of the equation,

$$f'_1 + \alpha f'_2 = \frac{1}{h}(af_1 + bf_2 + cf_3 + df_4) \tag{5.5}$$

can be used with $\alpha = 3$, $a = -\frac{17}{6}$, $b = \frac{3}{2}$, $c = \frac{3}{2}$, $d = -\frac{1}{6}$. This preserves the tridiagonal nature of the system and the fourth order accuracy.

5.1.4 Collocation

A collocation method involves the determination of an approximate solution by a suitable set of functions (trial functions) that satisfy the boundary conditions at certain points (collocation points). In time-dependent problems, the method is used to approximate the spatial dependence of the variables. A comprehensive history of collocation methods from the method used by Kantorovich in the 1930's to the present is provided by Fairweather and Meade [64]. Kantorovich's method uses a collocation approximations for the methods of lines solution of a partial differential equation in two variables. The collocation is applied for each variable while the other is held fixed.

Examples of Collocation Methods

In general, a collocation method uses a function, or set of functions, that satisfies certain conditions at the collocation points to approximate a field variable. This concept is incorporated into many approaches for solving continuum problems. For example, in the Weighted Residual Method (WRM), one determines an approximation of the field variable, $f = f(x,t)$ as a linear sum of basis functions,

$$f = \sum_i \alpha_i(t)\phi_i(x), \tag{5.6}$$

that satisfies a differential equation,

$$\mathbf{L}[f^*] = 0; \qquad \mathbf{L}[f] - \mathbf{L}[f^*] = \mathbf{R}[f], \tag{5.7}$$

where f^* is the true field variable and $\mathbf{R}[f]$ is the residue function. For a particular method, a weight function W_m is assigned, and the $\alpha_i(t)$ are determined that lead to,

$$\int_\Omega W_m \mathbf{R}[f] = 0 \tag{5.8}$$

over the whole domain of the problem. For example, Finite Volume methods divide the problem into sub-domains D_m such that

$$W_m = \begin{cases} 1 & \text{in } D_m, \\ 0 & \text{outside } D_m. \end{cases} \tag{5.9}$$

Another popular example is the Galerkin method, where $W_m = \phi_m$, where $\{\phi_m\}$ is a complete set (for example: $\phi \in \{1, x, x^2, x^3, \dots\}$). The collocation WRM can be expressed as

$$W_m = \delta(x - x_m), \tag{5.10}$$

where δ is the Dirac delta function and x_m represents the collocation points. This leads to $\mathbf{R}[f(x_m, t)] = 0$. The Finite Difference method is a special case of the collocation

WRM without an approximate solution – the actual solution is collocated at the node points. The spectral method is a form of WRM, similar to the Galerkin method, that uses orthogonal functions for the basis and weight functions. When nonlinear terms are involved, the solution becomes very time consuming. For economy, often the solution is expressed in terms of nodal unknowns (collocation). The explicit use of nodal unknowns allows boundary information to be incorporated easily into the solution. In the literature, the collocation spectral method is referred to as the pseudospectral method [65].

Spline collocation is the use of spline functions, that satisfy conditions at the nodal point, to represent the field variables over the domain of the problem. In comparison with finite difference methods, spline collocation provides approximations to the solution and its derivative with respect to x at all points of the domain of the problem.

A cubic spline collocation method, used to solve the compressible Navier-Stokes equations on the dual-staggered grid, is developed and analyzed in the remainder of this chapter.

5.2 Cubic spline approximations

A new method is used in this thesis to model convective heat transport in a heat-conducting, viscous fluid [66, 15, 28, 29, 67]. The method involves approximating the field variables with cubic splines and using this approximation to solve the flux-form of the compressible Navier-Stokes equations. The fluxes of the conserved quantities are also approximated by cubic spline functions to determine the spatial derivatives, and hence,

the time derivatives of the conserved quantities.

Using equally-spaced nodes with $h \equiv x_{j+1} - x_j,\ \forall j$, a spline function $S_\Delta(x)$ is created such that $S_\Delta(x_j) = f(x_j) \equiv f_j$. The second derivative of the spline function at the nodes is denoted as, $M_j \equiv S''_\Delta(x_j)$. Now, since the cubic spline consists of piecewise third-order polynomials, $S''_\Delta(x)$ is piecewise linear. Therefore

$$S''_\Delta(x) = M_j \frac{x_{j+1} - x}{h} + M_{j+1} \frac{x - x_j}{h}, \quad x \in [x_j, x_{j+1}]. \tag{5.11}$$

Integration leads to,

$$S'_\Delta(x) = -M_j \frac{(x_{j+1} - x)^2}{2h} + M_{j+1} \frac{(x - x_j)^2}{2h} + A_j, \tag{5.12}$$

$$S_\Delta(x) = M_j \frac{(x_{j+1} - x)^3}{6h} + M_{j+1} \frac{(x - x_j)^3}{6h} + A_j(x - x_j) + B_j. \tag{5.13}$$

From $S_\Delta(x_j) = f_j$ and $S_\Delta(x_{j+1}) = f_{j+1}$, A_j and B_j are defined

$$A_j = \frac{f_{j+1} - f_j}{h} - h\frac{M_{j+1} - M_j}{6}; \qquad B_j = f_j - M_j \frac{h^2}{6}. \tag{5.14}$$

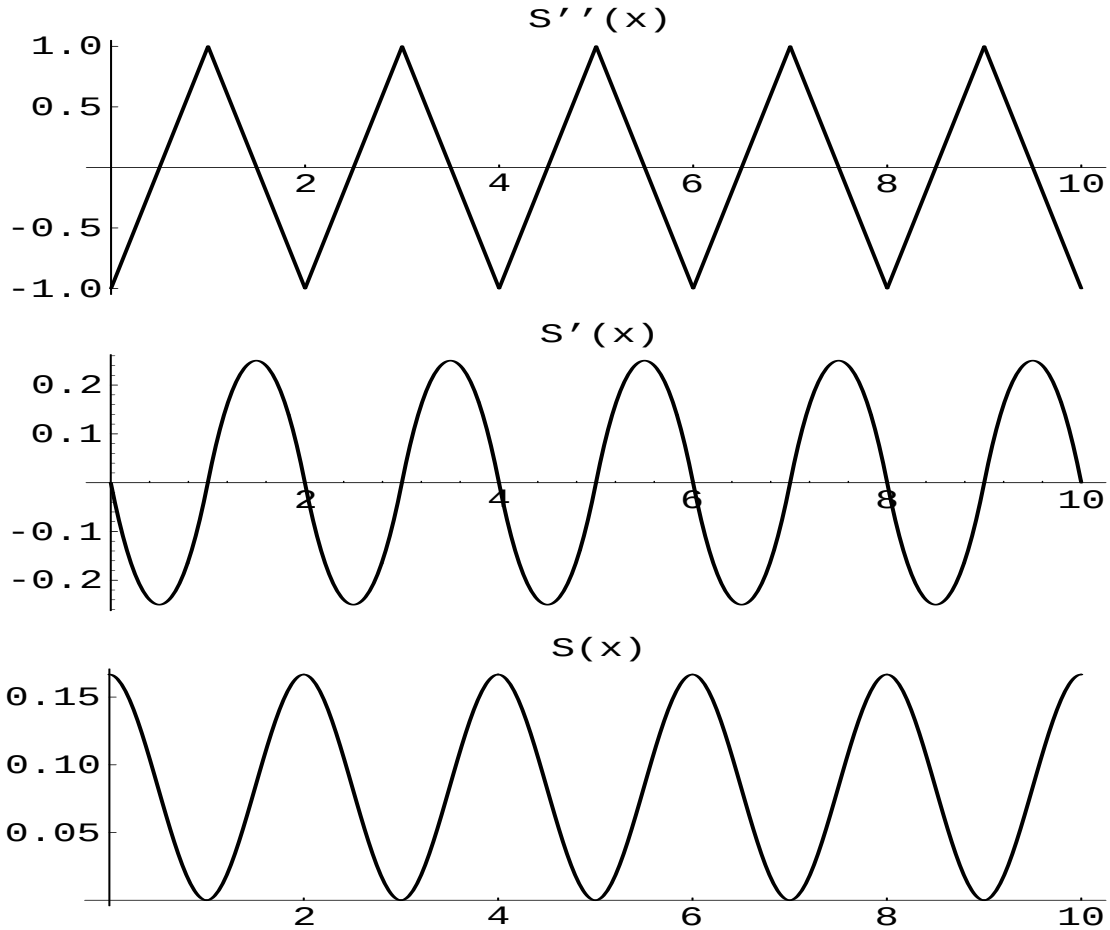


Figure 5.1: Piecewise-cubic spline interpolating polynomial $S_\Delta(x)$ and its derivatives $S'_\Delta(x)$ and $S''_\Delta(x)$. This piecewise-linear function for the second derivative was generated from $S''(x_j) = (j \mod 2)$.

5.3 Midpoint interpolation and differentiation

Because the state variables are known at the vertices of an equally spaced grid, the midpoint interpolant $\hat{f}_0$ and spatial derivative f'_0 are:

$$\hat{f}_0 = \frac{f_{j+1} + f_j}{2} - \frac{h^2}{16}(M_{j+1} + M_j) \tag{5.15}$$

$$f_0' = \frac{f_{j+1} - f_j}{h} - \frac{h}{24}(M_{j+1} - M_j) \tag{5.16}$$

where $f_0 \equiv f(x_j + \frac{h}{2})$.

5.4 Accuracy – local truncation error

Convergence of midpoint interpolant

To determine the local truncation error of the midpoint interpolation using cubic splines [Equation (5.15)] is $O(h^4)$, consider Taylor's expansion about $f_0 \equiv f(x_j + \frac{h}{2})$. We use $k \equiv \frac{h}{2}$ to keep the expansions in familiar form:

$$f_{j+1} = f_0 + kf_0' + \frac{k^2}{2}f_0'' + \frac{k^3}{6}f_0''' + \frac{k^4}{24}f_0'''' + O(k^5) \tag{5.17}$$

$$f_j = f_0 - kf_0' + \frac{k^2}{2}f_0'' - \frac{k^3}{6}f_0''' + \frac{k^4}{24}f_0'''' + O(k^5). \tag{5.18}$$

Adding (5.17) and (5.18) leads to

$$f_0 = \frac{f_{j+1} + f_j}{2} - \frac{k^2}{2}f_0'' - \frac{k^4}{12}f_0'''' + O(k^6). \tag{5.19}$$

differentiating (5.19) twice gives,

$$f_0'' = \frac{f_{j+1}'' + f_j''}{2} - \frac{k^2}{2} f_0'''' + O(k^4). \tag{5.20}$$

Substituting (5.20) into (5.19) gives

$$f_0 = \frac{f_{j+1} + f_j}{2} - \frac{k^2}{4}(f_{j+1}'' + f_j'') + \frac{11}{24} k^4 f_0'''' + O(k^5). \tag{5.21}$$

To get (5.21) into the form (5.15), while tracking the truncation error, we need to derive the relation between $f''(x_j)$ and M_j. First, we define δ such that $0 < \delta < h \equiv x_{j+1} - x_j$, with $f_+ \equiv f(x_j + \delta)$ and $f_- \equiv f(x_j - \delta)$. Taylor's expansions about a node are:

$$f_+ = f_j + \delta f_j' + \frac{\delta^2}{2} f_j'' + \frac{\delta^3}{6} f_j''' + \frac{\delta^4}{24} f_j'''' + O(\delta^5) \tag{5.22}$$

$$f_- = f_j - \delta f_j' + \frac{\delta^2}{2} f_j'' - \frac{\delta^3}{6} f_j''' + \frac{\delta^4}{24} f_j'''' + O(\delta^5). \tag{5.23}$$

Adding (5.22) and (5.23) leads to

$$f_+ + f_- = 2f_j + \delta^2 f_j'' + \frac{\delta^4}{12} f_j'''' + O(\delta^6). \tag{5.24}$$

Similarly, for the cubic spline function, let $S_j \equiv S_\Delta(x_j)$, $S_+ \equiv S_\Delta(x_j + \delta)$, and $S_- \equiv$

$S_\Delta(x_j - \delta)$, so that

$$S_+ + S_- = 2S_j + \delta^2 S_j'' + \frac{\delta^4}{12} S_j'''' + O(\delta^6). \tag{5.25}$$

We are careful to keep δ strictly greater than 0 and less than h so that the fourth derivative is defined

$$S_\Delta''''(x) = 0, \quad x_j < x < x_{j+1}. \tag{5.26}$$

Subtracting (5.24) from (5.25), and noting that $S_j'' \equiv M_j$ gives

$$(S_+ - f_+) + (S_- - f_-) = 2(S_j - f_j) + \delta^2(M_j - f_j'') - \frac{\delta^4}{12} f_j'''' + O(\delta^6). \tag{5.27}$$

Now, since $f_j = S_j$, $\forall j$, we can set δ arbitrarily close to h so the $(S_+ - f_+)$ and $(S_- - f_-)$ terms are arbitrarily close to zero, yielding

$$M_j - f_j'' = \frac{h^2}{12} f_j'''' + O(h^4). \tag{5.28}$$

We can now put (5.21) into the form (5.15), using (5.28) to determine the midpoint interpolation truncation error,

$$\boxed{\hat{f}_0 = \frac{f_{j+1} + f_j}{2} - \frac{h^2}{16}(M_{j+1} + M_j) + O(h^4)} \tag{5.29}$$

Convergence of midpoint spatial derivative

To show that the local truncation error of the midpoint spatial derivative using cubic splines (5.16) is $O(h^4)$, we again use the Taylor expansions. Again we use $k \equiv \frac{h}{2}$. Subtracting (5.18) from (5.17) yields

$$f_0' = \frac{f_{j+1} - f_j}{2k} - \frac{k^2}{6} f_0''' + O(k^4). \tag{5.30}$$

differentiating twice gives,

$$f_0''' = \frac{f_{j+1}'' - f_j''}{2k} + O(k^2). \tag{5.31}$$

Substituting (5.31) into (5.30) gives

$$f_0' = \frac{f_{j+1} - f_j}{2k} - \frac{k}{12}(f_{j+1}'' - f_j'') + O(k^4). \tag{5.32}$$

This time, using the error associated with approximating f_j'' by M_j does not give the whole story. Instead, we need the error associated with approximating $(f_{j+1}'' - f_j'')$ with $(M_{j+1} - M_j)$. From (5.28), we get

$$M_{j+1} - f_{j+1}'' = \frac{h^2}{12} f_{j+1}'''' + O(h^4) \tag{5.33}$$

$$M_j - f_j'' = \frac{h^2}{12} f_j'''' + O(h^4). \tag{5.34}$$

Subtracting (5.34) from (5.33) and multiplying by $\frac{h}{24}$ yields

$$\frac{h}{24}(M_{j+1} - M_j) = \frac{h}{24}(f_{j+1}'' - f_j'') + \frac{h^3}{288}(f_{j+1}'''' - f_j'''') + O(h^5). \tag{5.35}$$

Differentiating the midpoint Taylor expansions (5.17) and (5.18) four times gives

$$f_{j+1}'''' = f_{j+\frac{1}{2}}'''' + \frac{h}{2} f_{j+\frac{1}{2}}''''' + \frac{h^2}{8} f_{j+\frac{1}{2}}'''''' + O(h^3) \tag{5.36}$$

$$f_j'''' = f_{j+\frac{1}{2}}'''' - \frac{h}{2} f_{j+\frac{1}{2}}''''' + \frac{h^2}{8} f_{j+\frac{1}{2}}'''''' + O(h^3). \tag{5.37}$$

Subtracting (5.37) from (5.36) gives

$$f_{j+1}'''' - f_j'''' = h f_{j+\frac{1}{2}}''''' + O(h^3). \tag{5.38}$$

Substituting (5.38) into (5.35) gives

$$\frac{h}{24}(M_{j+1} - M_j) = \frac{h}{24}(f_{j+1}'' - f_j'') + O(h^4). \tag{5.39}$$

Which leads to the truncation error for the midpoint spatial derivative:

$$\boxed{f_0' = \frac{f_{j+1} - f_j}{h} - \frac{h}{24}(M_{j+1} - M_j) + O(h^4)} \tag{5.40}$$

5.5 Flux conservation

Once the differential equations are written in conservative form, the numerical method must account for the loss and gain of fluxes locally so that the total change in the mass, momentum, and energy at each time step is random and with a magnitude on the order of the machine accuracy (double-precision is used in these calculations). Since these calculations are locally one dimensional, the midpoint gradient approximation (5.40) is used for the divergence calculation.

To show that this method is strictly conservative, we integrate the time-rate-of-change of the conserved quantity Q along each (1D) grid line. Using the conservative form of the differential equation (4.4),

$$\int_L \frac{\partial}{\partial t} Q dx = \int_L [-\frac{\partial}{\partial x} F + S]\, dx \tag{5.41}$$

where F is the flux and S is the source. Discretizing the integral yields

$$\frac{\partial}{\partial t} h \sum_{j=0}^{n-1} Q_{j+\frac{1}{2}} = -h \sum_{j=0}^{n-1} F'_{j+\frac{1}{2}} + h \sum_{j=0}^{n-1} S_{j+\frac{1}{2}}. \tag{5.42}$$

Here, the conserved quantity, its source, and the midpoint spatial derivative of the flux are on one grid (the midpoint in this case) and the flux is on the other grid. Substituting (5.16) for F' and Q_{tot} and S_{tot} for the integrals of Q and S respectively, we get

$$\frac{\partial}{\partial t}Q_{tot} = -h\sum_{j=0}^{n-1}\left[\frac{F_{j+1}-F_j}{h} - \frac{h}{24}(M_{j+1}-M_j)\right] + S_{tot}. \tag{5.43}$$

Expanding the sums

$$\begin{aligned}\frac{\partial}{\partial t}Q_{tot} &= (F_0 - F_1) + (F_1 - F_2) + \cdots + (F_{n-1} - F_n) \\ &\quad - \frac{h^2}{24}[(M_0 - M_1) + (M_1 - M_2) + \cdots + (M_{n-1} - M_n)] + S_{tot} \end{aligned} \tag{5.44}$$

$$= (F_0 - F_n) - \frac{h^2}{24}(M_0 - M_n) + S_{tot}. \tag{5.45}$$

By setting $M_0 = M_n$, we recover,

$$\frac{\partial}{\partial t}Q_{tot} = (F_0 - F_n) + S_{tot} \tag{5.46}$$

where the time-rate-of-change of the total conserved quantity is equal to the flux in from the boundaries minus the flux out to the boundaries plus the total other (boundary independent) sources. For periodic boundaries, $F_0 = F_n$ and $M_0 = M_n$, so this condition is satisfied. For fixed boundaries, the choice of $M_0 = M_n$ is satisfied by the "natural" end conditions, $M_0 = M_n = 0$. This condition, however, appears to introduce an extra

error at the rigid boundaries where,

$$f'_{\frac{1}{2}} = \frac{f_1 - f_0}{h} - \frac{h}{24}(M_1); \qquad f'_{n-\frac{1}{2}} = \frac{f_n - f_{n-1}}{h} + \frac{h}{24}(M_{n-1}) \tag{5.47}$$

because M_0 and M_n have been arbitrarily set equal to zero. The apparent decrease in the convergence rate, $O(h)$ rather than $O(h^4)$, does not seem to affect the rest of the calculation. For turbulent simulations, along with other high-Reynolds-number flows, a boundary layer forms near rigid boundaries, In thermal and viscous boundary layers, the transport is linear, making the assumption $M_0 = M_n = 0$ reasonable.

Equation (5.42) describes the discrete form of the conservation equation for a conserved quantity updated on the $(j + \frac{1}{2})$-grid (in this case mass density). The results show that the total change in mass for each time step is equal to the flux at the boundaries plus the source of mass. In the simulations in this thesis, there is no mass source ($S_\rho = 0$) and no mass flux at the walls ($F_\rho(x = 0) = F_\rho(x = L) = (\rho v)_{boundary} = 0$). Therefore, the total mass of the simulation is constant. In the case of conserved quantities updated on the j-grid (momentum and energy), equation (5.41) is discretized as

$$\frac{\partial}{\partial t} h \sum_{j=1}^{n-1} Q_j = -h \sum_{j=1}^{n-1} F'_j + h \sum_{j=1}^{n-1} S_j \tag{5.48}$$

in the horizontal direction (the vertical, periodic direction is again trivial). The boundary terms, Q_0 and Q_n, are not included in the sum because the momentum and energy are defined by the boundary conditions and are not updated. Again, substituting (5.16) for

F' and Q_{tot} and S_{tot} for the integrals of Q and S respectively, we get

$$\frac{\partial}{\partial t}Q_{tot} = -h\sum_{j=1}^{n-1}\left[\frac{F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}}}{h} - \frac{h}{24}(M_{j+\frac{1}{2}} - M_{j-\frac{1}{2}})\right] + S_{tot}. \tag{5.49}$$

Expanding the sums

$$\begin{aligned}\frac{\partial}{\partial t}Q_{tot} &= (F_{\frac{1}{2}} - F_{\frac{3}{2}}) + (F_{\frac{3}{2}} - F_{\frac{5}{2}}) + \cdots + (F_{n-\frac{3}{2}} - F_{n-\frac{1}{2}}) \\ &\quad - \frac{h^2}{24}[(M_{\frac{1}{2}} - M_{\frac{3}{2}}) + (M_{\frac{3}{2}} - M_{\frac{5}{2}}) + \cdots + (M_{n-\frac{3}{2}} - M_{n-\frac{1}{2}})] + S_{tot} \qquad (5.50)\\ &= (F_{\frac{1}{2}} - F_{n-\frac{1}{2}}) - \frac{h^2}{24}(M_{\frac{1}{2}} - M_{n-\frac{1}{2}}) + S_{tot}. \qquad (5.51)\end{aligned}$$

In this case, we set $M_{\frac{1}{2}} = M_{n-\frac{1}{2}} = 0$ for conservation leaving,

$$\frac{\partial}{\partial t}Q_{tot} = (F_{\frac{1}{2}} - F_{n-\frac{1}{2}}) + S_{tot}, \tag{5.52}$$

where the flux boundary is displaced from the walls by half a cell size. The apparent error introduced at the boundary due to the conservation constraint,

$$f_1' = \frac{f_{\frac{3}{2}} - f_{\frac{1}{2}}}{h} - \frac{h}{24}(M_{\frac{3}{2}}); \qquad f_{n-1}' = \frac{f_{n-\frac{1}{2}} - f_{n-\frac{3}{2}}}{h} + \frac{h}{24}(M_{n-\frac{3}{2}}) \tag{5.53}$$

is $O(h)$. Again, if we assume that the viscous and thermal boundary layers have a length scale greater than the cell size, we can assume linear transport, making $f''(\frac{h}{2}) = 0$ and $f''(L - \frac{h}{2}) = 0$. These reasonable assumptions imply truncation errors of $O(h^4)$.

5.6 Calculation of the spline function

Assuming the end conditions are established, there are an equal number of unknowns and equations. We can solve a linear system for the second derivative of the spline function at the nodes $\{M_j\}$.

5.6.1 Fixed boundaries

For a grid line with fixed boundaries, the linear system is

$$\begin{bmatrix} 4 & 0 & & & & & 0 \\ 1 & 4 & 1 & & & & \\ & 1 & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & 1 & \\ & & & & 1 & 4 & 1 \\ 0 & & & & & 0 & 4 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \cdot \\ \cdot \\ \cdot \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ d_{n-1} \\ d_n \end{bmatrix} \tag{5.54}$$

where $M_j \equiv S''_\Delta(x_j)$ and

$$d_j = \begin{cases} 0 & j = 0, n \\ \frac{6}{h^2}\left[f_{j+1} - 2f_j + f_{j-1}\right] & j = 1, 2, \ldots, n-1 \end{cases} \tag{5.55}$$

To solve this linear system, we use the *LU* decomposition method which, for a tridi-

agonal matrix, reduces to the *Thomas* algorithm. The tridiagonal linear system can be written as

$$a_j M_{j-1} + b_j M_j + c_j M_{j+1} = d_j \qquad j = 1, \ldots N \tag{5.56}$$

with $a_1 = c_N = 0$. The Thomas algorithm solves the system in two stages – a forward step followed by a backward step:

Forward step

$$\begin{aligned} \beta_1 = b_1; \qquad \beta_j &= b_j - a_j \frac{c_{j-1}}{\beta_{j-1}}, \qquad j = 2, \ldots N \\ \gamma_1 = \frac{d_1}{\beta_1}; \qquad \gamma_j &= \frac{(-a_j \gamma_{j-1} + d_j)}{\beta_j,} \qquad j = 2, \ldots N \end{aligned} \tag{5.57}$$

Backward step

$$M_N = \gamma_N; \qquad M_j = \gamma_j - M_{j+1} \frac{c_j}{\beta_j}, \qquad j = N-1, \ldots 1. \tag{5.58}$$

This requires 5N operations. Because the matrix for the cubic spline solution is diagonally dominant, the procedure will converge.

The subroutine TRIDIA uses the Thomas algorithm to solve a tridiagonal system in two stages.

Subroutine TRIDIA

```
      subroutine TRIDIA(a,b,c,r,u,n)
C*************************************************************
C     subroutine TRIDIA solves the linear system Tu=r.
C     T is tridiagonal with major and minor diagonals
C     c and a respectively. b is the main diagonal.
C*************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      parameter (Nmax=1000)
      dimension gam(nmax),a(n),b(n),c(n),r(n),u(n)
      if(b(1).eq.0.0d0) pause
      bet=b(1)
      u(1) = r(1)/bet
      do 10 j=2,n
         gam(j)=c(j-1)/bet
         bet=b(j)-a(j)*gam(j)
         if(bet.eq.0.0d0) then
            write(*,*) j,a(j),b(j),c(j)
            pause
         end if
         u(j)=(r(j)-a(j)*u(j-1))/bet
 10   continue
C
C     BACK SUBSTITUTION
C
      do 20 j=n-1,1,-1
         u(j)=u(j)-gam(j+1)*u(j+1)
 20   continue
      return
      end
```

5.6.2 Periodic boundaries

For the grid lines with periodic boundaries, the preceding method cannot be used because of the non-zero terms in the extreme corners. The linear system, for periodic

boundaries is

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & \cdot & \cdot & & & \\ & & \cdot & \cdot & \cdot & & \\ & & & \cdot & \cdot & 1 & \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \cdot \\ \cdot \\ \cdot \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \cdot \\ \cdot \\ \cdot \\ d_{n-1} \\ d_n \end{bmatrix} \tag{5.59}$$

where

$$\begin{aligned} d_0 = \frac{6}{h^2}[f_1 - 2f_0 + f_n]; \qquad d_n = \frac{6}{h^2}[f_0 - 2f_n + f_{n-1}]; \\ d_j = \frac{6}{h^2}[f_{j+1} - 2f_j + f_{j-1}], \qquad j = 2, \cdots, n-1. \end{aligned} \tag{5.60}$$

This nearly tridiagonal system can also be solved efficiently, where the computational effort scales linearly with the number of nodes. This procedure [68] requires the subroutine DBBFA to LU decompose the border-banded matrix T_p without pivoting. The subroutine DBBSL is called twice – once to solve the $Ly = b$ system and once to solve the $Ux = y$ system. The BLAS routines DAXPY and DSCAL are called by DBBSL. These double precision routines DBBFA, DBBSL, DAXPY and DSCAL are in Appendix B.1.

Another approach for solving the nearly-tridiagonal equation for the cubic spline with periodic boundaries is given by Hirsch [69]. This approach leads to an algorithm in which

two tridiagonal systems must be solved. For a periodic nearly-tridiagonal matrix $T_p(\vec{a}, \vec{b}, \vec{c})$ with $N+1$ rows and columns, the solution is written as a linear combination

$$\vec{x} = \vec{x}^{(1)} + x_{N+1}\vec{x}^{(2)}, \tag{5.61}$$

where $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ are the solution of the tridiagonal systems with the last row and the last column removed, respectively,

$$T_p^{(1)}\vec{x}^{(1)} = \vec{f}; \qquad T_p^{(2)}\vec{x}^{(2)} = \vec{g} \tag{5.62}$$

where,

$$\vec{g}^t \equiv \{-a_1, \ldots, 0, \ldots, -c_N\}. \tag{5.63}$$

The last unknown is,

$$x_{N+1} = \frac{f_{N+1} - c_{N+1}x_1^{(1)} - a_{N+1}x_N^{(1)}}{b_{N+1} + a_{N+1}x_N^{(2)} + C_{N+1}x_1^{(2)}} \tag{5.64}$$

The subroutine TRIPER uses this method to solve a periodic tridiagonal linear system.

Subroutine TRIPER

```
      SUBROUTINE TRIPER(AA,BB,CC,FF,N1,N,GAM2)
C*****************************************************************
C     SOLUTION OF A TRIDIAGONAL SYSTEM OF EQUATIONS WITH PERIODICITY
C     IN THE POINTS K=N1 AND K=N+2
C
C     THE ELEMENTS IN THE UPPER RIGHT AND LOWER LEFT CORNERS ARE
```

```
C     AA(N1) AND CC(N+1) RESPECTIVELY
C     AA,BB,CC,FF,GAM2 ARE VECTORS WITH DIMENSIONS N+2.
C     GAM2 IS USED FOR STORAGE. THE SOLUTION IS STORED IN FF
C*****************************************************************
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION AA(1),BB(1),CC(1),FF(1),GAM2(1)
BB=(N1)=1./BB(N1)
GAM2(N1)=-AA(N1)*BB(N1)
AA(N1)=FF(N1)*BB(N1)
N2=N1+1
      N1N=N1+N
      DO 10 K=N2,N
      K1=K-1
      CC(K1)=CC(K1)*BB(K1)
      BB(K) =BB(K)-AA(K)*CC(K1)
      BB(K) =1./BB(K)
      GAM2(K)=-AA(K)*GAM2(K1)*BB(K)
      AA(K) =(FF(K)-AA(K)*AA(K1))*BB(K)
 10   CONTINUE
      GAM2(N)=GAM2(N)-CC(N)*BB(N)
C
C     BACK SUBSTITUTION
C
      FF(N)=AA(N)
      BB(N)=GAM2(N)
      DO 20 K1-N2,N
      K=N1N-K1
      K2=K+1
      FF(K)=AA(K)-CC(K)*FF(K2)
      BB(K)=GAM2(K)-CC(K)*BB(K2)
 20   CONTINUE
C
      K1=N+1
      ZAA=FF(K1)-CC(K1)*FF(N1)-AA(K1)*FF(N)
      ZAA=ZAA/(BB(K1)+AA(K1)*BB(N)+CC(K1)*BB(N1))
      FF(K1)=ZAA
      DO 30 K=N1,N
      FF(K)=FF(K)+BB(K)*ZAA
 30   CONTINUE
C
      FF(N+2)=FF(N1)
      RETURN
      END
```

5.7 Time integration

The classic, fourth-order Runge-Kutta (RK4) method is used for the time integration of the differential equations. This method achieves a high order of accuracy (error of order $\frac{\delta^5}{5!}$) by evaluating the right-hand side four times in the range between $t = n\delta t$ and $(n+1)\delta t$. The extra computational effort is offset by an extra order of accuracy gained by symmetry. The equation describing the evolution of the state variable x_s using the RK4 method is

$$x_s^{n+1} = x_s^n + [k_0 + 2k_1 + 2k_2 + k_3]/6 \tag{5.65}$$

where

$$\begin{aligned} k_0 &= \delta t f(\vec{x}^n, t^n) \\ k_1 &= \delta t f(\vec{x}^n + \frac{1}{2}k_0, t^n + \frac{1}{2}\delta t) \\ k_2 &= \delta t f(\vec{x}^n + \frac{1}{2}k_1, t^n + \frac{1}{2}\delta t) \\ k_3 &= \delta t f(\vec{x}^n + k_2, t^n + \delta t) \end{aligned} \tag{5.66}$$

and $dx_s/dt = f(\vec{x}, t)$.

To demonstrate the accuracy and stability of the fourth-order Runge-Kutta method,

consider the one-dimensional diffusion equation,

$$\dot{T} = D_T \nabla_{xx} T \tag{5.67}$$

solved on a discrete grid with the initial condition,

$$T_i(0) = \begin{cases} A & i = k \\ \\ 0 & otherwise. \end{cases} \tag{5.68}$$

The second derivative is approximated by second order centered-difference

$$\dot{T}_k \approx D_T \frac{T_{k-1} - 2T_k + T_{k+1}}{\Delta x^2} \tag{5.69}$$

so that at $t = 0$ we have,

i:	...	k-1	k	k+1	...
	—O—	—O—	—O—	—O—	—O—
T:	0	0	A	0	0
$\dot{\text{T}}$:	0	ΓA	$-2\Gamma A$	ΓA	0

where $\Gamma \equiv \frac{D_T}{\Delta x^2}$. By applying $f(T_k) = \Gamma(T_{k-1} - 2T_k + T_{k+1})$ to Equations 5.66, for the

initial condition

$$
\begin{aligned}
k_0 &= -2\delta t\Gamma T_k^0 \\
k_1 &= -2\delta t\Gamma T_k^0 + 2(\delta t\Gamma)^2 T_k^0 \\
k_2 &= -2\delta t\Gamma T_k^0 + 2(\delta t\Gamma)^2 T_k^0 - 2(\delta t\Gamma)^3 T_k^0 \\
k_3 &= -2\delta t\Gamma T_k^0 + 4(\delta t\Gamma)^2 T_k^0 - 4(\delta t\Gamma)^3 T_k^0 + 4(\delta t\Gamma)^4 T_k^0
\end{aligned}
\tag{5.70}
$$

Equation 5.65 becomes

$$
T_k^+ = T_k^0 \left[1 + \gamma + \frac{1}{2}\gamma^2 + \frac{1}{6}\gamma^3 + \frac{1}{24}\gamma^4 \right] \tag{5.71}
$$

where $\gamma \equiv -2\delta t\Gamma = \frac{-2D\delta t}{\Delta x^2}$, which is obviously the series expansion of $e^{\gamma t}$ ($\gamma < 0$) with a truncation error of $O(\delta t^5)$.

The future value is calculated by multiplying the current value by the factor

$$
F(\gamma) \equiv 1 + \gamma + \frac{1}{2}\gamma^2 + \frac{1}{6}\gamma^3 + \frac{1}{24}\gamma^4. \tag{5.72}
$$

The condition for stability require that $|F(\Omega\delta t)| < 1$ for the characteristic value $\Omega < 0$ (see, for example, reference [70]). MATHEMATICA is used to solve the roots of $F(x) = 1$. Along the real axis, the range for stable $\Omega\delta t$ is [-2.78,0] for this fourth-order method. The

eigenvalues for the implicit centered-difference approximation of the diffusion equation are

$$\Omega_j = \frac{-4D_T}{\Delta x^2}\sin^2(\frac{\pi j}{N}) \qquad j = 1, \ldots, N, \tag{5.73}$$

which leads to the condition on the time step,

$$\frac{D_T}{\Delta x^2}\delta t < \frac{2.78}{4}. \tag{5.74}$$

Chapter 6

Validation of the method

Validation of the method is accomplished by simulation of simple, two-dimensional convection for compressible fluids. The results for simulations requiring reasonable computer effort are, in general, extrapolated to the continuum limit in order to minimize the effect of discretization error. The simulations and analysis of these convecting system for steady and time-dependent, including turbulent-like, flows are compared to experimental results and simulations of similar systems reported by other researchers.

6.1 Description of system

The system that is used for validation is particularly simple, but still preserves the essence of a real fluid. For all simulations, an ideal-gas equation of state is used with constant transport coefficients, η, η_v, κ. To make the connection with a three-dimensional Stoke's fluid, the bulk viscosity (of the two-dimensional fluid) is set to a third of the shear

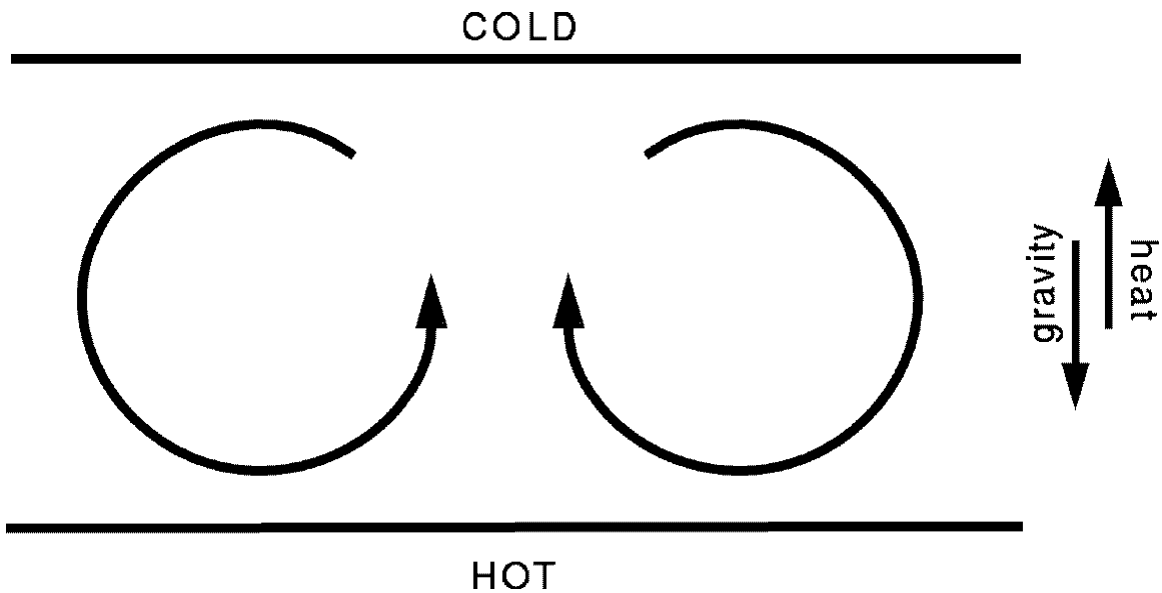


Figure 6.1: Simple diagram of two-dimensional steady Rayleigh-Bénard convection.

viscosity, as discussed in Section 3.1.2. The Prandtl number is set to unity, which is close to the value for many common gases such as Helium gas ($0.65 \leq Pr \leq 1.5$), and is consistent with kinetic theory predictions for a dilute two-dimensional gas (Pr=0.67). Simple kinetic theory also predicts that the ratios of the molecular constants do not change much from substance to substance or with temperature. The units are in terms of the Boltzmann constant ($k_B \equiv 1$). The body force (gravity) is set so that the change in potential and thermal energy for a small volume of fluid moving from the lower to the upper boundary is equal ($g = k_B \Delta T / \rho L$).

Periodic boundaries are used on the sides. The top and bottom boundaries are rigid (normal velocity vanishes), non-slip (tangential velocity of the fluid near the wall approaches the velocity of the wall, $\vec{u}_{wall} = 0$), and isothermal. The grid spacing dx is set to unity so that the height is just equal to the number of grid spacings. The resolution,

for a particular simulation, is defined as the number of grid points in one direction. For these simulations, the number of grid points in the vertical direction is used for plots. The simulated cell has a height L and a width equal to 2L (aspect ratio equals 2). Generally, the hot boundary has a temperature of 1.5 and the cold boundary is at 0.5. The average density is set to one. The Rayleigh number, with these simplifications, is defined by the temperature difference, height, and transport coefficients ($Ra = \frac{\Delta T^2 L^2}{\eta\kappa}$). Usually the value of the transport coefficients is changed, leaving Pr=1, to set the Rayleigh number, but the height and temperatures can also be used.

6.2 Extrapolation to continuum limit

The results from simulations with different resolutions can be extrapolated to the continuum (zero-mesh-size) limit. It would be too expensive to continually increase the resolution to lessen the global error. A prediction of a particular quantity without any error due to discretization can be made by estimating $\lim_{N\to\infty} f(x)$, where f is the quantity of interest whose error scales with the grid resolution. Figure 6.2 shows the horizontal kinetic energy, per unit mass, for simulations at $Ra = 20,000$. The results for various resolutions are plotted as a function of N^{-4}, where N is the number of vertical nodes. This nearly-linear relation can be used to estimate the zero-mesh-size value by determining the y-intercept of a least-squares fit or a carefully drawn line. Because of the slight curvature, it is best to extrapolate with a line through the two results with the greatest resolution.

The global error for a run with a particular resolution is defined as the absolute differ-

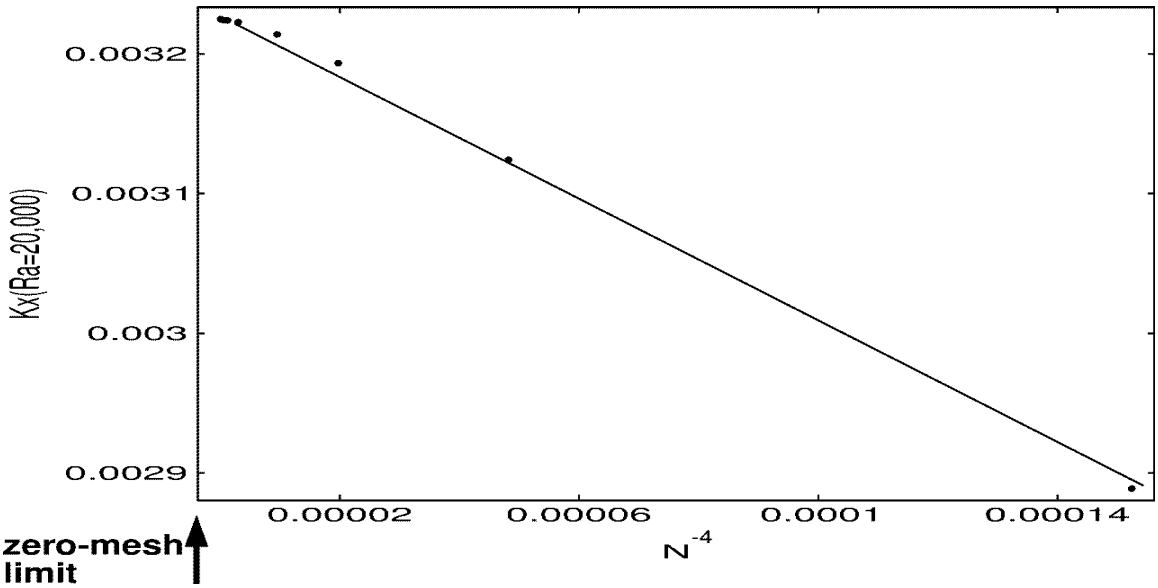


Figure 6.2: Results extrapolated to the continuum limit.

ence between the result and the continuum value.

$$\epsilon(\vec{x}; N) \equiv |f(\vec{x}; N) - f_\infty(\vec{x})| \tag{6.1}$$

The continuum value may be approximated by the extrapolated value or by the result of a high resolution run. The power-law relating the global error to the grid spacing can be determined using the estimated continuum value. Figure 6.3 shows that the global error for estimating the total horizontal kinetic energy decreases as the fourth power of the grid resolution. The slope of the log-log scale is determined by least-squares regression of the data.

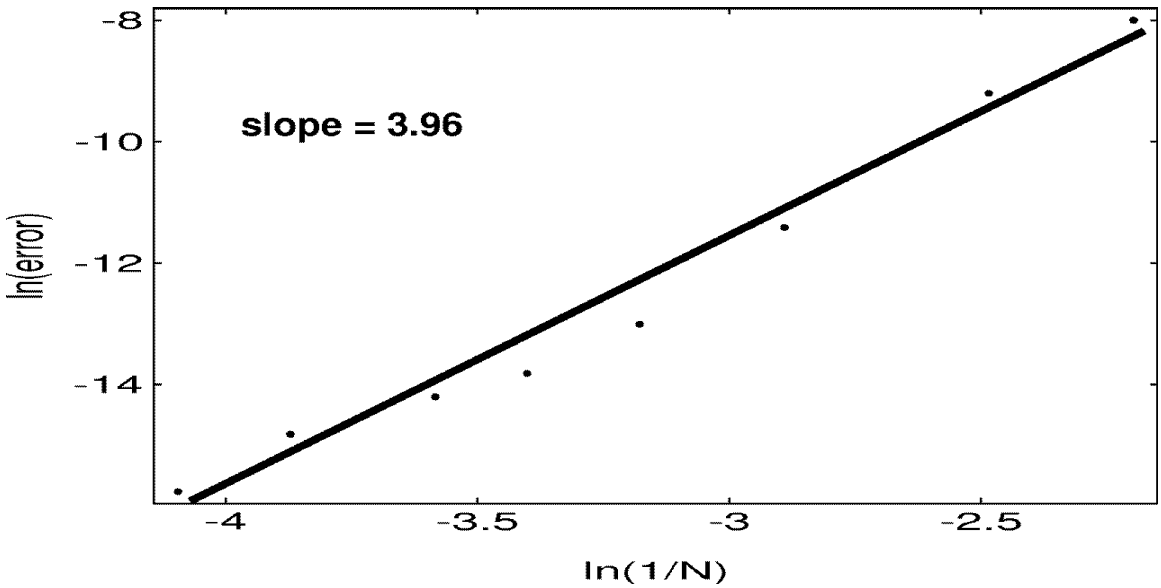


Figure 6.3: Global error in calculation of the horizontal kinetic energy. A least-squares fit through this data shows that the error scales linearly with the fourth power of the grid resolution ($1/N$). The data is for systems of height (left to right) 60,48,36,30,24,18,15,12,and 9.

6.3 Validating results

Results from simulations that reproduce results from experiments or other simulations are used to show that the method described here, along with the simplifications used, is valid for modeling convection, both time- dependent and independent. The next few sections briefly describe these validating simulations for steady to turbulent convection. The details appear in the later chapters of this Dissertation.

6.3.1 Steady flow

Steady (time-independent) convection is the simplest to simulate numerically. Both the cubic spline method and second-order centered finite-difference methods are used suc-

cessfully to simulate the steady flow. Steady results are obtained, with double-precision accuracy, within a thousand sound-traversal times. A variety of steady simulations were conducted including those demonstrating the coexistence of two-, four-, and six-roll solutions at the same Rayleigh number.

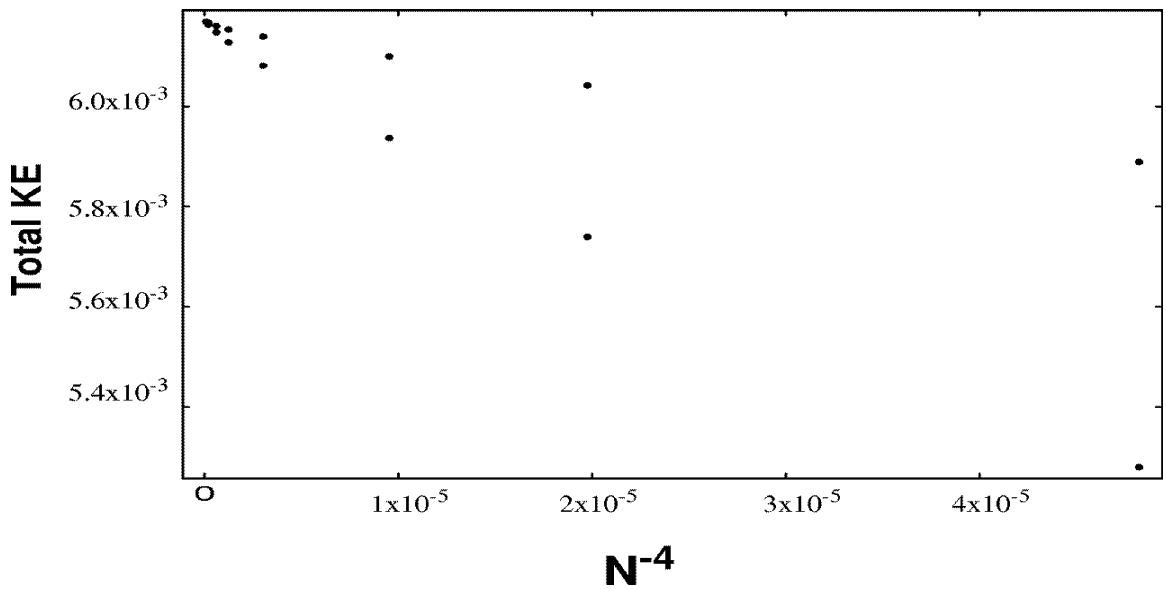


Figure 6.4: Comparison of calculation of a global quantity at various grid sizes for the cubic spline (upper set) and centered difference method (lower set). The cubic spline method, for a given resolution has less error.

The results from both methods, extrapolated to the continuum limit, agree within 1%. Figure 6.4 gives the total kinetic energy per unit mass for simulations based on the cubic spline method and the centered finite difference method. Although they both converge to the same value, the cubic spline method is more accurate for a given grid resolution.

Figure 6.5 shows the calculations of the maximum vertical velocity for steady flows at $Ra = 20,000$ for various grid sizes using both the cubic spline method and the second-

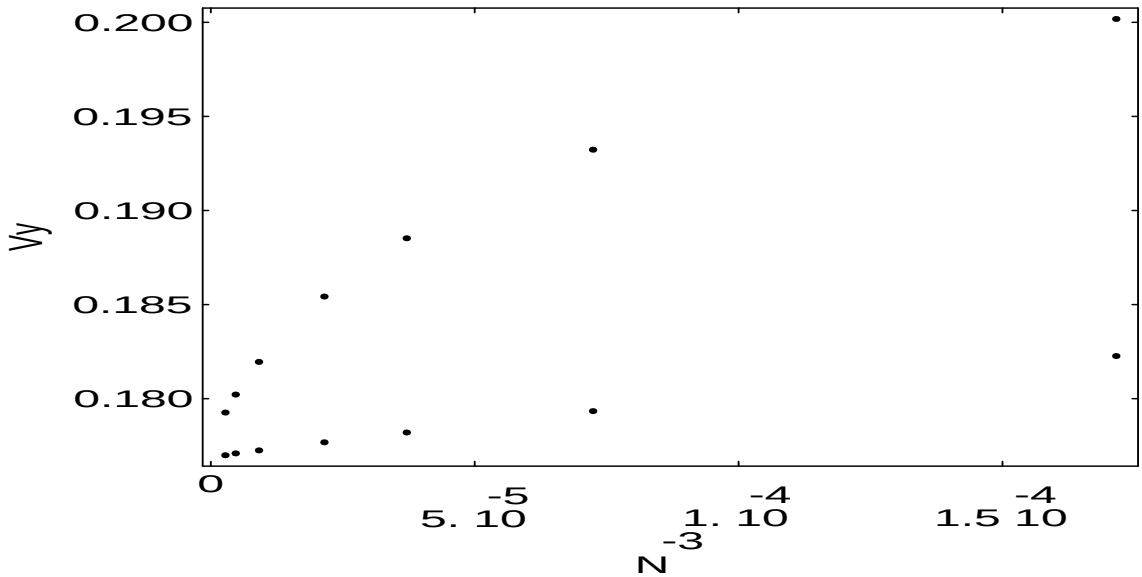


Figure 6.5: Comparison of calculation of a local quantity at various grid sizes for the cubic spline (lower set) and finite difference method (upper set). The cubic spline has a faster convergence rate than does the finite difference method.

order centered finite difference method. The error in the calculation of this *local* quantity for the cubic spline method decreases as the third power of the number of grid points in one direction.

For steady flows, the cubic spline method agrees with the centered finite-difference method, but does not offer a great advantage. The strength of the cubic spline method is in simulations of high Rayleigh number flows where the finite-difference method fails. Steady convection is discussed in more detail in Chapter 7.

6.3.2 Time-dependent flow

As the Rayleigh number is increased, the flow becomes time-dependent. That is, the state (mass density, velocity, and internal energy) at a particular point changes with

time. The first type of time-dependent convection, referred to as harmonic, oscillatory, or periodic, has a simple sinusoidal time profile for the observables, such as the heat flux. Here, periodic convection occurs when $Ra = 80,000$ and at a cycle time corresponding to 0.2 of the thermal diffusion time. This transition point and cycle time is consistent with *ab initio* molecular dynamics simulations in a slightly different two-dimensional geometry. Consistency with molecular dynamics simulations is desirable because MD simulations are inherently compressible.

As the system is driven further from equilibrium the system makes a transition to chaotic convection. Our simulations of a two-dimensional ideal gas show that there is a drop in the Nusselt number associated with this transition. This is the result of well-established simulations where transient effects are small. The transition and the corresponding drop in the dimensionless heat flux occurs between a Rayleigh number of 200,000 and 300,000. This is consistent with experiments with liquid mercury (a nearly incompressible fluid with a low Prandtl number) and gaseous helium conducted by different researchers. The temperature histogram at a point in the center of the simulated cell makes a transition from a dual-maximum to a single-maximum as the system goes from periodic to chaotic. This anomaly was also reported for the mercury experiment. Periodic and chaotic convection will be discussed in detail in Chapter 8.

6.3.3 Turbulent flow

For large Rayleigh numbers ($Ra \sim 10^8$), the simulations describe turbulent-like flows, even though they are two-dimensional. Many characteristics of turbulent convection are observed. First, there is vorticity on many length scales – from the length scale of the system to that of a few cell widths. In these simulations, the smallest vorticity length scale observed is about $1/25^{th}$ of the system size. A mechanism for the generation of vorticity on these smaller scales is the convection instability itself. That is, on a length-scale smaller than the length of the system, the temperature difference may be great enough to induce local convection. The shear from the fast, downward-moving cold plume and the slower, upward-moving warm plume also induces vorticity in the fluid near the plumes. Other qualitative agreement with experiment and other simulations include the asymmetry in the thermal plumes and the presence of a thermal and viscous boundary layer.

Quantitative agreement can be seen in a plot of the dimensionless heat flux from convection $Nu - 1$ verses the Rayleigh number of the system. The well-known scaling relation, $Nu - 1 \sim Ra^{2/7}$, is recovered. This scaling relation is seen in experiments and other simulations, contradicting scaling arguments predicting a $\frac{1}{3}$ power-law. Details of the turbulent-like convecting flow simulations are detailed in Chapter 9.

Part III

Applications

Chapter 7

Steady compressible convection

Steady state convection is the simplest type of heat transport in a fluid after pure conduction. The state variables – velocity, mass density, and internal energy – at a particular point, are constant in time. In the two-dimensional model described here, steady convection is observed for systems with Rayleigh numbers from a few thousand to about 80,000. The exact value for the critical Rayleigh number – the value at which pure conduction is unstable in favor of convection – was not investigated here. This has been investigated exhaustively for incompressible flows $[3, 9, 71]$ and for compressible flows $[41, 10]$. Figure 7.1 shows the velocity, temperature, and density fields for a steady convecting system at $Ra = 20,000$. The velocity magnitude ranges from 0 to $0.177c$, where c is the sound speed, and vanishes at the non-slip boundaries. The temperature, in this plot, goes from $1.5T_0$ (where $T_0 \equiv 1$), at the bottom, to $0.5T_0$, at the top. This temperature range is typically used in the simulations because $\Delta T = \bar{T}$. The density

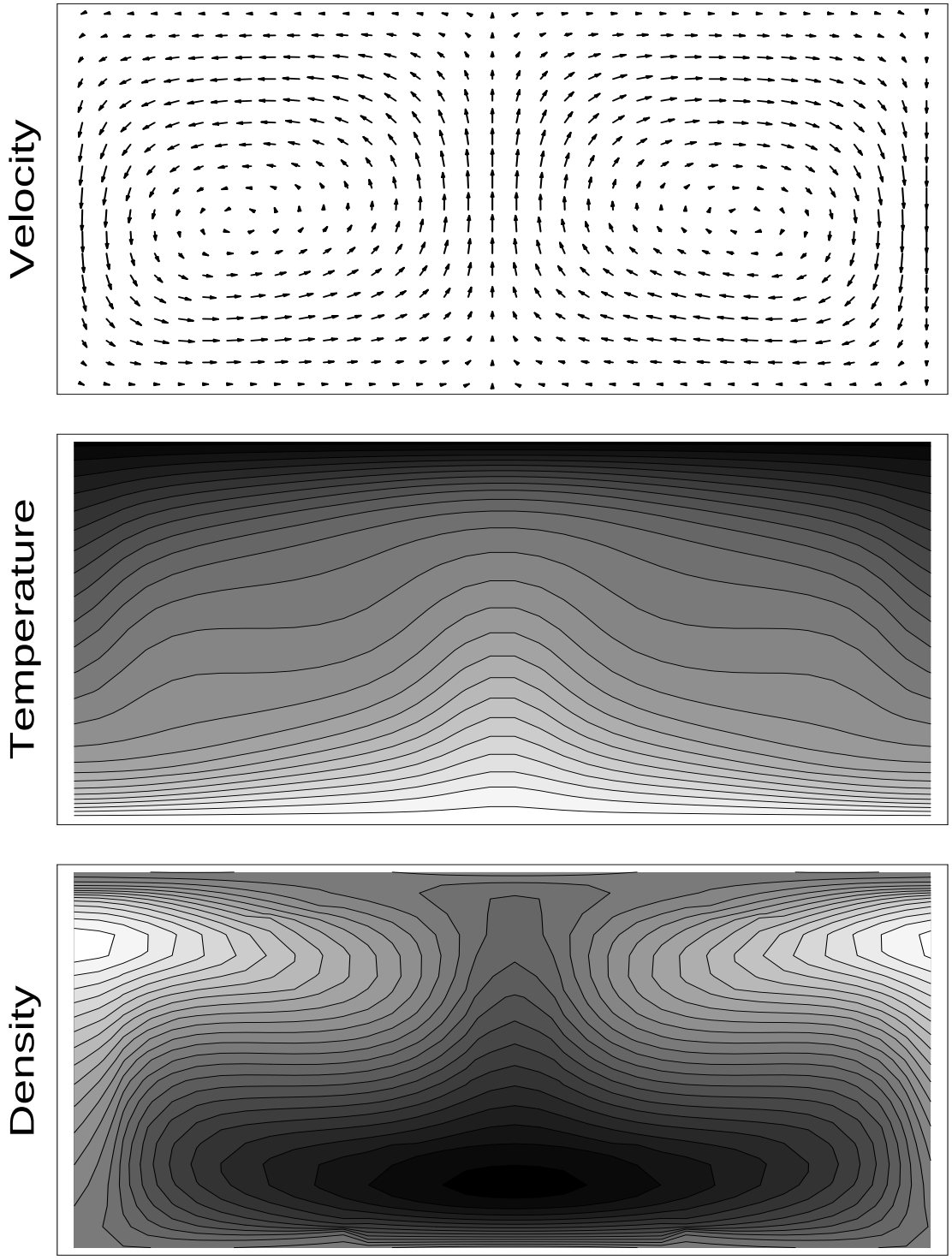


Figure 7.1: Velocity, temperature, and density fields for steady compressible convection at $Ra = 20,000$.

varies from $0.798\rho_0$ in the center-bottom to $1.26\rho_0$ at the top-sides. This deviation from the equilibrium density ($\rho_0 = 1$) is a measure of the compressibility of the fluid.

7.1 Coexisting solutions for steady flow

In the course of these investigations we found that solutions of the full set of continuum equations can be degenerate over a wide range of conditions. The coexistence of two stable stationary flows, for the same boundary conditions, was first discovered by accident, watching both develop from "random" initial conditions. Thus the computational situation is not so different from the experimental one, in which roll patterns have been observed to change on a time scale of days [10]. Of the three flows shown in Figure 7.2, the two-roll flow in a box of aspect ratio two is very well known. The incompressible approximation to this case has been exhaustively discussed by Chandrasekhar [9], Clever [22] and Busse [21, 23].

Busse went on to explore the stability of roll patterns (for incompressible fluids) of various wavenumbers, α, where $\alpha = \pi$ corresponds to the two-roll pattern. The "Busse Bubble" refers to the region of wavenumber-Rayleigh number space for which roll patterns are stable. This stability region is centered near $\alpha = \pi$ but covers a range so small as to suggest that the four-roll pattern, $\alpha = 2\pi$, is not stable at any Rayleigh number. Puhl and coworkers [10] presented work based on a finite-difference model of the compressible hydrodynamics equations that indicated that both one- and two-roll solutions can coexist for the same system. Additionally, they used a stability model based on the linearized

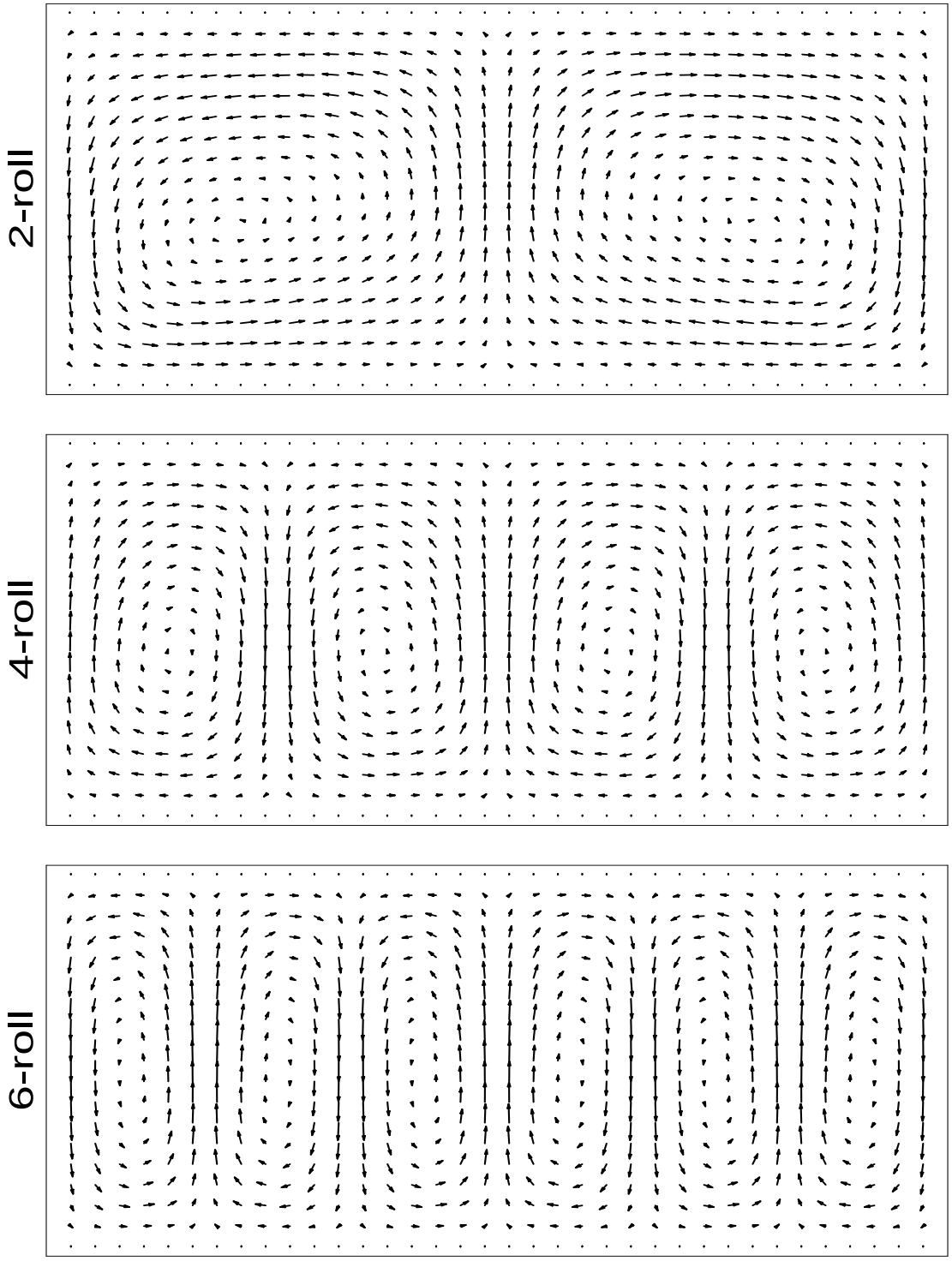


Figure 7.2: Coexisting solutions (2-, 4-, and 6-roll) for compressible convection at $Ra = 40,000$.

hydrodynamic equations to support the existence of one-, two-, and three-roll patterns for a system with an aspect ratio equal to two. This was for a box with four rigid sides and stress-free boundaries (otherwise odd numbers of rolls would not be possible).

Unlike Busse's incompressible flows, the present work uses a fully compressible model, a natural choice for the study of buoyancy-driven convection. The results discussed here correspond to wavenumbers of $\{\pi, 2\pi, 3\pi\}$. In testing the robust nature of the two-roll solution, a four-roll solution would sometimes appear. Further testing showed that four-roll solutions persist, as do the two-roll ones, as the number of degrees of freedom describing the system is made arbitrarily large (at fixed Rayleigh and Prandtl numbers) for times greater than 10^4 sound-traversal times. In addition to showing that these patterns exist for long times the internal energy per unit mass, the heat flux, and the approximate linear growth rate is quantitatively characterize for all the patterns. All of these results are extrapolated to the continuum limit.

In three dimensions two roll patterns formed in a diagonal orientation, corresponding to a roll width intermediate between the two-roll and four-roll cases. There is also a fairly long-lived six-roll solution, still with exactly the same boundary conditions; see again Figure 7.2. This six-roll solution is stable to a catastrophe time equal to 470 sound traversal times, after which the system evolves to either a two- or four-roll pattern. The approximate unpredictability of the final state suggests a complicated, perhaps fractal, boundary separating these attracting states. Additionally, persisting non-stationary, mixed-mode limit cycles have been observed to form after the catastrophe. Similar results hold in

two and three dimensions, though comprehensive tests of the numerical stability in the latter case would severely strain the limits of current computers. The methods used to obtain these solutions and present evidence that degenerate compressible solutions persist in the continuum limit are described here. The conclusions from this work are discussed in Chapter 10.

A variety of initial conditions are used for this study. If the nodal velocities are chosen randomly the corresponding kinetic energy dies out rapidly except for one or more unstable modes, which grow and lead to a finite-amplitude stationary state, of the type shown in Figure 7.2. Solutions with a particular desired symmetry can be constrained to retain that symmetry in the velocity field until the other state variables converge such that fluctuations from the stationary state become numerically insignificant. Then, the constraints can be released, and the sensitivity to computational noise studied. By studying the mesh-dependence of this sensitivity, a variety of stable solutions were discovered.

Nodal velocities for typical solutions are plotted in Figure 7.2 for a Rayleigh number of 40000, about 23 times the critical value for two-roll convection. These solutions exhibit deviations from the fully-converged continuum limit which vary smoothly with the mesh spacing. Either extrapolation or the use of a relatively large number of grid points makes accurate results possible. The tabulated results are based on series of simulations using $2H^2$ cells, mostly with $H = \{36, 48, 60, 72\}$. See Table 7.1.

For the three states shown in Figure 7.2, continuum values are estimated for the horizontal and vertical kinetic energies and the internal energy per unit mass, the heat flux,

Table 7.1: The horizontal and vertical contributions to the kinetic energy per unit mass, the internal energy per unit mass, the boundary heat flux, and the small-amplitude growth rate of the kinetic energy, multiplied by the system width W are given for a series of fully-converged solutions at a Rayleigh number of 40000. Two-roll, four-roll, and six-roll solutions are compared. $W = 2H$. $N = 2H^2$. In the continuum limit energies vary as N, traversal times, diffusion times, and dissipation rates as $N^{1/2} \propto H$.

Rolls	K_x/Nm	K_y/Nm	E/Nm	$Q_{boundary}$	W/τ
2	0.003730	0.00357	1.014	0.0120	1.42
4	0.001139	0.00410	1.018	0.0118	1.70
6	0.000274	0.00226	1.012	0.0106	1.25

and the approximate linear growth rate associated with the three types of roll patterns.

The last two points require further discussion. In the stationary state, the horizontal average of the vertical energy flow must be constant, independent of the vertical y coordinate and equal to the rate at which heat enters (and exits) the system through the bottom (and top) boundaries.

The energy flow per unit area, including convection, but equal to the heat flux at the boundary can be computed as follows:

$$\begin{aligned}\vec{q}_{tot} &= -\kappa\nabla T - \vec{u}\cdot\overline{\overline{\sigma}} + \rho\vec{u}(e + \frac{1}{2}\vec{u}\cdot\vec{u}) \\ &= \left[-\kappa\frac{\partial T}{\partial x} - u\sigma_{xx} - v\sigma_{xy} + \rho u e + \frac{\rho u}{2}(u^2+v^2)\right]\hat{i} \\ &+ \left[-\kappa\frac{\partial T}{\partial y} - u\sigma_{xy} - v\sigma_{yy} + \rho v e + \frac{\rho v}{2}(u^2+v^2)\right]\hat{j}.\end{aligned} \tag{7.1}$$

The total energy per unit time through the boundary is

$$Q_{tot} \equiv \oint_A \langle\vec{q}_{tot}\rangle\cdot d\vec{A}. \tag{7.2}$$

Numerical estimates are included in Table 7.1.

The linear growth rate of the rolls' kinetic energy is accurately proportional to $1/W$ as the time and length scales vary proportionally. Rates were estimated by first determining the stationary roll-state deviations from the quiescent state. The deviations, multiplied by a small number, were then used to perturb an otherwise quiescent initial state. Growth rates could be obtained in this way with uncertainties of one or two percent.

Using this problem to test various numerical methods, we were able to confirm the reproducibility of the results and also reproduced some sample results given in Reference [63]. Though the programs differed slightly for finite meshes, due to the ambiguities in differencing mentioned in Chapter 4, the independent programs agreed within 1% of the results given in the Table. In appropriate special cases our two- and three-dimensional programs agreed to the full 64-bit precision used in these simulations.

Similar roll-type flows were investigated with simulations with Smooth Particle Applied Mechanics [72, 60, 14]. The three-dimensional SPAM work reported here is new. The smooth particle results are more complex because SPAM simulations, like molecular dynamics, fluctuate forever, and cannot reach true stationary states. In many situations the convective rolls and cells can come and go, and change orientation, in times which are quite long relative to a typical roll-rotation time. Though the stability of such two-roll patterns is well known in simulations, with both molecular dynamics [73, 12] and smooth particle applied mechanics [14], the four-roll and six-roll solutions reported in this present work remain subjects for continuing investigation in three dimensions.

7.2 Min-max internal entropy production principle

Struchtrup and Weiss [74] (S&W) have proposed a new variational Principle for stationary nonequilibrium flows, such as the steady boundary-driven flows of mass and heat shown in Fig. 7.3. S&W suggest that of all the possible flows satisfying the specified boundary conditions, the solution with smallest maximum local entropy production is correct. They suggest that their min-max Principle could be extended to non-stationary flows. S&W considered the one-dimensional heat flow of a Boltzmann gas. They calculated the local entropy production and expressed its maximum as a function of the dimensionless heat flux, $\hat{q}$. They showed that the heat flux corresponding to the minimum of the local entropy production also corresponds to the solution of this linear transport problem.

Despite their success with this simple example problem, more-complex, nonlinear problems are required to test any assertion of generality for such a Principle. Three potential limitations for the validity of the Principle are the following: (i) the Principle ignores the possible coexistence of stable multiple solutions for the same boundary conditions; (ii) the Principle fails to treat the overall stability of two disparate, but weakly coupled, nonequilibrium systems; (iii) the Principle excludes transitions to more stable states, such as the transition from quiescent Fourier conduction to the steady convective state which produces entropy at a greater rate $[3, 9]$. To make this point more explicit $[67]$, a set of stationary, convecting flows for a compressible, heat-conducting viscous fluid described by the Navier-Stokes equations is displayed The flows have identical boundary conditions with a Rayleigh number of $40,000$. The sides of the simulated cell are periodic and the

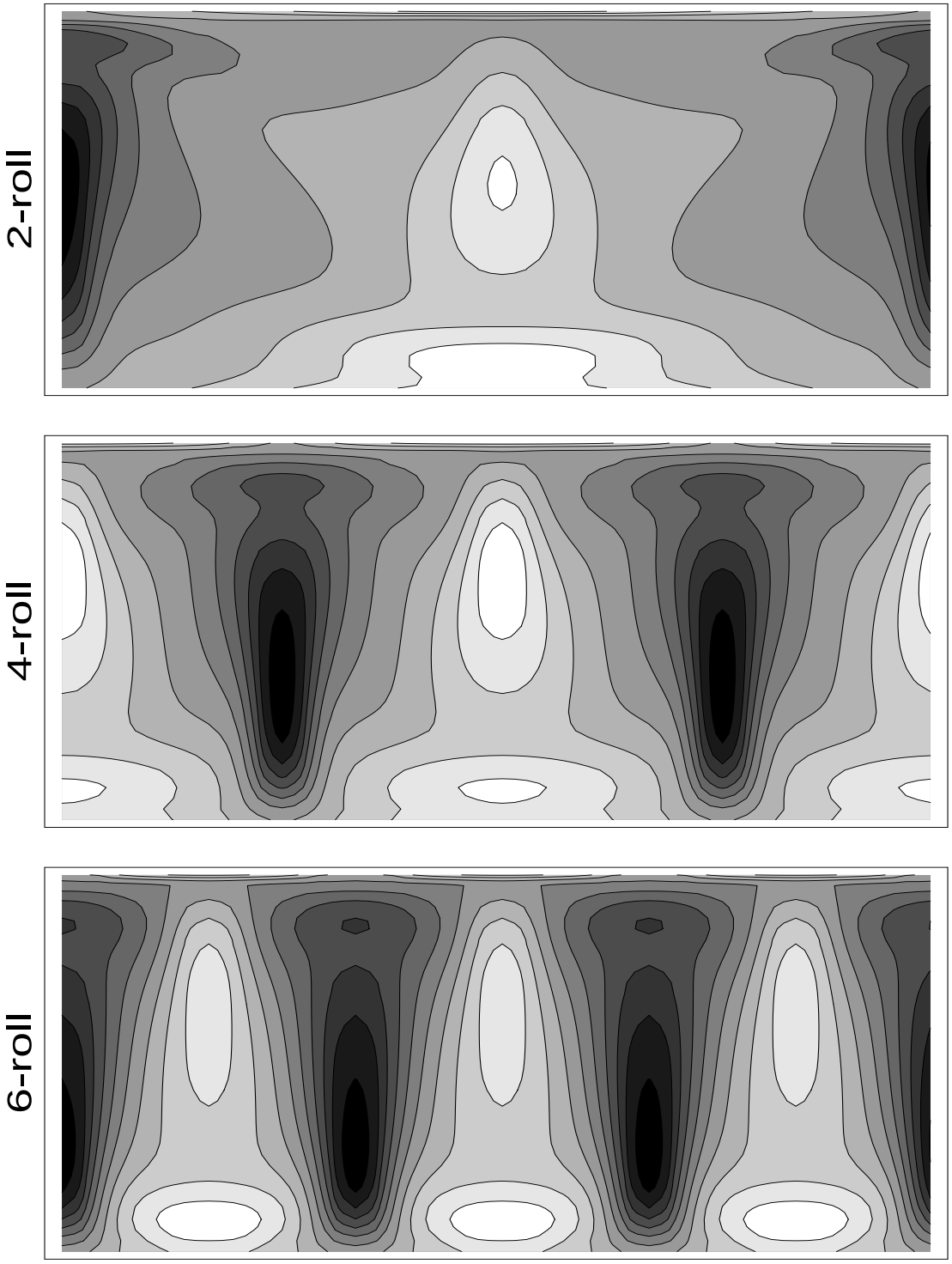


Figure 7.3: Internal entropy production density for coexisting solutions (2-, 4-, and 6-roll) at $Ra = 40,000$. White indicates regions of highest entropy production.

Table 7.2: Maximum internal entropy production density for the 2-, 4-, and 6-roll solutions for $Ra = 40,000$.

Rolls	Max(g_i)	stability
2	0.00703	stable
4	0.00624	stable
6	0.00555	unstable

width is twice the height. The 2- and 4-roll flows are stable attractors in the solution space while the more complex 6-roll solution is unstable. For details, see reference [15].

The local internal entropy production [75, 29, 67] for the compressible Navier-Stokes equations is

$$g_i = -\frac{\vec{q} \cdot \nabla T}{T^2} + \frac{\bar{\bar{\sigma}} : \nabla \vec{u}}{T}, \tag{7.3}$$

where $\bar{\bar{\sigma}}$ is the stress tensor and $\vec{q}$ is the heat flux. This expression for the local entropy production was validated by confirming that the integral

$$\dot{S}_i = \int_V g_i \, dV, \tag{7.4}$$

is precisely equal to the total flow of entropy at the boundaries,

$$\dot{S}_i = \dot{S}_e \equiv \frac{Q}{T_H} - \frac{Q}{T_C}; \qquad Q \equiv \int_A \langle \vec{q} \rangle \cdot dA, \tag{7.5}$$

where T_H and T_C are the temperatures of the hot and cold boundaries.

The maxima in the local entropy productions per unit volume $\max[g_i]$ for the (2, 4,

6)-roll flows are respectively 0.00703, 0.00624, and 0.00555. Thus two or more stable solutions (2- and 4-roll) exist, for the same boundary conditions but with different $\max[g_i]$. It is amusing that an unstable "solution", with six rolls, has the smallest $\max[g_i]$ of the three cases considered here.

Chapter 8

Harmonic and chaotic convection

This Chapter discusses time-dependent flow morphologies that are not quite turbulent. These include the laminar-harmonic and chaotic flows. A dual morphology region is discovered where both may coexist. Within this region, the heat flux, entropy production, and Lyapunov instability for flows with the same Rayleigh number are compared.

8.1 Transition to harmonic flow

At higher Rayleigh numbers, the convective transport of heat is time-dependent. The first type of time-dependent convection to appear with increasing Rayleigh number from steady flow is simple harmonic (also referred to as periodic or oscillatory) flow – in which observables, such as the heat flux, have a simple harmonic time dependence. The harmonic motion in this case is a result of the periodic up and down motion of the convective rolls. Figure 8.1 shows the sinusoidal time profile of the Nusselt number for harmonic convection

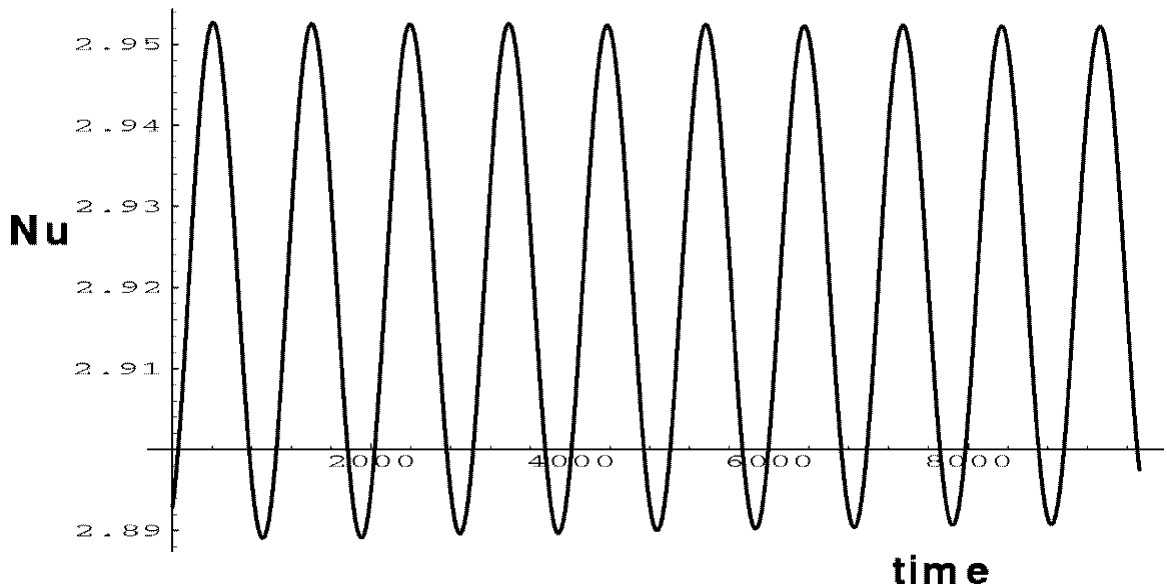


Figure 8.1: Sinusoidal time dependence of the Nusselt number for harmonic convection.

at $Ra = 80,000$. The cycle time for the heat flux is $0.2\tau_D$, where $\tau_D \equiv L^2/D_T$ is the thermal diffusion time. For this fluid, with unit Prandtl number, the thermal and viscous diffusion times are equal. With the simplifications described in Section 6.1, the diffusion time also equals $L\sqrt{Ra}$. In these simulations, harmonic convection commences near $Ra = 80,000$ and continues to about $Ra = 240,000$. The onset of harmonic convection is consistent with molecular dynamics (MD) simulations by Rapaport [12] who observed harmonic convection at $Ra = 78,000$ for a two-dimensional gas of hard spheres with $\rho_{avg} = 0.4$. The disks are confined in a box with an aspect ratio of one and rigid, no-slip boundaries on the top, bottom, and sides. The estimate of the transport coefficients used to determine the Rayleigh number is based on the Enskog formulas and serves as a rough estimate considering the relatively high density. Rapaport estimated the Prandtl number to be 0.45. The course-grained time dependence of the temperature at particular points

in the MD simulations has a periodic profile with a cycle time of $0.4\tau_D$. The agreement with the simulations discussed here is reasonable considering the differences in geometry, boundaries, and Prandtl number. Agreement with molecular dynamics simulations is important as these methods inherently allow for compressibility effects.

The results presented here also agree with two-dimensional simulations by Goldhirsch *et alii* [24] using the Oberbeck-Boussinesq approximation. They report results for different geometries (aspect ratios, rigid and periodic boundary conditions) for fluids with Prandtl numbers of 0.71 and 6.8. The most appropriate comparison is with results for a fluid with a Prandtl number of 0.71 in a cell with unit aspect ratio and $Ra = 100,000$ which showing a harmonic time profile with a cycle time ≈ 0.15. A similar result for a fluid with $Pr = 6.8$ has a cycle time of 0.21. Again, the side boundaries are rigid, but still provide good agreement. It is interesting that they did not observe aperiodic (chaotic) flows for $Pr = 0.71$.

8.2 Transition to chaotic flow

The simulations show that the harmonic convection regime continues to about $Ra = 240,000$. For high Rayleigh numbers, a different flow morphology is observed – where thermal plumes sweep back-and-forth.

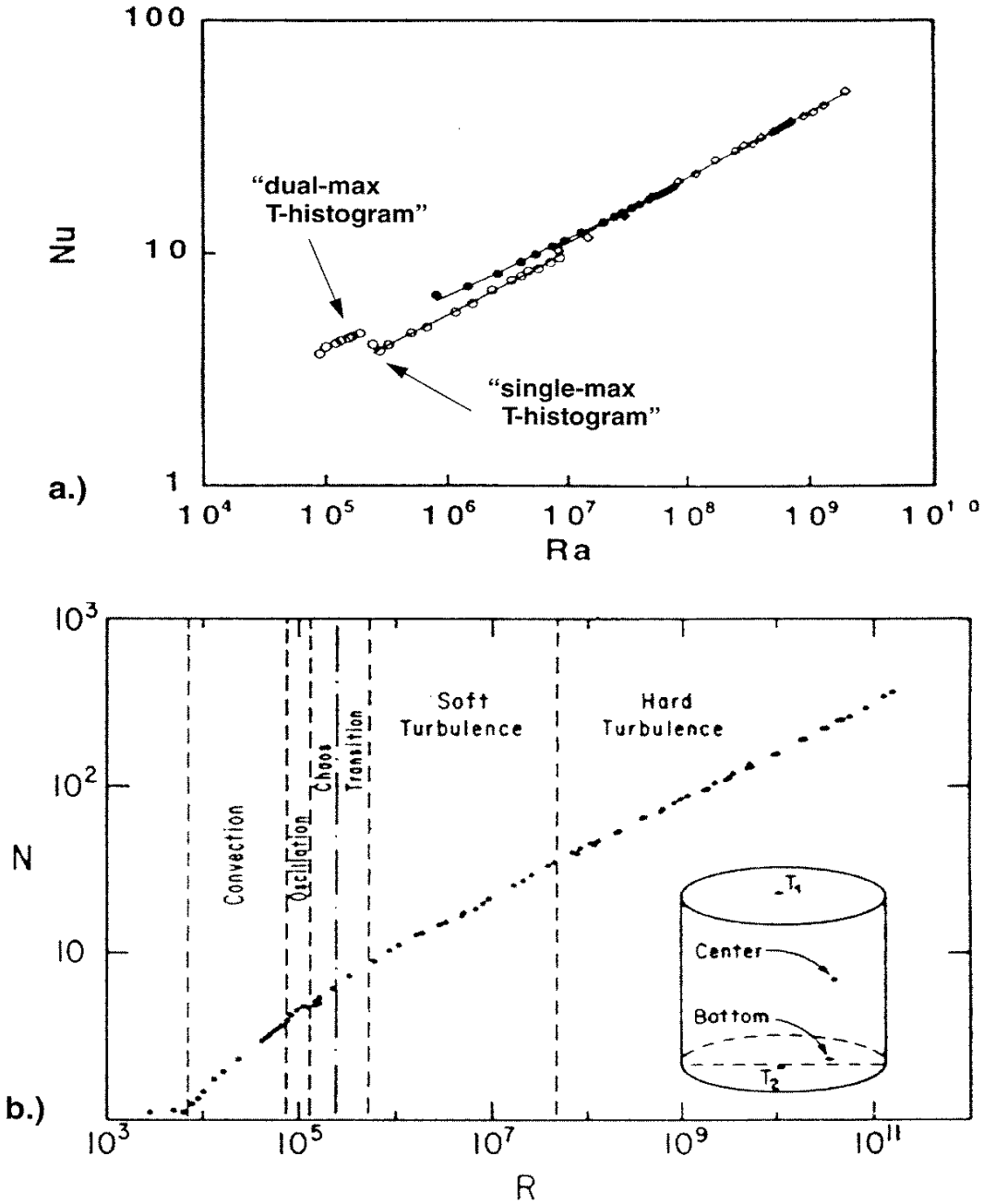


Figure 8.2: Experimental results for convecting a. liquid mercury and b. gaseous helium. Both systems show a drop in the Nusselt number near $Ra \approx 2.5 \times 10^5$.

Numerical simulations of the fully-compressible Navier-Stokes equations are used to study the transition from simple-periodic, "harmonic", to chaotic thermal convection as the Rayleigh number, Ra, is increased. The simulations suggest that a sharp discontinuity in the relationship between the Nusselt number, Nu (the ratio of the total heat flux to the Fourier heat flux), and the Rayleigh number, is associated with this transition in flow morphology [28]. This drop in Nusselt number is also seen in the data reported in independent experiments involving the convection of two characteristically different fluids - liquid mercury [37] (a nearly incompressible fluid with $Pr = 0.024$) and gaseous helium [20] (a compressible fluid with unit Pr). The harmonic flow generates a dual-maximum (quasi-harmonic) temperature histogram while the chaotic flow generates a single-maximum histogram at the center point in the simulated cell. This is consistent with the temperature distributions reported for the convecting mercury before and after the drop in Nu. Our simulations also suggest a hysteresis in the Nu-Ra curve linking the two distinctly different flow morphologies, heat fluxes, and temperature-fluctuation histograms at the same Rayleigh number. The timed-averaged Nusselt number (dimensionless heat flux) is plotted in Figure 8.3 for flows with $8 \times 10^4 \leq Ra \leq 6 \times 10^5$. Within the dual-morphology region, the Nusselt number for the two-frequency, periodic flow (higher) and the three-frequency, chaotic flow (lower) are joined by a hysteresis loop. The upper branch of the loop continues from the steady-state region into the periodic region. The lower branch is continues from the chaotic region to fully-developed turbulent convection. As the Rayleigh number is increased quasi-statically from a steady state, periodic convection

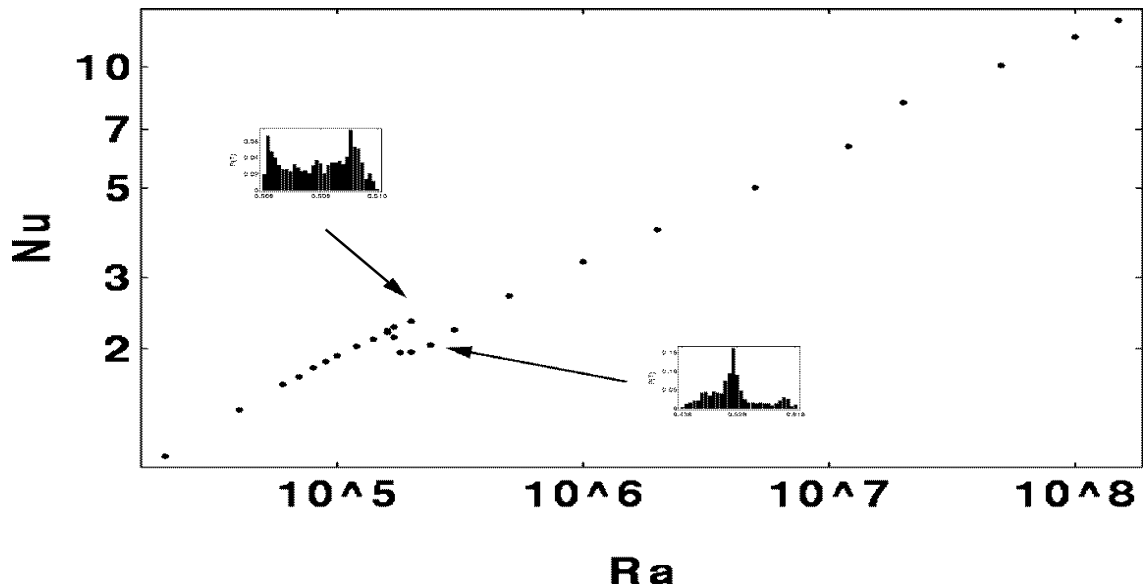


Figure 8.3: Simulation results showing the drop in Nusselt number at the transition from harmonic to chaotic convection. The temperature histograms for a point in the center of the simulated cell are shown for both types of flows.

develops first. As the Rayleigh number is increased further, the system makes a transition to chaotic flow. The Nusselt number, as seen in Figure 8.3, shows an initial drop along this path, followed by an increase with increasing Ra. The term "quasi-statically" is used to mean that a well-developed state at a given Rayleigh number is used as an initial condition for a run at a slightly different Rayleigh number. The simulation is allowed to run for several thousand sound traversal times to eliminate transient effects. The Rayleigh number can be varied in our simulations by changing either the transport coefficients (which also varies the diffusion traversal times but not the sound traversal time), the temperature of the hot boundary (which also varies the sound traversal time and the thermal expansion coefficient), or the length scale (which varies the diffusion and sound traversal times). In all three cases, a hysteresis is present, but when the length scale is varied, the range of

the dual-morphology region is decreased.

8.3 Coexisting flow morphologies

Our simulations of the fully-compressible Navier-Stokes equations of a viscous, heat-conducting fluid enclosed between two rigid, thermal boundaries in the presence of a body force reveal that, within a range of Rayleigh numbers, both laminar-like periodic flow and turbulent-like chaotic flow morphologies are stable solutions for the same Rayleigh number [29]. Figure 8.5 shows time sequences for the laminar-like (left hand side) and

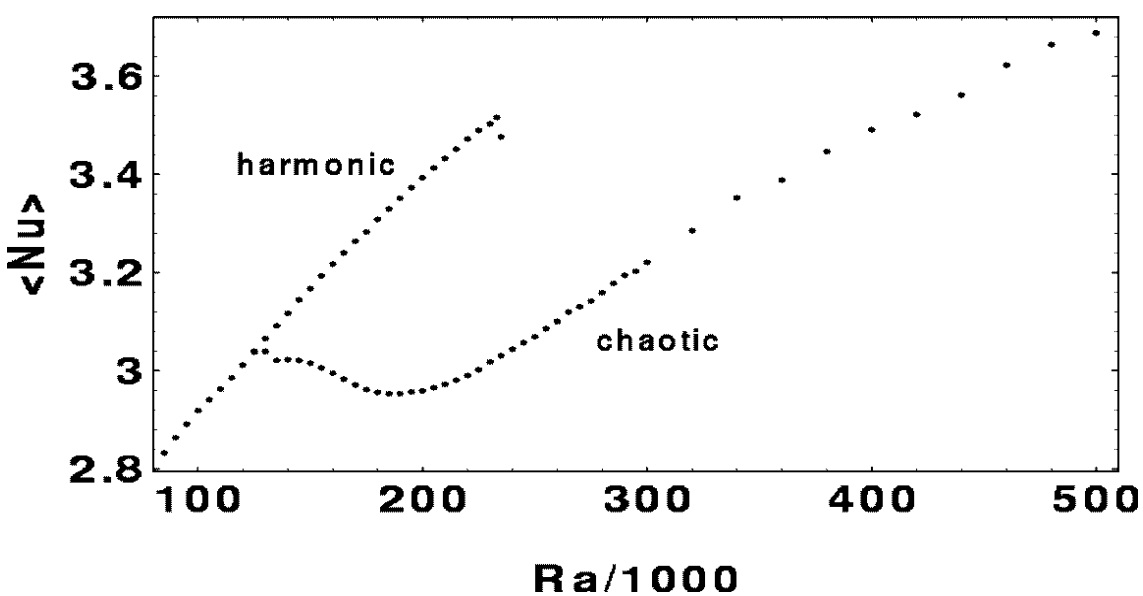


Figure 8.4: Nusselt number for harmonic and chaotic flows indicates a dual-morphology region where harmonic and chaotic flows may coexist.

chaotic (right hand side) flows – both at the same Rayleigh number, 2×10^5. The harmonic flow sequence shows the thermal plumes oscillating vertically. This frequency is

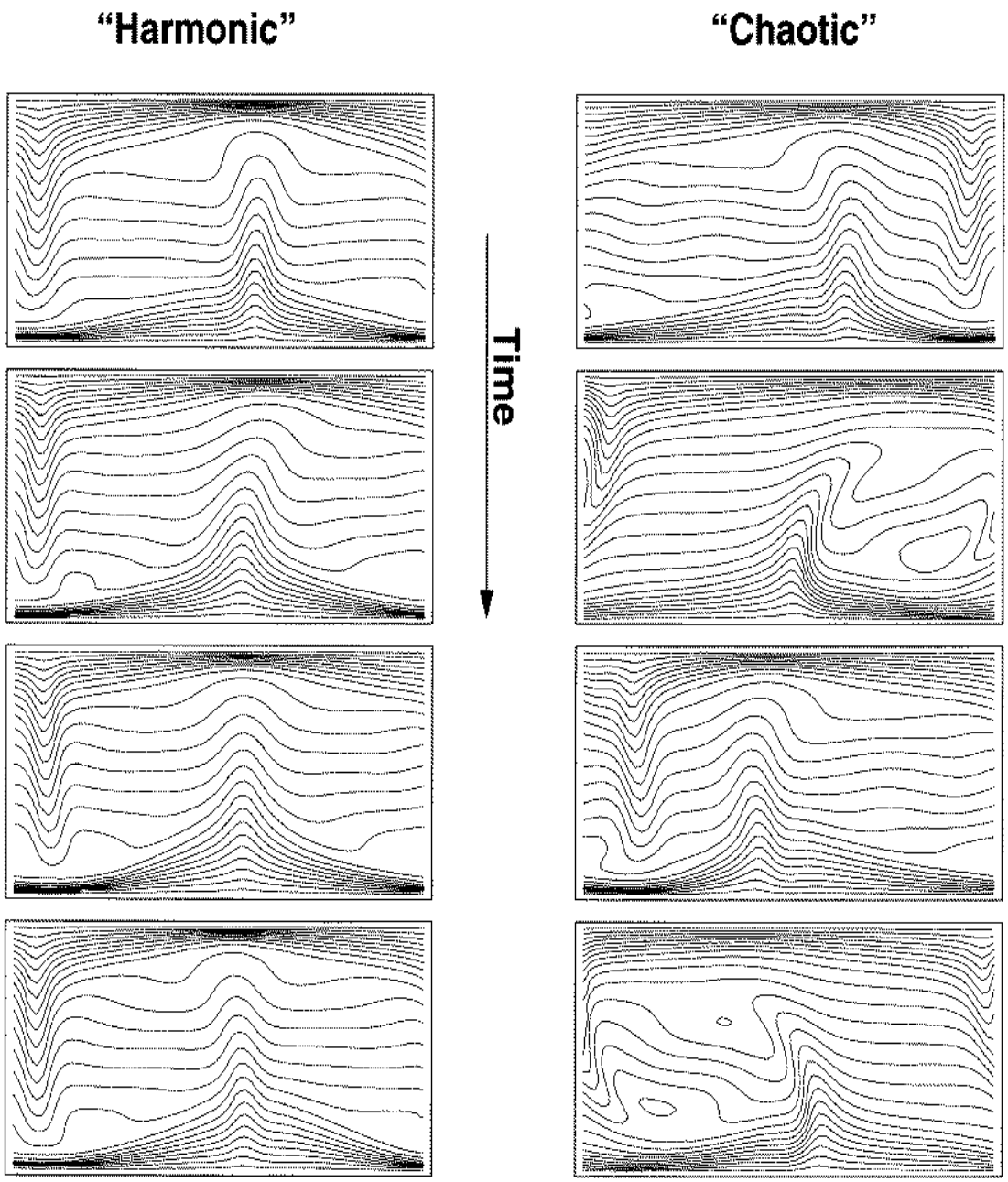


Figure 8.5: Temperature contours for harmonic and chaotic flows at $Ra = 200,000$. The time sequence appearing on the left shows the flow for "harmonic" convection.

in addition to the frequency of the fluid moving around the two counter-rotating convection cells. The harmonic flow is more effective in transporting heat. The chaotic flow has at least one additional characteristic frequency – that of the plumes sweeping from side to side, disturbing the thermal boundary layer on the opposite side. The horizontal sweeping near the opposite thermal boundary causes fluid flow in a direction counter to the net flow of heat, resulting in a less efficient heat transfer. The addition of a third incommensurate frequency is, according to Ruelle's idea of the route to chaos [76–78], enough to induce highly unstable chaotic motion.

8.3.1 Entropy production and the loss of phase space information

Because the temporally periodic flow transports more heat, it must likewise produce entropy at a greater rate. The total internal entropy production is the integral over the system volume of the internal entropy production per unit volume, g_i. According to local thermodynamic equilibrium with linear transport theory, g_i is defined

$$\dot{S}_i = \int_V g_i \, dV; \quad g_i \equiv -\frac{\vec{q} \cdot \nabla T}{T^2} + \frac{\bar{\bar{\sigma}} : \nabla \vec{u}}{T}, \tag{8.1}$$

where $\vec{q}$ is the heat flux and $\bar{\bar{\sigma}}$ is the stress tensor. This total internal entropy production can be made dimensionless by dividing it by the internal entropy production for the

equivalent quiescent system (with heat conduction only),

$$\dot{S}_{ND} \equiv \dot{S}_i/\dot{S}_F; \quad \dot{S}_F = -W \int_0^L dy \, \frac{\vec{q}_F \cdot \nabla T_F}{T^2}, \tag{8.2}$$

where $\vec{q}_F = -\kappa \nabla T_F$, $\nabla T_F = \vec{y}\Delta T/L$, and W is the width of the system. Figure 8.6

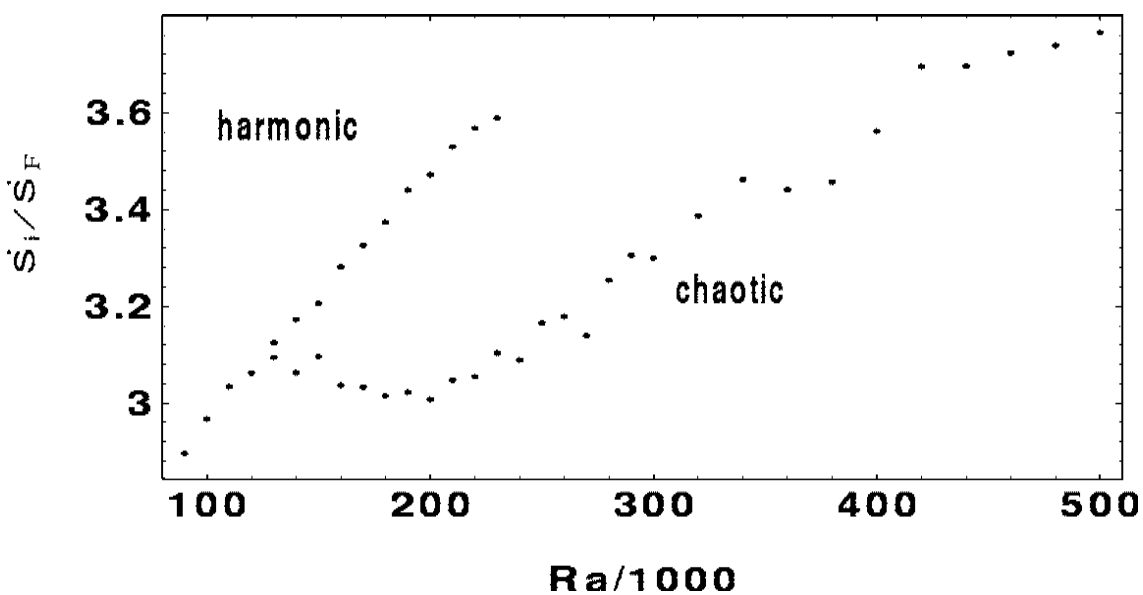


Figure 8.6: Dimensionless internal entropy production for the harmonic and chaotic flows.

shows the time-averaged dimensionless internal entropy production for various Rayleigh number flows including the dual morphology region. The dimensionless entropy production is equivalent to the Nusselt number (time-averaged quantities) in these simulations in the $\Delta x \to 0$ limit.

The total internal entropy production is calculated by starting with a well-established convection state and then averaging the volume integral for over 10^3 sound traversal times. In Figure 8.6, the upper branch of the hysteresis loop represents the simple, periodically

convecting systems and is generated by starting with steady convection and increasing the Rayleigh number quasi-statically. This simple periodic mode is about 10% more efficient at transporting heat. This efficient flow continues to a critical Rayleigh number where the Nusselt number drops and the flow becomes turbulent. The turbulent flow morphology continues as the Rayleigh number is increased. It should be noted here that in this turbulent regime, the dimensionless entropy production (identically equal to the Nusselt number) is related to the Rayleigh number by the 2/7ths power law, $\dot{S}_{ND} \sim Ra^{\frac{2}{7}}$. As the Rayleigh number is decreased from this point, the lower branch of this plot is generated as the flow remains somewhat turbulent. It is possible to generate either of these two different flow morphologies by starting with different initial conditions within this hysteresis range. Towards the center of the hysteresis range, $Ra \approx 2 \times 10^5$, the different flows are insensitive to rather large perturbations (on the order of 10%). Near the transitional Rayleigh number, the efficient flow is sensitive to perturbation. This hysteresis persists as the grid is refined and is therefore expected to exist in the continuum limit.

Our discrete approximation to the system uses a finite number of variables to approximate its state. Each state variable can be thought of as a degree of freedom in a multidimensional dynamical system whose set of Lyapunov exponents, $\{\lambda_i\}$, has a cardinality equal to the number of state variables. Examples of Lyapunov spectra for various systems can be seen in reference [79]. The spectrum describes the rates at which the phase space volume grows or decays in time. The maximum Lyapunov exponent, λ_1, indicates the chaotic nature of the system by indicating whether neighboring points in

phase space converge (for a stable system), diverge (for a chaotic system), or remain close neighbors (for a neutrally stable system) in general,

$$\boxed{\lambda_1 \equiv \lim_{\substack{t \to \infty \\ |\Delta(0)| \to 0}} \frac{1}{t} \ln \frac{|\Delta(t)|}{|\Delta(0)|}.} \tag{8.3}$$

The maximum Lyapunov exponent is also relatively easy to compute from a well-established

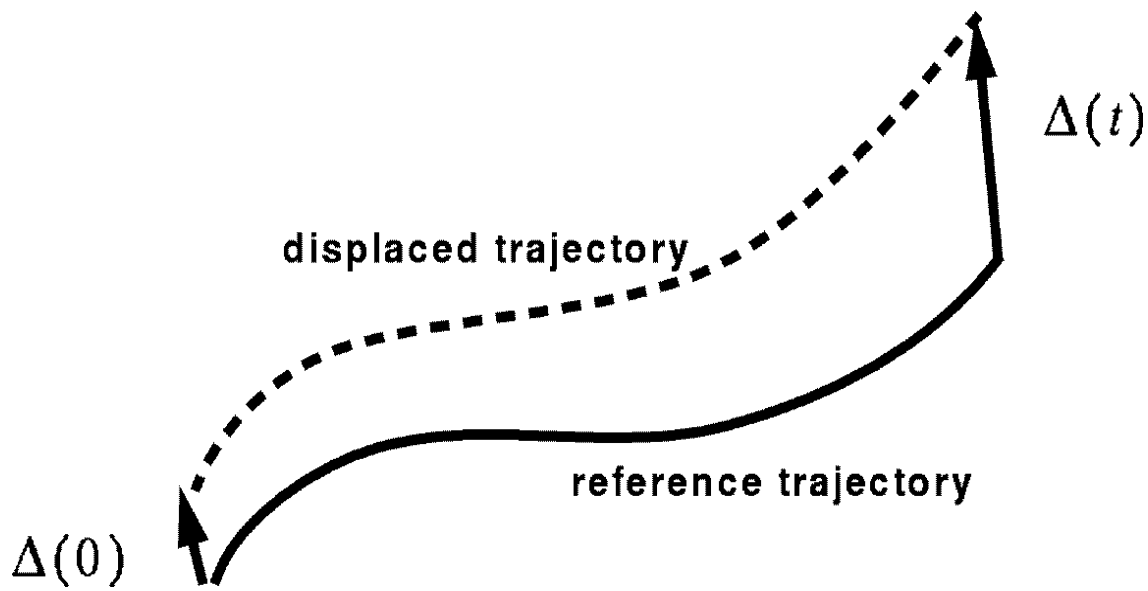


Figure 8.7: Maximum Lyapunov exponent is a measure of the rate that a perturbed system diverges from a reference trajectory.

fluid state. This is done by selecting a state (at random) that represents a small perturbation from the original state and allowing both to evolve for a short time before uniformly renormalizing all of the state variables so that the phase space distance is equal to the

initial perturbation [80]. By doing this several thousand times, the perturbation direction rotates to give the eigenvector corresponding to the maximum Lyapunov exponent. Once this alignment is established, the time-averaged Lagrange multiplier needed to re-adjust the phase space distance is exactly equal to λ_1. Figure 8.8 shows $\lambda_1\tau_s$ for various Rayleigh numbers, where τ_s is the sound traversal time. Five hundred sound traversal times are allowed to pass for the perturbation to establish itself and another five hundred sound traversal times are used for the time-averaging. The hysteresis loop, in this plot, goes counter-clockwise. As the Rayleigh number is increased, from steady state, the lower branch is generated until the transition to turbulence which is indicated by a change in the flow morphology and a sudden jump in the dimensionless maximum Lyapunov exponent. As the Rayleigh number is decreased, $\lambda_1\tau_s$ smoothly decreases along the upper branch of the hysteresis loop. The chaotic flow loses phase space information at a greater rate. The Kolmogorov entropy, the sum of the positive Lyapunov exponents, $\sum\lambda^+$, quantifies this rate. For our simulations, however, it is not easy to get the entire Lyapunov spectrum and it is also not clear how well the spectrum for our discrete approximation describes the continuous system. Instead, we calculate the maximum Lyapunov exponent and use it as an estimate of the rate of loss of phase space information. To make this rate dimensionless, the maximum Lyapunov exponent, λ_1, is multiplied by the sound traversal time, τ_s, the time for information to traverse the height of the system. Figure 8.8 shows the dimensionless rate at which phase space information is lost, $\lambda_1\tau_s$, for various systems. In this case, the hysteretic loop is also evident, but with the upper branch now representing

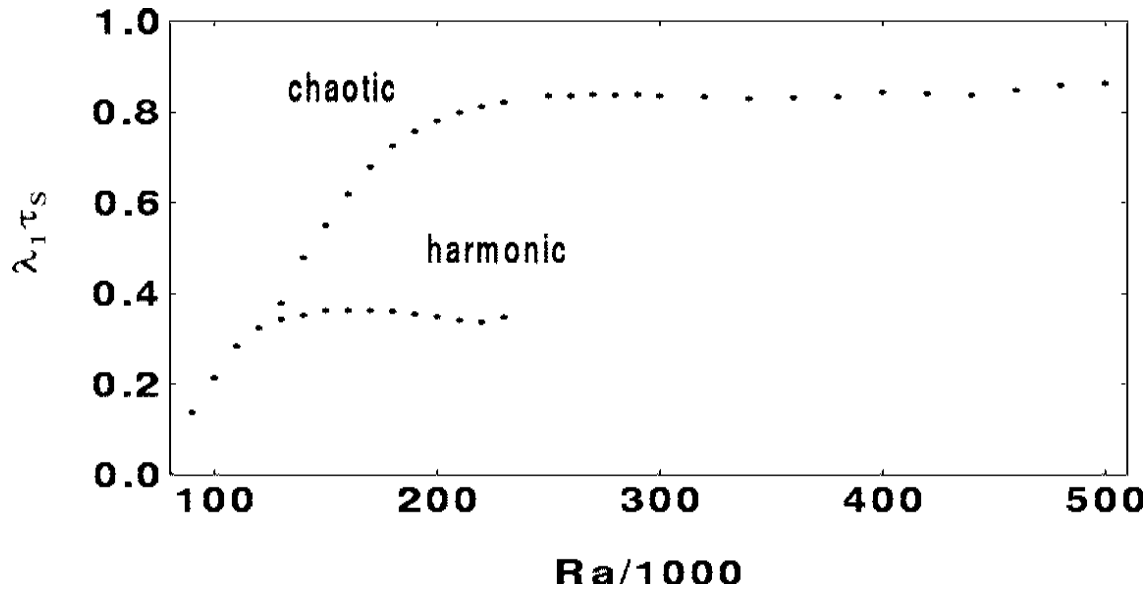


Figure 8.8: Dimensionless maximum Lyapunov exponent for chaotic and harmonic flow. Chaotic flows lose phase space information at a greater rate.

the rate for chaotic and turbulent flows.

From Figures 8.6 and 8.8, one can see that the difference in the dimensionless entropy production rates corresponds to the difference in the dimensionless rate of loss of phase space information for flows at the same Rayleigh number. This implies a connection between the thermodynamic entropy production and the information entropy rate near the onset of convective turbulence. A linear sum of the two rates (see Figure 8.9) connects the two morphologies over the entire range for which the they coexist. A linear coefficient is determined so that the hysteresis in the linear sum is greatly reduced. The linear sum appears to increases continuously with Ra.

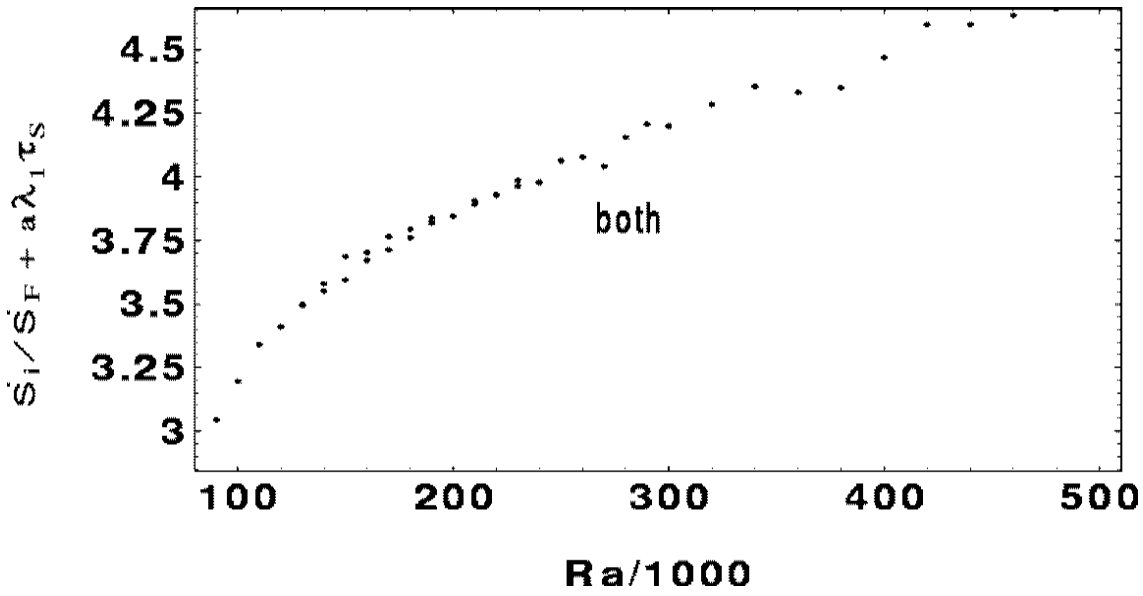


Figure 8.9: Linear sum of the dimensionless internal entropy production and dimensionless maximum Lyapunov exponent for the harmonic and chaotic flow morphologies.

Chapter 9

Convective turbulence

In this chapter, results of simulations of high-Rayleigh-number flows are discussed. Whether these flows are turbulent is addressed here. The term "turbulent" is not well-defined and is the subject of much investigation. In general, "turbulence" refers to flows that have both structure and randomness. This structure is often defined as a statistical property of the flow in that it exists as a spatial or time average. A good example of this is the Kolmogorov's scaling arguments defining the energy spectrum of the turbulent eddies. This is the well-known five-third's law.

Here, both qualitative and quantitative descriptions are used to argue that turbulent convection is achieved for the two-dimensional compressible fluid under study.

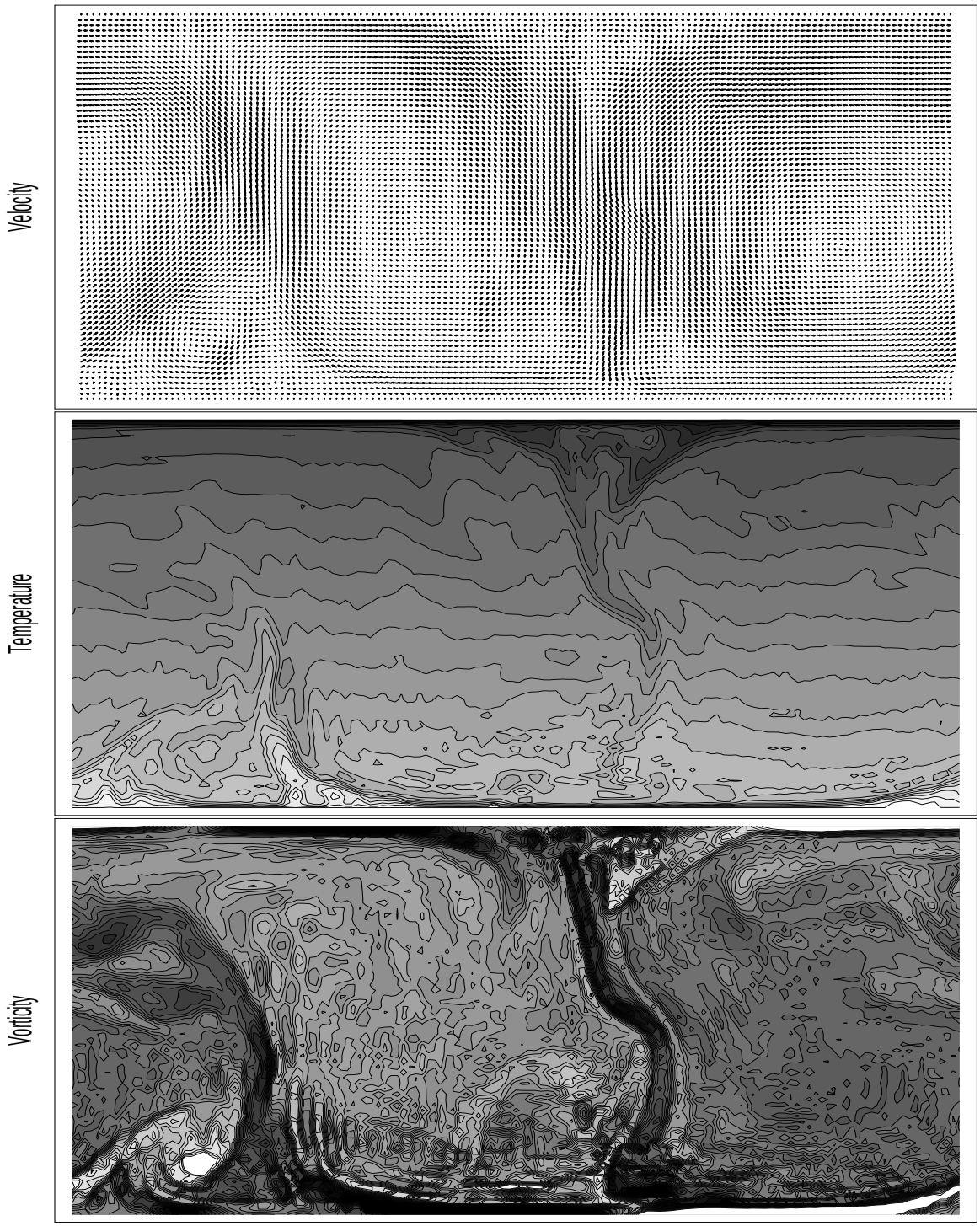


Figure 9.1: Velocity, temperature, and vorticity fields for two-dimensional turbulent convection at $Ra = 1.5 \times 10^8$.

9.1 Non-Boussinesq (asymmetric) flow

The Oberbeck-Boussinesq approximation, described in Section 1.2, is still used for much of the analytic work and some simulation of convective processes, even for very high Rayleigh numbers. The main problem with this approach is that the artificial buoyancy term, used to correct for the incompressibility simplification, has only a vertical component. This leads to flows with a symmetric character. For example, the hot, upward-flowing and cold, downward flowing plumes would have symmetry in the velocity and temperature fields [40]. In some situations, such as unit Prandtl number, turbulence was not observed for simulations using the Oberbeck-Boussinesq approximation for $0 < Ra \leq 1.6 \times 10^8$ [27].

Figure 9.1 shows the velocity, temperature, and vorticity fields for convection at $Ra = 1.5 \times 10^8$. The asymmetry in the plumes is clear – the warm, upward-flowing plume (on the left) is broad while the cold, downward flowing plume is narrow and fast. This has been observed in experiments (see, for example, figure 2.7) and in natural phenomena [48]. Vorticity is seen on many scales – from the length scale of the system to that of a few cell widths. In these simulations, the smallest vorticity length scale observed is about $1/25^{th}$ of the system size. A mechanism for the generation of vorticity on these smaller scales is the convection instability itself. That is, on a length-scale smaller than the length of the system, the temperature difference may be great enough to induce a micro-convecting system. The shear from the fast, downward-moving cold plume and the slower, upward-moving warm plume also induces vorticity in the fluid near the plumes. The existence of thermal and viscous boundary layers is evident in the temperature and velocity plots.

9.2 Scaling

A scaling relation between the dimensionless heat flux (equal to the dimensionless entropy production) and the Rayleigh number has been the subject of much investigation. Experimental studies of convecting helium gas [33] demonstrates the existence of a scaling region where the convective part of the heat flux is related to the Rayleigh number by $Nu - 1 \sim Ra^{0.282 \pm 0.006}$. This is different from the "classical" result [32] relating the heat flux to $Ra^{\frac{1}{3}}$ which is based on the assumption of marginal stability at a thermal boundary layer. Numerical simulations of incompressible Boussinesq fluids [25, 27] also reveal a scaling relation close to the experimental one for Rayleigh numbers between 10^8 and 10^{15}.

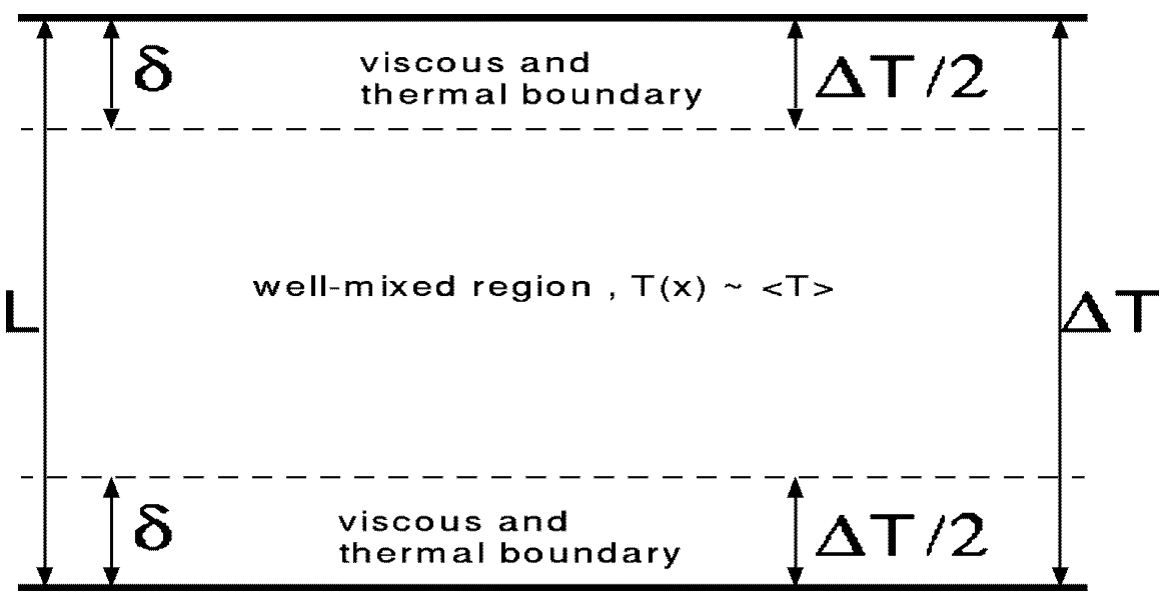


Figure 9.2: Diagram for the classic 1/3 scaling law theory.

The "classical" result relating the dimensionless heat flux to $Ra^{\frac{1}{3}}$ is based on the

assumption of a thin thermal boundary layer across which the time-averaged temperature changes by $\Delta T/2$, where ΔT is the change in temperature across the system. The thickness of the boundary layers, δ, is such as to just deny the instability leading to local convection. The "local" critical Rayleigh number, $Ra_c = \rho g \delta^3/\eta\kappa(\Delta T/2)$ and the Nusselt number, Nu, can be computed as the Fourier conduction across the boundary layer, $q_{bl} = -\kappa(\Delta T/2)/\delta$ divided by the conduction across the system, $q_F = -\kappa(\Delta T)/L$, where L is the height of the system. By solving for δ in terms of Nu, one obtains $Nu \sim (Ra/Ra_c)^{\frac{1}{3}}$. This power-law relation has been developed independently in at least two different ways $[31, 81]$.

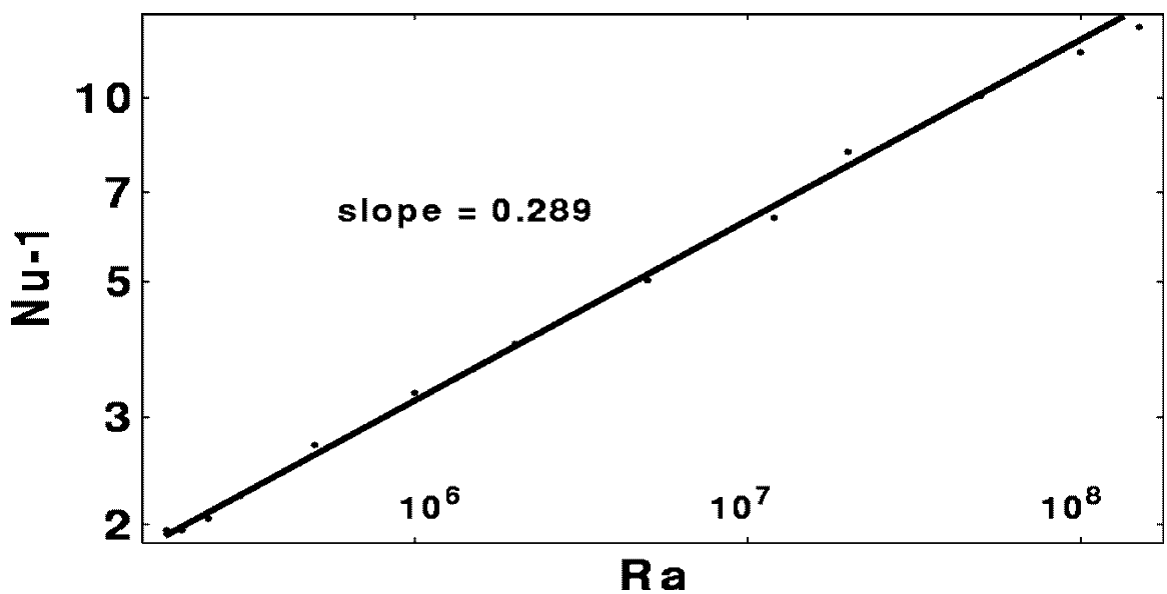


Figure 9.3: Dimensionless heat flux from turbulent convection as a function of the Rayleigh number. This suggests the existence of hard turbulence in two-dimensional compressible fluids.

There are a number of theories that account for the deviation from the classic $\frac{1}{3}$ value

in the hard turbulence region. Castaing *et alii* [33] propose that a large scale fluctuating wind may stabilize the thermal boundary layer to yield the $\frac{2}{7}$ power-law relation. She [34] analyzed the Boussinesq equations with the assumption that the central fluctuating temperature field interacts strongly with the turbulent velocity field. Shraiman and Siggia [35] analyze the Boussinesq equations with the assumption that thermal boundary layer is nested within the viscous boundary layer. Ching [38] uses the assumption that a large nonuniform shear influences the thickness of the thermal boundary layer.

The existence of this $\frac{2}{7}$ power-law relation between the Rayleigh number and the dimensionless heat flux, along with an exponential temperature distribution, are the main indicators of "hard" turbulence. This has been reported for two-dimensional convection of a Boussinesq fluid [25, 27, 26] and is evident in our simulations for the chaotic flows [29].

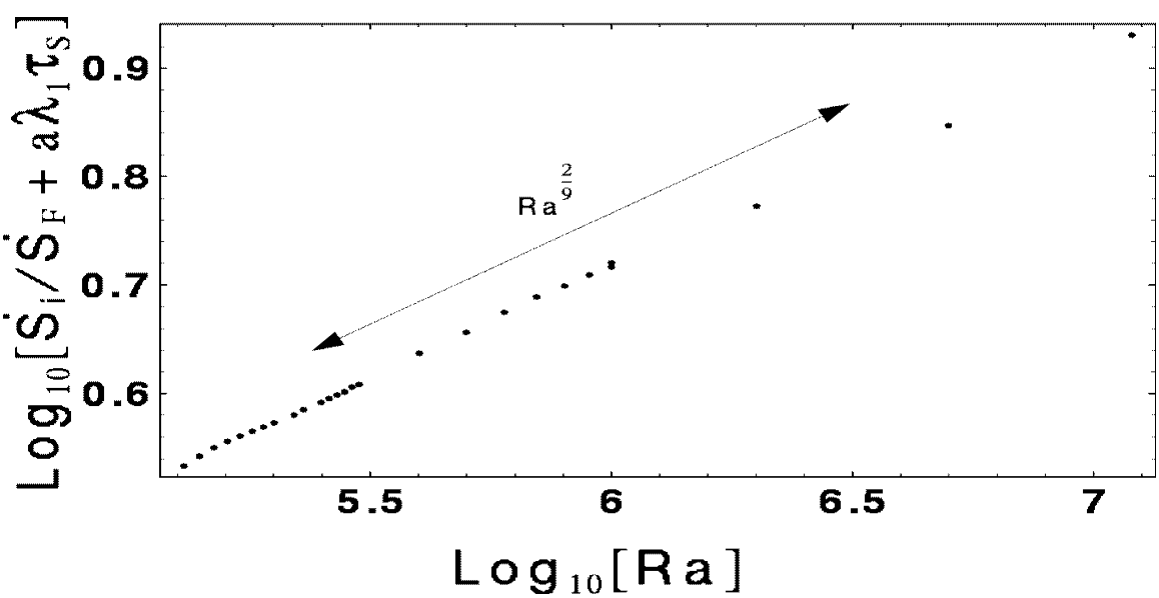


Figure 9.4: Linear combination of the dimensionless entropy rates for turbulent convection as a function of the Rayleigh number.

For higher Rayleigh numbers, within the turbulent region ($2.5 \times 10^5 < Ra$), a scaling relation between a linear combination of the dimensionless internal entropy production and the dimensionless maximum Lyapunov exponent (as discussed in the last chapter) and the Rayleigh number $\dot{S}_{ND} + a\lambda_1\tau_s \sim Ra^{2/9}$ is suggested by the data. This relation is somewhat inconclusive because the data for our simulations only range a little over one decade. Figure 9.4 shows a log-log plot of this linear combination as a function of the Rayleigh number.

Part IV

Conclusions

Chapter 10

Conclusions and prospects

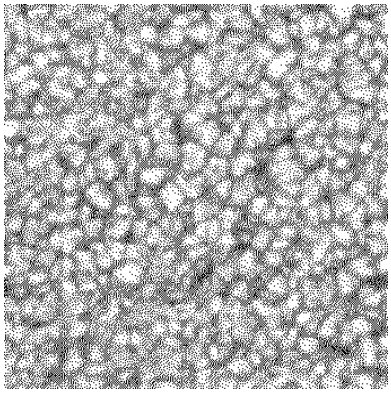


Figure 2.5*: Convection cells on the Sun

Thermal convection is a subject that is rich in complex behavior and provides interesting examples of nonequilibrium dynamical systems, both steady and time dependent. It is a phenomenon that depends on the properties of the fluid and the system environment. Because any fluid that expands upon heating and is in the presence of a steady gravitational field may spontaneously convect heat, this phenomena is universal and is observed in vastly different systems. Convective processes may be created in a simple laboratory with ordinary fluids and nominal effort; they also may be created on a kitchen stove.

It is a mechanism by which heat is transported in many systems and consequently gives rise to motions that contribute to the dynamics of other systems. Wind from Earth's atmospheric convection, for example, contributes to the erosion of mountains and the propagation of seeds.

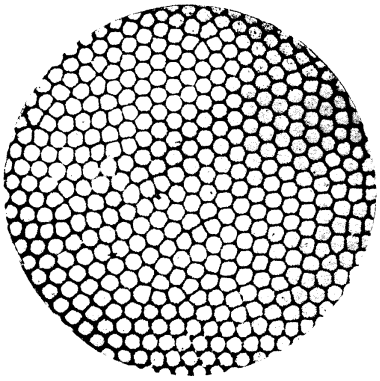


Figure 2.6*: Convection cells in Bénard's apparatus

In this dissertation, the compressible Navier-Stokes equations are solved by computer simulation to explore the dynamics of thermal convection. Unlike the Oberbeck-Boussinesq approximation, these equations are nonlinear and are difficult to explore without computer simulation. They, however, provide solutions that are more closely based on first principles. The numerical method described in this dissertation represents a novel approach for solving the compressible Navier-Stokes equations. This numerical method is used to investigate several aspects related to thermal convection of compressible fluids.

10.1 Numerical method

This method described in Part II is shown to be accurate, conservative, fast, simple, and valid.

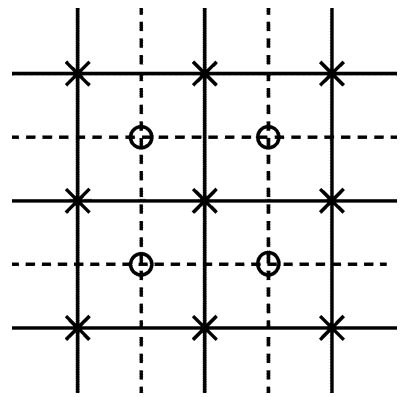


Figure 4.4*: Diagram of the "dual-staggered" mesh

As discussed in Chapter 4, the flux form of the Navier-Stokes equations is discretized on a "dual-staggered" mesh. The "dual"-part refers to the fact that the mass density is updated on one grid (zone-centered) while the energy and momentum are updated on the other. The "staggered"-part refers to the fact that the flux of the conserved quantities are updated at the midpoints in each direction to avoid a decoupling of the even and odd-numbered nodes (the even/odd instability).

It is shown in Chapter 5 that the solution of the differential equations requires accurate midpoint interpolation and spatial differentiation. This is accomplished by representing the variables by cubic splines. The midpoint interpolant of the spline is

$$\boxed{\hat{f}_0 = \frac{f_{j+1} + f_j}{2} - \frac{h^2}{16}(M_{j+1} + M_j) + O(h^4).} \tag{5.29*}$$

The midpoint spatial derivative of the spline is

$$\boxed{f_0' = \frac{f_{j+1} - f_j}{h} - \frac{h}{24}(M_{j+1} - M_j) + O(h^4).} \tag{5.40*}$$

where $f_0 \equiv f(x_j + \frac{h}{2})$. Here, M_j is the second derivative of the spline function at x_j. In Section 5.4, this method is shown to have a local truncation error of the order h^4 for the midpoint interpolation and the midpoint derivative. The proof of the accuracy of the midpoint interpolation involves showing that the error in the approximation of the second derivative of the function at the node point is of order h^2,

$$\boxed{M_j - f''_j = O(h^2).} \tag{5.33*}$$

The proof of the accuracy for the midpoint derivative involves showing that the error in the approximation of the difference of the second derivative for two neighboring points is of order h^3,

$$\boxed{(M_{j+1} - M_j) - (f''_{j+1} - f''_j) = O(h^3).} \tag{5.39*}$$

Conservation is demonstrated in Section 5.5 by considering the flux form for the time-rate-of-change of a quantity Q,

$$\int_L \frac{\partial}{\partial t} Q dx = \int_L [-\frac{\partial}{\partial x} F + S]\, dx \tag{5.41*}$$

where F is the flux and S is the source. This equation is then discretized in terms of a

grid-centered flux divergence,

$$\frac{\partial}{\partial t} h \sum_{j=0}^{n-1} Q_{j+\frac{1}{2}} = -h \sum_{j=0}^{n-1} F'_{j+\frac{1}{2}} + h \sum_{j=0}^{n-1} S_{j+\frac{1}{2}}. \tag{5.42*}$$

Equation 5.40* is substituted for the midpoint derivative of the flux. Expanding the telescoping sums and eliminating canceling terms leaves

$$\frac{\partial}{\partial t} Q_{tot} = (F_0 - F_n) - \frac{h^2}{24}(M_0 - M_n) + S_{tot}. \tag{5.45*}$$

By setting $M_0 = M_n$, we recover

$$\boxed{\frac{\partial}{\partial t} Q_{tot} = (F_0 - F_n) + S_{tot},} \tag{5.46*}$$

where the time-rate-of-change of the total conserved quantity is equal to the flux in from the boundaries minus the flux out to the boundaries plus the sources. This is satisfied for periodic boundaries because $F_0 = F_n$ and $M_0 = M_n$. For fixed boundaries, the choice of $M_0 = M_n$ is satisfied by the "natural" end conditions, $M_0 = M_n = 0$. This numerical methods guarantees conservation of mass, momentum, and energy.

It was also shown in Section 5.6 that this method requires relatively low computational effort. The computation of the cubic spline requires the solution of a tri-diagonal matrix (in the case of fixed boundaries) or nearly tri-diagonal (in the case of periodic boundaries) matrix with non-zero corners. In both cases, the linear system can be solved in $O(N)$ steps, where N is the number of nodes. The computational effort scales linearly with the number of nodes. By extrapolating the results for various resolutions, as discussed in Section 6.2, the continuum (zero-mesh-size) limit can be determined. This can be used to get accurate results with less computational effort.

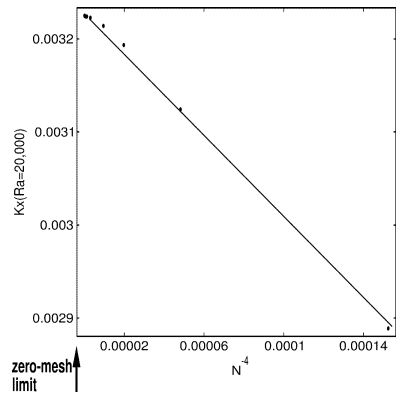


Figure 6.2*: Results extrapolated to the continuum limit

After the second derivative of the spline function is known, calculation of the midpoint interpolant or midpoint derivative is as simple as finite difference methods.

In Chapter 6, the validity of this method was demonstrated by comparison to results from experiments and other computer simulations.

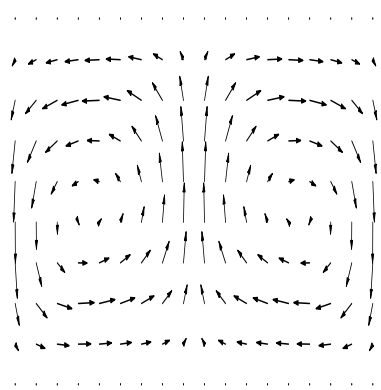


Figure 7.1*: Velocity field for steady two-dimensional convection

Steady (time-independent) convection is the simplest to simulate numerically. Both the cubic spline method and second-order centered finite-difference methods are used successfully to simulate the steady flow. Steady results are obtained, with double-precision accuracy, within a thousand

sound-traversal times. The results from both methods, extrapolated to the continuum limit, agree within 1%.

The first type of time-dependent convection, referred to as harmonic, oscillatory, or periodic, has a simple sinusoidal time profile for observables, such as the heat flux. Here, periodic convection commences near $Ra = 80,000$ with a cycle time corresponding to 0.2 of the thermal diffusion time. This is consistent with *ab initio* molecular dynamics simulations in a slightly different two-dimensional geometry. Consistency with molecular dynamics simulations is desirable because they are inherently compressible.

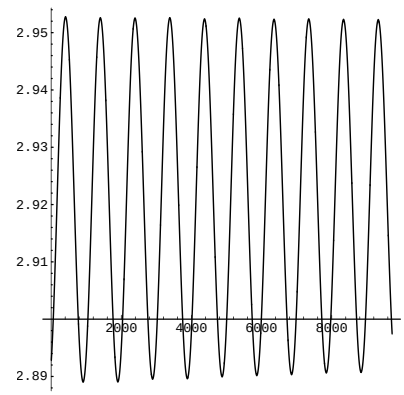


Figure 8.1*: Time profile of the heat flux for harmonic convection

As the system is driven further from equilibrium the system makes a transition to chaotic convection. Simulations of a two-dimensional ideal gas reveal that there is a drop in the Nusselt number associated with this transition. This is the result of well-established simulations where transient effects are small. The transition and the corresponding drop in the dimensionless heat flux occurs between a Rayleigh number of 200,000 and 300,000. This is consistent with experiments with liquid mercury (a nearly incompressible fluid with a low Prandtl number) and gaseous helium conducted by different researchers. The temperature histogram at a

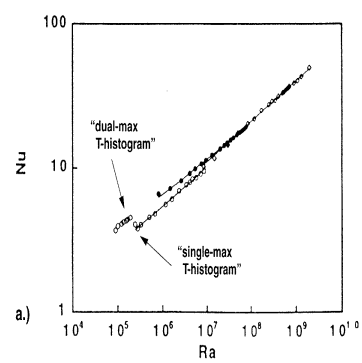


Figure 8.2.a*: Drop in heat flux in experiment with mercury

point in the center of the simulated cell makes a transition from a dual-maximum to a single-maximum as the system goes from periodic to chaotic. This was also reported in the mercury experiment.

Quantitative agreement can be seen in a plot of the dimensionless heat flux from convection $Nu - 1$ verses the Rayleigh number of the system. The well-known scaling relation, $Nu - 1 \sim Ra^{2/7}$, is recovered. This scaling relation is seen in numerous experiments and simulations of turbulent convection. This agreement not only validates the numerical method discussed in Chapters 4-6, but also suggests that hard turbulence is possible for two-dimensional compressible fluids.

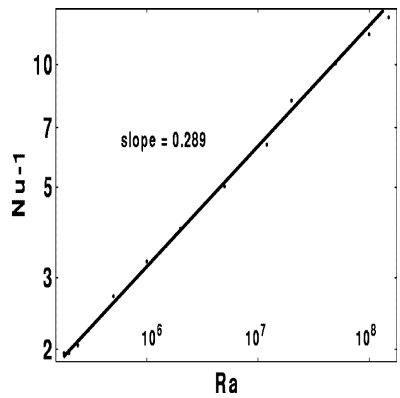


Figure 9.2*: Dimensionless heat flux from turbulent convection

10.2 Two-dimensional compressible convection

Once validated, this numerical approach was used to investigate many aspects of thermal convection. This includes demonstrating the existence of coexisting steady compressible flows, revealing a drop in the dimensionless heat flux at the transition to chaotic flow, revealing a dual morphology region in which harmonic and chaotic flows may coexist, the characterization of the entropy production and Lyapunov instability of these flow, and a characterization of turbulent compressible convection.

10.2.1 Coexisting attractors in compressible convection

The fully-converged solutions of the complete nonlinear viscous compressible conducting continuum equations establish the coexistence of several independent flow solutions, all for the same boundary conditions and the same constitutive model. Because they were discovered from randomly-selected initial conditions, the basins of attraction of these continuum solutions must have comparable measures. No doubt the boundaries separating such basins are sufficiently complicated to frustrate any useful analysis. Unlike equilibrium thermodynamic states these nonequilibrium flows cannot be precisely compared with regard to a sensible definition of "stability" because there is no reversible path connecting them.

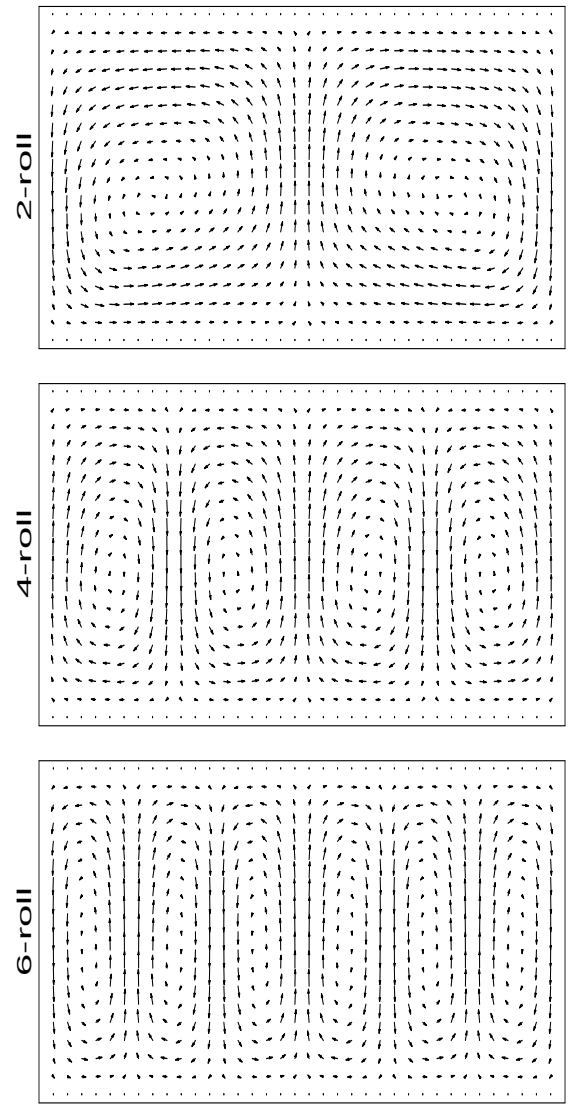


Figure 7.2*: Coexisting solutions (2-, 4-, and 6-roll) for compressible convection

In Chapter 7, the various aspects of the flows, which have some intuitive connection with stability, were compared. Such properties include (i) the kinetic and internal en-

ergies associated with the flows; (ii) the heat fluxes, equivalent to a knowledge of the internal entropy production; (iii) the growth rates of stationary modes, as measured from infinitesimal seeds. The first two of these properties suggest the relative stability of the six-roll pattern, while the growth rate favors the four-roll pattern. The details available from computer simulations may eventually lead to correlations among these, and other, measures of relative stability.

There is a logical small-scale-to-large-scale hierarchy of simulation techniques, beginning with microscopic molecular dynamics, continuing through smooth particle applied mechanics, with its fluctuations, and concluding with fluctuation-free continuum mechanics. The instabilities which characterize macroscopic turbulence and other flows can be followed through this hierarchy. The degeneracy found here suggests strongly that the strange attractors which are pervasive in nonequilibrium time-reversible atomistic simulations [59] eventually partition large-system phase spaces into disjoint parts. Simultaneously the time-reversibility and the ergodicity which characterize the smallest nonequilibrium systems are lost as the continuum limit is approached.

10.2.2 Heat flux at the onset of chaotic convection

Although our simulations are not an explicit attempt to model the experiments with convecting mercury or helium, the character of the results is very similar. Since the simulations use an ideal gas equation of state, model a compressible fluid, and have a unit Prandtl number, it is a fair model for the helium gas experiment. On the other hand, the simulations are two dimensional. The flows in a 2D system are different from those of a 3D system. The simulation are not a very good model for the mercury experiments - 2D rather than 3D, compressible rather incompressible, and intermediate rather than low Pr. Despite the difference, the behavior in all three situations have a common characteristic - a drop in the heat flux as the system makes the transition to chaotic flow.

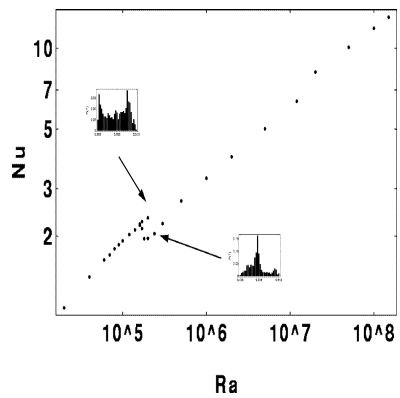


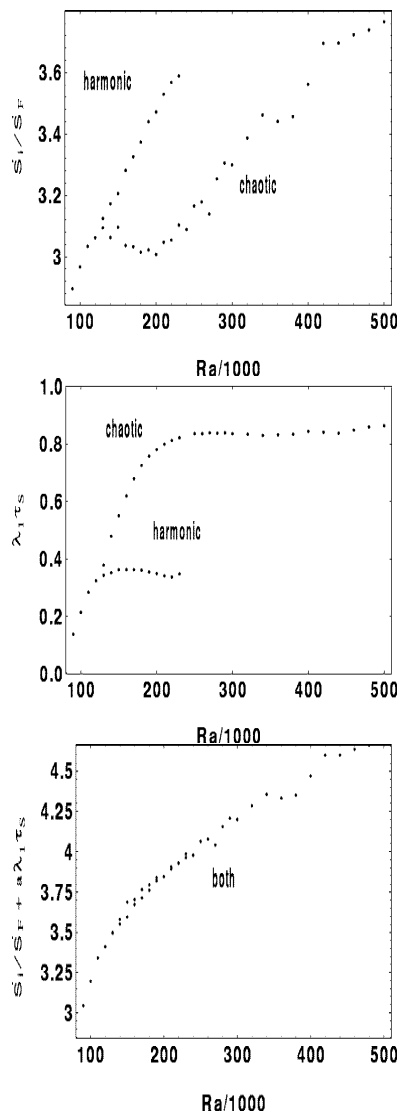
Figure 8.2*: Simulation results showing the drop in Nusselt number at the transition from harmonic to chaotic convection

10.2.3 Coexistence of "harmonic" and chaotic flows

It was shown in Chapter 8 that for time-dependent convective flows, with Rayleigh numbers in a region near the transition to chaotic convection, two distinctly different, but stable, flow morphologies coexists. A relationship is revealed between two important quantities that characterize these non-equilibrium, time-dependent flows – the entropy production rate and the rate at which phase-space information is lost.

These relations are based on well-established flow simulations so that the transient effects are minimized. The flows are also, in general, stable and insensitive to random perturbations.

The existence of this dual morphology region and the hysteresis loop in the dimensionless heat flux that connects the two flow morphologies is also corroborated by the existence of sharp discontinuities in Nusselt number data reported for two independent experiments with two very different fluids – gaseous helium and liquid mercury. These experiments not only support these claims, but suggest a universality for this phenomena.



Figures 8.3.1*, 8.3.1*, and 8.3.1*: Dimensionless entropy production, Lyapunov exponent, and linear combination over the dual morphology region

10.2.4 Entropy production and Lyapunov instability at the onset of turbulent convection

Because the Nusselt number is identically equal to the dimensionless entropy production of the system,

the results suggest that this dynamical system, driven farther from equilibrium, has a sudden drop in the entropy production as it makes a transition to chaos. The maximum Lyapunov exponent for the simulation is a measure of the rate at which phase-space information is lost. The drop in entropy production for the system at the transition is accompanied by a corresponding increase in this rate of information loss.

10.2.5 Turbulence in two-dimensional compressible convection

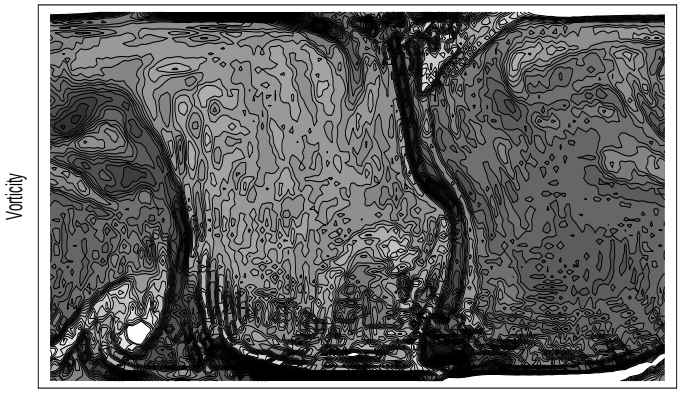


Figure 9.1.c*: Vorticity fields for turbulent convection

The presence of hard turbulence is suggested by the scaling relation between the Nusselt number and the Rayleigh number. Although this has been shown for two-dimensional simulations based on the Oberbeck-Boussinesq approximation, this is the first for two-dimensional compressible flow. The scaling relation between the Rayleigh number and the linear sum of the entropy production rate and the K-entropy [rate] may shed light on the continuing investigation on the relation between the dimensionless heat flux and the Rayleigh number. Recently, a power-law relation between the Lyapunov exponent times a characteristic time of the system (like a sound traversal time) and the Reynolds number has been announced [82], $\langle\lambda\rangle\,\tau_0 \sim Re^{0.459}$. This, along with the relation between the Reynolds number and the

Rayleigh number, further supports the work detailed in this dissertation.

Part V

Appendix

Appendix A

Convecting double pendulum

A.1 Introduction

A double pendulum, with thermally-conducting and thermally-expanding masses immersed in a constant-gradient temperature field, assists in transporting heat in a way that is analogous to buoyancy-driven convection [83]. A Rayleigh-number-like order parameter is defined for this system and characterizes how far from equilibrium the system is. The additional entropy produced by the convecting double pendulum is calculated as a function of the order parameter as the system is driven from regular to chaotic motion.

The intent in studying this model is to characterize the entropy production rate as this simple, non-equilibrium system is driven to chaotic trajectories. The results are compared to those of the more complex simulations of a compressible fluid, convecting heat in two dimensions. The simulations of the convecting fluid suggest that within a range of

Rayleigh numbers the time-averaged entropy production is related to the K-entropy (the rate at which phase-space information is lost).

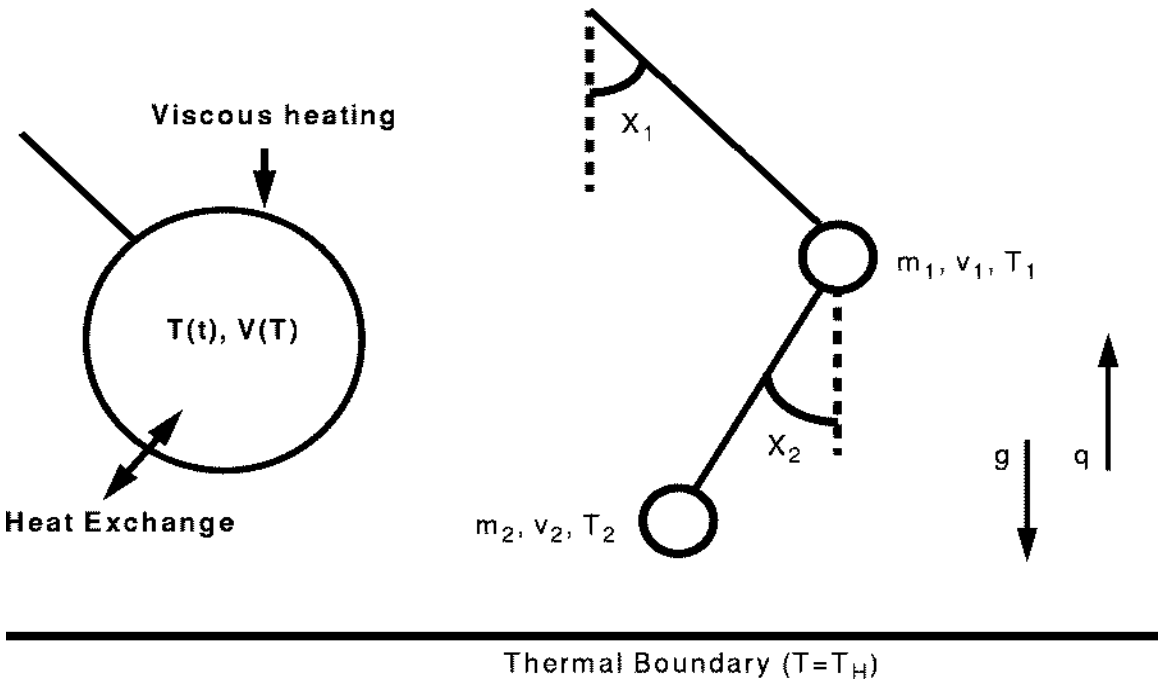


Figure A.1: Diagram of convecting double pendulum. The lower half of the cell is shown. A diagram of the thermally expanding mass is shown on the left.

The convecting double pendulum is suspended from a point in the center of the thermal cell of height L. Each of the massless rods has a length of $L/4$ so that the outer mass may reach the top and bottom thermal boundaries. The constant temperatures for the lower and upper boundaries is T_H and T_L respectively with $T_H > T_L$. The background temperature between the boundaries has a linear temperature profile (constant gradient). The masses interact with the background by exchanging heat at a rate that is proportional to the difference between the mass temperature and the background temperature. Ad-

ditionally, the background fluid provides some viscous heating that is proportional to the mass velocity. The heat conduction and viscous heating proportionality constants define the constant thermal conduction and viscosity coefficients. The volume of the masses are allowed to change with temperature, defining the constant expansion coefficient, α.

As for a convecting fluid, gravity acts in a direction opposite to the flow of heat. Here, the magnitude of gravity is set so that the change in potential energy balances the change in thermal energy as the mass moves from the bottom to the top of the cell,

$$mgL = k_B \Delta T. \tag{A.1}$$

To allow for mechanical equilibrium, the viscous drag force for the characteristic velocity is balanced with the buoyancy force for the characteristic temperature difference,

$$\alpha m g \Delta T = \eta |v|. \tag{A.2}$$

The "Rayleigh number" for this system is defined as the ratio of the power per area from convection to conduction,

$$\text{“Ra”} \equiv \frac{cm|v|\Delta T/L^2}{\kappa \Delta T/L}. \tag{A.3}$$

By using A.1 and A.2 to define the gravity and characteristic velocity, the "Rayleigh

number" can be expressed as

$$\text{“Ra”} = mck_B \frac{\alpha}{\eta\kappa}\left(\frac{\Delta T}{L}\right)^2. \tag{A.4}$$

This is similar to the Rayleigh number defined for for a fluid (see 3.19), except that the density and specific heat are replaced with the mass and heat capacity *per unit mass*. For the simulations, all parameters are left constant except for the thermal expansion coefficient α. As α is increased, the system moves from equilibrium.

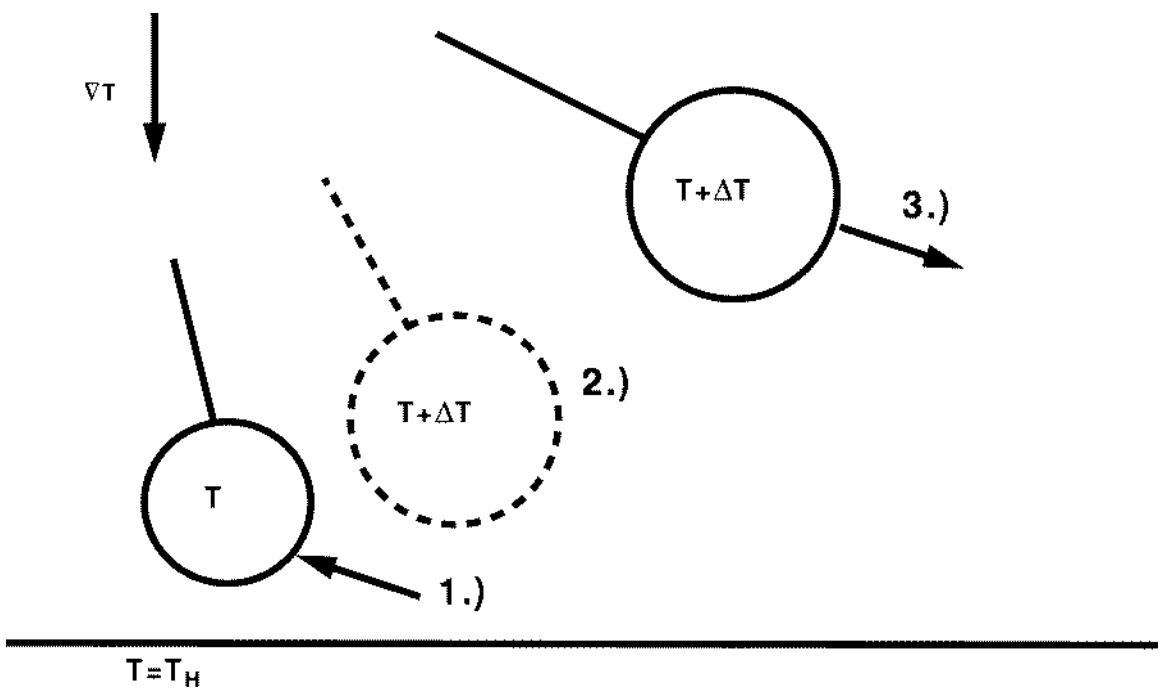


Figure A.2: Pendulum masses 1.) absorb heat at a higher background temperature 2.) expand, rise and 3.) release heat at a lower background temperature.

The thermal entropy of the system changes as each mass absorbs heat from a warmer surrounding fluid, expands, rises, and releases heat to a lower temperature fluid. The

entropy time-rate-of-change is defined as the amount of heat absorbed by each mass $cM\delta T$ divided by the fluid temperature per time step. The heat absorbed or released to the fluid is equal to the diffusion constant times the difference in the local temperatures.

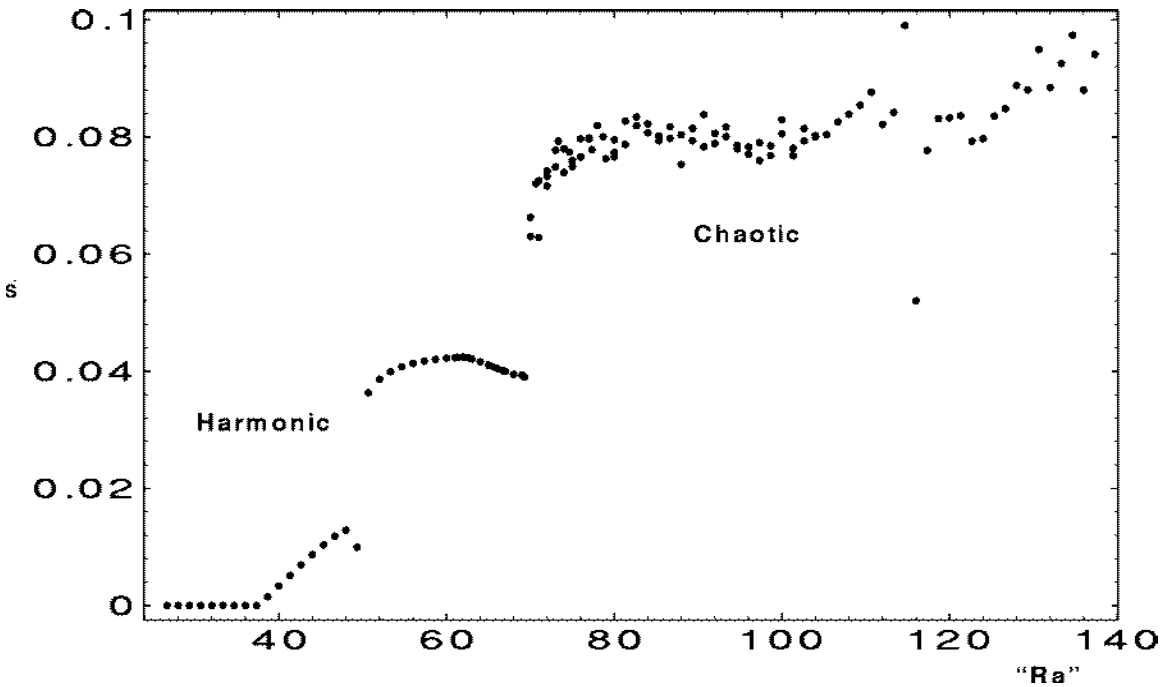


Figure A.3: Entropy production as the convecting double pendulum is driven away from equilibrium.

A.2 Results and discussion

As the "Rayleigh number" is increased, the system does indeed move from thermal and mechanical equilibrium. Change in thermal equilibrium is demonstrated in the change in the thermal entropy production – the time integral of the heat flux divided by the local fluid temperature. The change in the mechanical equilibrium is demonstrated by the

change in the pendulum trajectory – from simple-periodic to chaotic.

The time average of the entropy production rate is plotted for the pendulum for various "Rayleigh numbers" in Figure A.3. The figure shows a critical "Ra" near 37 above which the entropy production is non-zero. The entropy production continues to increase linearly to a point near "Ra" $= 48$ where a jump occurs. The entropy production increases at a lower rate with increasing "Ra" to a point near 60 after which it decreases with increasing "Ra". This is similar to the results discussed in Chapter 8 for the dimensionless internal entropy production for a convecting fluid. See Figure 8.6. At a point near "Ra" $= 72$ another jump occurs in the entropy production.

The stroboscopic trajectory of the convecting double pendulum is shown in Figure A.4 for systems with various "Rayleigh numbers". These figures represent trajectories for the different regions observed. First, for low "Ra", there is no motion and no heat transported (entropy produced) by the pendulum. Above the critical "Ra", simple motion occurs. The trajectory at "Ra" $= 40$, for example, shows the first mass (the one constrained at a constant distance from the center) moving along a short arc while the second mass moves in a small loop to the left of the first. The critical "Ra" is also a bifurcation point where the system may choose one of two mirror-symmetric trajectories. This is analogous to the bifurcation at the critical Rayleigh number for convecting fluids with periodic boundaries where a particular roll may be clock-wise or counter-clock-wise. (The direction of the other roll(s) is(are) then constrained). The jump in the internal entropy production corresponds to a transition where the second mass moves entirely above the

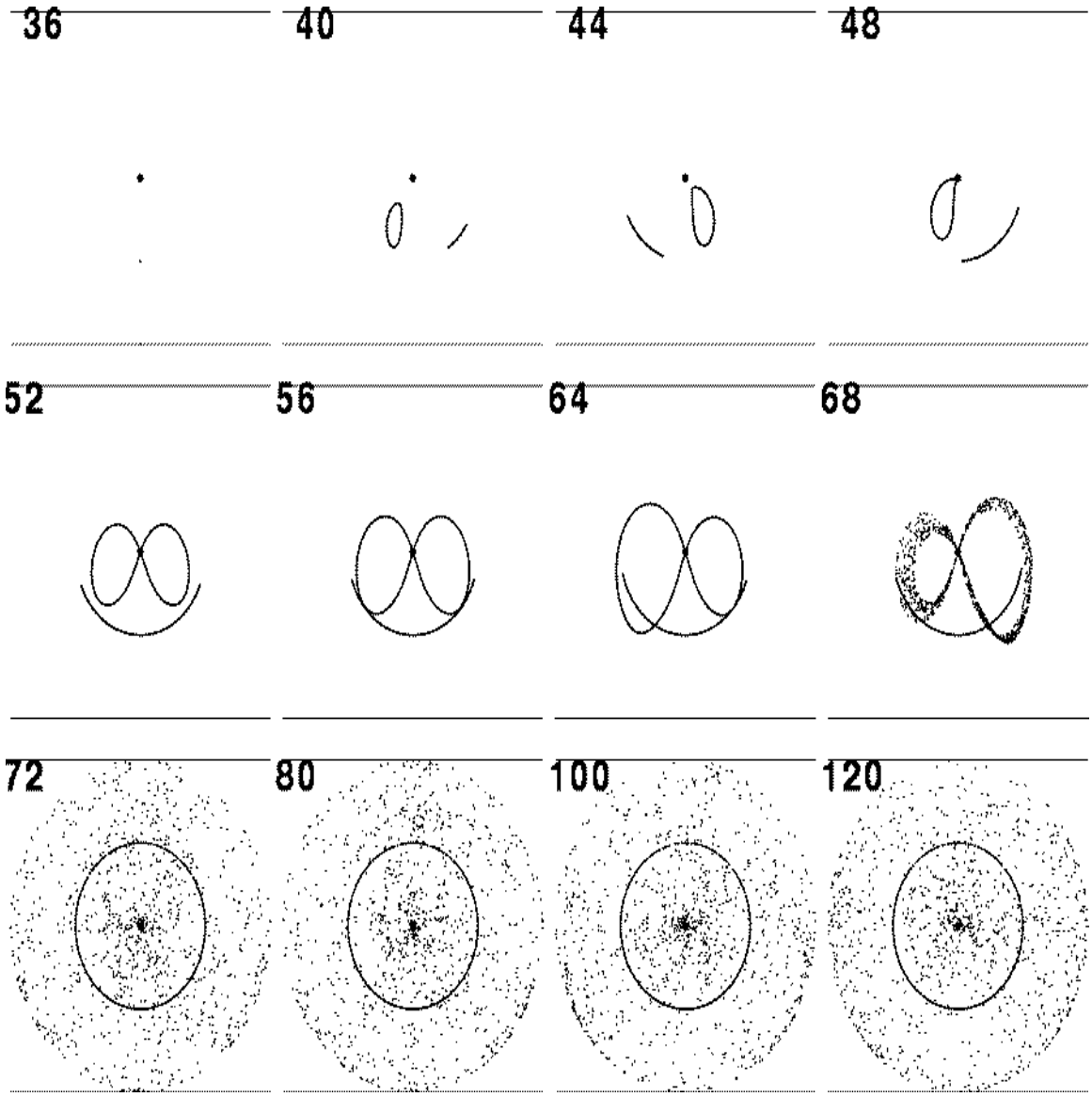


Figure A.4: Stroboscopic trajectory of the convecting double pendulum

first. The decrease in the entropy production corresponds to a region of "Ra" where the trajectory becomes erratic, possibly chaotic. Above the next jump in entropy production, the trajectory is chaotic. This is not quantified by a Lyapunov exponent in this preliminary study, but it is well-known that a double pendulum can have a chaotic trajectory at higher energies.

A.3 Future work

The convecting double pendulum is a simple (six-dimensional phase space) non-equilibrium system for which a Rayleigh-number-like order parameter and an entropy-production quantity can be defined. Phase transitions (in particular, a phase transition from harmonic to chaotic motion) are present and are analogous to a convecting fluid Further work includes the determination of the Lyapunov spectrum for comparison to the convection transition data.

A.4 Listing

```
Calculates the motion of a double pendulum.
C
      PROGRAM dp
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION x(6)
      COMMON/PARMS/ rL1, rL2, g
      COMMON/THERM/ alp, rkappa, eta, T1, T2
      CHARACTER*50 STATE_FILE    /'../dbpend/dpcu80.state'/
      CHARACTER*50 OUT_FILE      /'../dbpend/dpcu140.state'/
C                                f77 -O -r8 -o xdpu140 dpconv.f

      Ra = 140.0d0
      alp= 4.0d0
      rkappa= 0.15d0
      eta= 0.5d0
      eta= alp/Ra/rkappa

      Pi=4.0d0*DATAN(1.0d0)
      rL1=1.0d0
      rL2=1.0d0
      rL = 2.0d0*(rL1+rL2)
      Th=1.5d0
      TL=0.5d0
      deltaT=Th-Tl
      g=deltaT/rL

C     establish initial conditions
      write (*,'("reading file: ",a30 )') STATE_FILE

      open(22,file=STATE_FILE,form='formatted',status='OLD')
      DO i=0,1000
         read (22,'(i8,2x,2f8.4,2x,2f12.4,2x,2f8.4)',END=99)
     2        iter, x1,x2,x1dot, x2dot, T1, T2
      END DO
c      write (*,'(i8,2x,2f8.4,2x,2f12.4,2x,2f8.4)')
c     2     iter, x1,x2,x1dot, x2dot, T1, T2

 99   close(22,STATUS='KEEP')

      x(1)=x1
      x(2)=x2
      x(3)=x1dot
      x(4)=x2dot
      x(5)=T1
      x(6)=T2

      dt=0.001d0
      maxiter=10000000

C     START ITERATION
c      write (*,*) "Starting ... "
c      write (*,'(a, 3f12.6)')
c     2    "alp, et, kappa", alp, eta, rkappa
      Do iter=0,1000000
         CALL RK(x, dt)
      END DO
      sdottot=0.0d0
      ttot = 0.0d0
      DO iter=0,maxiter
         if (jmod(iter,maxiter/1000).eq.0) then
            open(14,file=OUT_FILE,form='formatted',ACCESS='APPEND'
     2           ,STATUS= 'UNKNOWN')
            write(14,'(i8,2x,2f8.4,2x,2f12.4,2x,2f8.4)') iter/10000,
     2           x(1),x(2),x(3),x(4),T1,T2
            close(14)
         end if
         CALL RK(x, dt)
         x1 = x(1)
         x2 = x(2)
         x1dot = x(3)
         x2dot = x(4)
         T1 = x(5)
         T2 = x(6)
         y1=2.0d0-Dcos(x1)
         y2=y1-Dcos(x2)
         T1b=1.5d0-y1/4.0d0
         T2b=1.5d0-y2/4.0d0
         sd1 = (-rkappa*(T1-T1b))/T1b
         sd2 = (-rkappa*(T2-T2b))/T2b
         sdot = (sd1+sd2)*dt
         sdottot = sdottot + sdot
         ttot = ttot + dt
      END DO
      STOP
      END

      SUBROUTINE RHS(xx,ff)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION xx(6), ff(6)
      COMMON/PARMS/ rL1, rL2, g
      COMMON/THERM/ alp, rkappa, eta, T1, T2
c     make things more readuhble
         x1 = xx(1)
         x2 = xx(2)
         x1dot = xx(3)
         x2dot = xx(4)
         T0=1.0d0
         T1 = xx(5)
         T2 = xx(6)
         vv1=x1dot**2
         vv2=x2dot**2 + x1*x2*dcos(x2-x1)*2.0d0
         v1=x1dot
         v2=x2dot
         s1 = Dsin(x1)
         s2 = Dsin(x2)
         s21 = Dsin(x2-x1)
         c21 = Dcos(x2-x1)
         y1=rL2+rL1*(1.0d0-Dcos(x1))
         y2=y1-rL2*Dcos(x2)
         T1b=1.5d0-y1/(2.0d0*(rL1+rL2))
         T2b=1.5d0-y2/(2.0d0*(rL1+rL2))

         ff(1) = x1dot
         ff(2) = x2dot
         ff(3) = (s21*(v2*x2dot+c21*v1*x1dot) +
     2       g*(c21*s2-2.0d0*s1)) /rL1/(2.0d0-c21*c21)
     3       + alp*g*(T1-T0)*rL1*s1 + alp*g*(T2-T0)*rL1*s1*Cos(x2)
     4       - eta*v1*rL1
         ff(4) = (s21*(2.0d0*v1*x1dot+c21*v2*x2dot) -
     2       2.0d0*g*(c21*s1-s2)) /rL2/(c21*c21-2.0d0)
     3       + alp*g*(T2-T0)*rL2*s2
     4       - eta*v2*rL2
         ff(5) = -rkappa*(T1-T1b) + eta*vv1
         ff(6) = -rkappa*(T2-T2b) + eta*v2*v2
      RETURN
      END

      SUBROUTINE RK(x, h)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      DIMENSION x(6), x1(6), f1(6), f2(6), f3(6), f4(6)
         hh = h/2.0d0
         hhh = h/6.0d0
         CALL RHS(x,f1)
         DO i=1,6
            x1(i) = x(i)+hh*f1(i)
         END DO
         CALL RHS(x1,f2)
         DO i=1,6
            x1(i) = x(i)+hh*f2(i)
         END DO
         CALL RHS(x1,f3)
         DO i=1,6
            x1(i) = x(i)+h*f3(i)
         END DO
         CALL RHS(x1,f4)
         DO i=1,6
            x(i) = x(i) + hhh*(f1(i)+2.0d0*f2(i)+2.0d0*f3(i)+f4(i))
         END DO
         Pi=4.0d0*DATAN(1.0d0)
         x(1)=Dmod(x(1), 2.0d0*Pi)
         x(2)=Dmod(x(2), 2.0d0*Pi)
      RETURN
      END
```

Appendix B

FORTRAN codes

B.1 The Navier-Stokes solver

The program used to generate the results discussed in this thesis is made up of the following header file and main file. The main file consists of an initialization part, cubic spline subroutines, and a Runge-Kutta integrator.

B.1.1 Header file

The header file defines the system size and allocated array memory. Common variables are defined. The variables are initialized from another file or a good guess.

```
CCCCCC-----------------------------------------------------------------X
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
C     s :Nx=18,Ny=9      sa:Nx=24,Ny=12     sb:Nx=30,Ny=15
C     m :Nx=36,Ny=18     ma:Nx=48,Ny=24     mb:Nx=60,Ny=30
C     l :Nx=72,Ny=36     la:Nx=96,Ny=48     lb:Nx=120,Ny=60
C     h :Nx=144,Ny=72    ha:Nx=192,Ny=96    hb:Nx=240,Ny=120
C
C     p1:Nx=34,Ny=17     p2:Nx=38,Ny=19     p3:Nx=46,Ny=23
C     p4:Nx=58,Ny=29     p5:Nx=62,Ny=31     p6:Nx=74,Ny=37

      PARAMETER (Nx=72,Ny=36)
      PARAMETER (Nn=Ny+1)
C
C     grid variables (G1)
      COMMON/G1ST/G1Rho(Nx,Nn),U(Nx,Nn),V(Nx,Nn),E(Nx,Nn)
      COMMON/G1CONS/Xm(Nx,Nn),Ym(Nx,Nn),Te(Nx,Nn)
      COMMON/G1CDOT/Xmdot(Nx,Nn),Ymdot(Nx,Nn),Tedot(Nx,Nn)
C
C     cell variables (G2)
      COMMON/G2ST/Rho(Nx,Ny)
      COMMON/G2CDOT/rhodot(Nx,Ny)
      COMMON/G2HOLD/G2ux(Nx,Ny),G2uy(Nx,Ny),G2vx(Nx,Ny),G2vy(Nx,Ny)
C
C     Grid3 variables
      COMMON/G3FLX/Gxm(Nx,Ny),Gym(Nx,Ny),Gte(Nx,Ny),Frho(Nx,Ny)
      COMMON/G3Gry/uy(Nx,Ny), vy(Nx,Ny), ey(Nx,Ny)
      COMMON/G3ST/G3Rho(Nx,Ny),G3u(Nx,Ny),G3v(Nx,Ny),G3e(Nx,Ny)
      COMMON/G3Grx/G3ux(Nx,Ny), G3vx(Nx,Ny)
C
C     Grid4 variables
      COMMON/G4FLX/Fxm(Nx,Nn),Fym(Nx,Nn),Fte(Nx,Nn),Grho(Nx,Nn)
```

```
      COMMON/G4Grx/ux(Nx,Nn), vx(Nx,Nn), ex(Nx,Nn)
      COMMON/G4ST/G4Rho(Nx,Nn),G4u(Nx,Nn),G4v(Nx,Nn),G4e(Nx,Nn)
      COMMON/G4Gry/G4uy(Nx,Nn), G4vy(Nx,Nn)
C
      COMMON/PARM/ delta,g,rLy,rLx,rkappa,eta,Th,Tl,bulk,Pi,rolls
      COMMON/RPARM/ iter
C
```

B.1.2 Setup and initialization

The input and output files are defined as are the run parameters such as the Rayleigh and Prandtl numbers.

```
Cubic Spline on staggered grid for
Continuum mechanics are used on a rectangular grid to
Calculate the Rayleigh-Be'nard flow patterns.
Conservative form of differential equations are used.
      PROGRAM rbcufd
      INCLUDE 'rbcu.h'
C     set up output data files
C     name output files
c
c      CHARACTER*50 IN_STATE_FILE    /'steady/cufd1M21.state'/
      CHARACTER*50 OUT_STATE_FILE   /'steady/cufd25C21.state'/
      CHARACTER*50 KE_FILE          /'steady/cufd25C21.ke'/
      CHARACTER*50 NOTE_FILE        /'steady/cufd25C21.note'/
C            >>>>>>>>>>>>>>>>>>> f77 -O -r8 -o xcu25C21 rbcufd.f

C     set initialization
      Ra = 2 500.0d0
      rolls=2.0d0
      IX=0
      IY=0
      cn=0.5d0
      delta=1.0d0
      rKb= 1.0d0
      bulk=0.0d0
      Th = 1.5d0
      Tl = 0.5d0
      rLy= Ny*delta
      rLx= 2.0d0*rLy
C      Pr = 0.025d0
      Pr = 1.0d0
      eta= rLy*Sqrt(Pr/Ra)
      rkappa=eta/Pr
      g = -rKb*(Th-Tl)/rLy
      dt=delta*cn/Sqrt(2.0*Th)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      tottime=15000.0d0
      maxiter=INT(tottime/dt)
      iprkefreq=INT(maxiter/100)
c$IF(THREE_D_STOKES_FLUID)
      bulk=eta/3.0d0
c$ENDIF
C     Write info to note file
      Pi=4.0d0*DATAN(1.0d0)
      open(9,file=NOTE_FILE,form='formatted',STATUS= 'UNKNOWN')
      write(9,'(A)') "(*rb note file*)"
      write(9,'("rolls,Nx,Ny",f8.2,i8,i8)') rolls,Nx,Ny
      write(9,'("Rn,g,eta,rLy ",4f16.6)') Ra,g,eta,rLy
      close(9)
C
C     initialize variables
      CALL INITIAL(IN_STATE_FILE,dt)
CCCCCC
c$IF(PRINT_KE_FILE)
      totrho=0.0d0
      totKx=0.0d0
      totKy=0.0d0
      DO i=1,Nx
         totrho=totrho+Rho(i,1)
         DO j=2,Ny
            totrho=totrho+Rho(i,j)
            totKx=totKx+G1rho(i,j)*u(i,j)*u(i,j)/2.0d0
            totKy=totKy+G1rho(i,j)*v(i,j)*v(i,j)/2.0d0
         END DO
      END DO
      open(14,file=KE_FILE,form='formatted',ACCESS='APPEND'
     2     ,STATUS= 'UNKNOWN')
      write(14,'(f10.3,2(3x,f18.14))')
     2     time, totKX/totrho,totkY/totrho
      close(14)
c$ENDIF
CCCCCC
C     START ITERATION
      DO iter=1,maxiter
         CALL RK(dt)
         time=time+dt
      write(9,'(A)') "at RK"
c$IF(PRINT_KE_FILE) uncomment the following lines
C     output to ke grid center data file
         if (jmod(iter,iprkefreq).eq.0) then
            totrho=0.0d0
            DO i=1,Nx
               DO j=1,Ny
                  totrho=totrho+Rho(i,j)
               END DO
            END DO
            totKx=0.0d0
            totKy=0.0d0
            DO i=1,Nx
               DO j=2,Ny
                  totKx=totKx+G1rho(i,j)*u(i,j)*u(i,j)/2.0d0
                  totKy=totKy+G1rho(i,j)*v(i,j)*v(i,j)/2.0d0
               END DO
            END DO
            open(14,file=KE_FILE,form='formatted',ACCESS='APPEND',
     2           STATUS= 'UNKNOWN')
            write(14,'(f10.3,2(3x,f18.14))')
     2           time, totKX/totrho,totkY/totrho
            close(14)
         end if
c$ENDIF
      END DO
c$IF(PRINT_STATE_FILE) uncomment the following lines
      open(22,file=OUT_STATE_FILE,form='formatted',ACCESS='APPEND',
     2     STATUS= 'UNKNOWN')
      DO i=1,Nx
         DO j=1,Ny
            write (22,'(2i5,4f24.18)') i,j,rho(i,j),u(i,j),v(i,j),e(i,j)
         END DO
         j=Ny+1
         write (22,'(2i5,4f24.18)') i,j,u(i,j),v(i,j),e(i,j)
      END DO
      close(22)
c$ENDIF
      STOP
      END
C
C-----------------------------------------------------------------------X
C     Sets initial and boundary values
      SUBROUTINE INITIAL(IN_STATE_FILE,dt)
      INCLUDE 'rbcu.h'
      CHARACTER*50 IN_STATE_FILE
      DIMENSION f1(Nx), f2(Ny+1), f1b(Nx), f2b(Ny+1)
C     initialize grid point values (note period bc in x-direction)
cc$IF(READ_STATE_FILE) uncomment these lines
c      open(22,file=IN_STATE_FILE,form='formatted',status='OLD')
c      DO i=1,Nx
c         DO j=1,Ny
c            read (22,'(2i5,4f24.18)',END=99)ii,jj,rho(i,j),
c     2           u(i,j),v(i,j),e(i,j)
c            if(i.ne.ii) write(*,*) "error"
c            if(j.ne.jj) write(*,*) "error"
c         END DO
c         j=Ny+1
c         read (22,'(2i5,4f24.18)',END=99)ii,jj,
c     2        u(i,j),v(i,j),e(i,j)
c      END DO
c 99   close(22,STATUS='KEEP')
cC     enforce boundary conditions
c      DO i=1,Nx
c         e(i,1)=Th
c         u(i,1)=0.0d0
c         v(i,1)=0.0d0
c         u(i,Ny+1)=0.0d0
c         v(i,Ny+1)=0.0d0
c         e(i,Ny+1)=Tl
c      END DO
c$ELSE
C     initialize velocity field
      DO i=1,Nx
         DO j=1,Ny+1
            u(i,j)=0.0d0
            v(i,j)=0.0d0
         END DO
      END DO
C     initialize internal energy
      DO j=1,Ny+1
         T = Th + (Tl-Th)*(j-1)/(Ny)
         DO i=1,Nx
            e(i,j)=T
         END DO
      END DO
C     initialize cell center values
      DO j=1,Ny
         DO i=1,Nx
            rho(i,j)=1.0d0
         END DO
      END DO
C     PERTURBATION
```

```
      j=int(Ny/2)
      DO i=1,Nx
            rho(i,j)=rho(i,j)*(1.0d0+0.1d0*Dsin(i*Pi/Nx))
      END DO

cc$IF(RAYLEIGH_BENARD) uncomment these lines
cC     initialize velocity field
c      DO i=1,Nx
c         u(i,1)=0.0d0
c         v(i,1)=0.0d0
c         u(i,Ny+1)=0.0d0
c         v(i,Ny+1)=0.0d0
c      END DO
c      velfacy=0.10d0
c      velfacx= velfacy/(rolls*0.50d0)
c      DO j=2,Ny
c         facj=(j-(Ny+1)/2)*Pi/(Ny-1)
c         DO i=1,Nx
cC            faci=((i+0.50d0)-(Nx+1)/2)*Pi/(Nx-1)
cC            faci=((i-1)-Nx/2)*Pi/Nx
c            faci=((float(i)-0.50d0)-Nx/2)*Pi/Nx
c            u(i,j)=DSin(rolls*faci)*DSin(2.0d0*facj)*velfacx
c            v(i,j)=DCos(rolls*faci)*DCos(facj)*velfacy
c         END DO
c      END DO
cc$ENDIF
c$ENDIF
C     interpolate density value to all grids
C     interpolate new density to other grids
C     rho -> G4rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=rho(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4rho(i,j)=f2b(j-1)
         END DO
      END DO
C     rho -> G3rho
      DO j=1,Ny
         DO i=1,Nx
            f1(i)=rho(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,Nx)
         CALL CubePbINT(f1,f1b,delta,Nx)
         G3rho(1,j)=f1b(Nx)
         DO i=2,Nx
            G3rho(i,j)=f1b(i-1)
         END DO
      END DO
C     G3rho -> G1rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=G3rho(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G1rho(i,j)=f2b(j-1)
         END DO
      END DO
C
C     initialize "conserved" variables on grid
      DO j=1,Ny+1
         DO i=1,Nx
            Xm(i,j) = G1rho(i,j)*u(i,j)
            Ym(i,j) = G1rho(i,j)*v(i,j)
            Te(i,j) = G1rho(i,j)*(e(i,j)
     &           +(u(i,j)*u(i,j)+v(i,j)*v(i,j))/2.0d0)
         END DO
      END DO
cc$IF(INITIAL_TRAP) uncomment these lines
c      uuumax=0.0d0
c      rhominmin=2.0d0
c      Ni=int(rolls/2.0)
c      iNx=INT(Nx/Ni)
c      inittime=INT(2000/dt)
c      DO iter=1,inittime
c         CALL RK1(dt)
c         time=time+dt
cc         if (jmod(iter,100).eq.0) then
c            DO ii=1,Ni-1
c               DO i=1,iNx
c                  in=ii*iNx+i
c                  rho(in,1)=rho(i,1)
c                  DO j=2,Ny
c                     G1rho(in,j)=G1rho(i,j)
c                     rho(in,j)=rho(i,j)
c                     Xm(in,j) = Xm(i,j)
c                     Ym(in,j) = Ym(i,j)
c                     Te(in,j) = Te(i,j)
c                     u(in,j) = u(i,j)
c                     v(in,j) = v(i,j)
c                     e(in,j) = e(i,j)
c                  END DO
c               END DO
c            END DO
c      END DO
c      DO ii=1,Ni-1
c         DO i=1,iNx
c            in=ii*iNx+i
c            rho(in,1)=rho(i,1)
c            DO j=2,Ny
c               G1rho(in,j)=G1rho(i,j)
c               rho(in,j)=rho(i,j)
c               Xm(in,j) = Xm(i,j)
c               Ym(in,j) = Ym(i,j)
c               Te(in,j) = Te(i,j)
c               u(in,j) = u(i,j)
c               v(in,j) = v(i,j)
c               e(in,j) = e(i,j)
c            END DO
c         END DO
c      END DO
cc$ENDIF
      RETURN
      END
C
C-----------------------------------------------------------------------X
C

      INCLUDE 'rk.sub.f'

      INCLUDE 'cubic.sub.f'
```

B.1.3 Cubic spline subroutines

The cubic spline representation of the variable is determined for periodic (PB) or fixed (FB) boundaries. The interpolant (INT) or gradient (DX) is returned. The subroutine CUBEPBDX, for example, returns the gradient for a periodic boundary case.

```
CUBIC.F
Contains subroutines related to the
cubic spline interpolation of state variables.
C
C-----------------------------------------------------------------------X
C
      subroutine TRIDIA(a,b,c,r,u,n)
C     subroutine tridia solves system Au=r where A is tridiagonal
C     with major and minor diags c and a resp. and main diag b.
C     b is diagonal, a/c are lower/upper off-diagonals
      IMPLICIT REAL*8 (A-H,O-Z)
      parameter (nmax=1000)
      dimension gam(nmax),a(n),b(n),c(n),r(n),u(n)
      if(b(1).eq.0.0d0) pause
      bet=b(1)
      u(1) = r(1)/bet
      do 11 j=2,n
         gam(j)=c(j-1)/bet
         bet=b(j)-a(j)*gam(j)
         if(bet.eq.0.0d0) then
            write(*,*) j,a(j),b(j),c(j)
             pause
         end if
         u(j)=(r(j)-a(j)*u(j-1))/bet
 11   continue
      do 12 j=n-1,1,-1
         u(j)=u(j)-gam(j+1)*u(j+1)
 12   continue
      return
      end
C
C-----------------------------------------------------------------------X
C*****Random Number Generator
      FUNCTION RANDOM(IX,IY)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      I = 1029*IX+1731
      J = I + 1029*IY + 507*IX - 1731
```

```
      IX = MOD(I,2048)
      J = J + (I-IX)/2048
      IY = MOD(J,2048)
      RANDOM = (IX+2048*IY)/4194304.0d0
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CubePbDX(f,fp,h,n)
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
C     ASSUMES M=0 at endpoints
C     USES DBTDLU and DBTDSOL
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fp(n)
      dimension am(n,n), bm(3,n-1), rm(1,n-1), cm(n,1), xm(n)
      data ldb/3/, ldr/1/, ml/1/, mu/1/, mp/1/
      ldc=n
c     Set elements of matrix A in normal (full) form
      do 20 j = 1,n
         do 10 i = 1,n
            if (j-i .le. mu .and. i-j .le. ml) then
               am(i,j) = 1.0d0
            else
               am(i,j) = 0.0d0
            endif
 10      continue
         am(j,j) = 4.0d0
 20   continue
      am(1,n)=1.0d0
      am(n,1)=1.0d0
c     Load b, r, c from A
      m = ml + mu + 1
      nmp = n - mp
      do 50 j = 1,nmp
         do 30 k = 1,m
            i = k + j - mu - 1
 30         if (i .ge. 1 .and. i .le. nmp) bm(k,j) = am(i,j)
            do 40 k = 1,mp
 40            rm(k,j) = am(k+nmp,j)
 50         continue
            do 70 k = 1,mp
               do 60 i = 1,n
 60               cm(i,k) = am(i,k+nmp)
 70   continue
c
C     SETUP RHS:
      xm(1)=(6.0d0/h/h)*(f(2)-2.0d0*f(1)+f(N))
      DO i=2,N-1
         xm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      xm(N)=(6.0d0/h/h)*(f(1)-2.0d0*f(N)+f(N-1))
c
c Call solver routines and print results
      call dbbfa (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, ier)
      if (ier .ne. 0) then
        write(6,130) ier
 130    format(//'Error return from dbbFA.  ier =',i4)
        stop
      endif
      call dbbsl (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, xm)
      DO i=1,n-1
         fp(i)=(f(i+1)-f(i))/h - (h/24.0d0)*(xm(i+1)-xm(i))
      END DO
      fp(n)=(f(1)-f(n))/h - (h/24.0d0)*(xm(1)-xm(n))
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CDPbDX(f,fp,h,n)
C     USES CENTERED DIFFERENCE
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fp(n)
      DO i=1,n-1
         fp(i)=(f(i+1)-f(i))/h
      END DO
      fp(n)=(f(1)-f(n))/h
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CubePbINT(f,fbar,h,n)
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
C     ASSUMES M=0 at endpoints
C     USES DBTDLU and DBTDSOL
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fbar(n)
      dimension am(n,n), bm(3,n-1), rm(1,n-1), cm(n,1), xm(n)
      data ldb/3/, ldr/1/, ml/1/, mu/1/, mp/1/
      ldc=n
c     Set elements of matrix A in normal (full) form
      do 20 j = 1,n
         do 10 i = 1,n
            if (j-i .le. mu .and. i-j .le. ml) then
               am(i,j) = 1.0d0
            else
               am(i,j) = 0.0d0
            endif
 10      continue
         am(j,j) = 4.0d0
 20   continue
      am(1,n)=1.0d0
      am(n,1)=1.0d0
c     Load b, r, c from A
      m = ml + mu + 1
      nmp = n - mp
      do 50 j = 1,nmp
         do 30 k = 1,m
            i = k + j - mu - 1
 30         if (i .ge. 1 .and. i .le. nmp) bm(k,j) = am(i,j)
            do 40 k = 1,mp
 40            rm(k,j) = am(k+nmp,j)
 50         continue
            do 70 k = 1,mp
               do 60 i = 1,n
 60               cm(i,k) = am(i,k+nmp)
 70   continue
c
C     SETUP RHS:
      xm(1)=(6.0d0/h/h)*(f(2)-2.0d0*f(1)+f(N))
      DO i=2,N-1
         xm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      xm(N)=(6.0d0/h/h)*(f(1)-2.0d0*f(N)+f(N-1))
c
c Call solver routines and print results
      call dbbfa (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, ier)
      if (ier .ne. 0) then
        write(6,130) ier
 130    format(//'Error return from dbbFA.  ier =',i4)
        stop
      endif
      call dbbsl (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, xm)
      DO i=1,n-1
         fbar(i)=(f(i+1)+f(i))/2.0d0 - (h*h/16.0d0)*(xm(i+1)+xm(i))
      END DO
      fbar(n)=(f(1)+f(n))/2.0d0 - (h*h/16.0d0)*(xm(1)+xm(n))
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CubeFbDx(f,fy,h,n)
C     differentiates with fixed b.c's
C     ip=1 -> odd function, ip=2 -> even function
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fy(n), rMom(n+1)
      dimension am(n+1),bm(n+1),cm(n+1),rm(n+1)
C     SETUP RHS:
      rm(1)=0.0d0
      DO i=2,N
         rm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      rm(N+1)=0.0d0
C     SETUP LHS:
      am(1)=0.0d0
      bm(1)=4.0d0
      cm(1)=0.0d0
      DO i=2,N
         am(i)=1.0d0
         bm(i)=4.0d0
         cm(i)=1.0d0
      END DO
      am(N+1)=0.0d0
      bm(N+1)=4.0d0
      cm(N+1)=0.0d0
C     SOLVE SYSTEM:
      call tridia(am,bm,cm,rm,rMom,N+1)
      DO i=1,N
         fy(i)=(f(i+1)-f(i))/h - (h/24.0d0)*(rMom(i+1)-rMom(i))
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CDFbDx(f,fy,h,n)
C     USES CENTERED DIFFERENCE
C     differentiates with fixed b.c's
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fy(n)
      DO i=1,N
         fy(i)=(f(i+1)-f(i))/h
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CubeFbInt(f,fbar,h,n)
C     interpolation with fixed b.c's
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fbar(n), rMom(n+1)
      dimension am(n+1),bm(n+1),cm(n+1),rm(n+1)
C     SETUP RHS:
      rm(1)=0.0d0
      DO i=2,N
         rm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      rm(N+1)=0.0d0
C     SETUP LHS:
      am(1)=0.0d0
      bm(1)=4.0d0
      cm(1)=0.0d0
      DO i=2,N
         am(i)=1.0d0
         bm(i)=4.0d0
         cm(i)=1.0d0
      END DO
      am(N+1)=0.0d0
      bm(N+1)=4.0d0
      cm(N+1)=0.0d0
c$IF(FD)
c      DO i=1,N
c         fbar(i)=(f(i+1)+f(i))/2.0d0
c      END DO
c$ENDIF
c$IF(CUBIC)
C     SOLVE SYSTEM:
      call tridia(am,bm,cm,rm,rMom,N+1)
      DO i=1,N
         fbar(i)=(f(i+1)+f(i))/2.0d0 - (h*h/16.0d0)*(rMom(i+1)+rMom(i))
      END DO
c$ENDIF
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE DBBFA (N, ML, MU, MP, B, R, C, LDB, LDR, LDC, IER)
      DOUBLE PRECISION B(LDB,*), R(LDR,*), C(LDC,*)
```

```
C-----------------------------------------------------------------------
C LU factorization of border-banded matrix without pivoting.
C Double precision version.
C Call DBBFA to factor a given matrix A, then call DBBSL to solve
C a system Ax = b with given vector b.
C
C Input arguments:
C
C N    = order of matrix
C ML   = lower half-bandwidth (.LT. N)
C MU   = upper half-bandwidth (.LT. N)
C MP   = width of border at right and lower edges  (.LE. N).
C B    = LDB by N-MP array with band elements (LDB .ge. ML+MU+1).
C        Element a(i,j) is to be stored in B(I-J+MU+1,J) for
C          1 .LE. I,J .LE. N-MP  and  -ML .LE. J-I .LE. MU.
C R    = LDR by N-MP array with lower edge elements (LDR .GE. MP).
C        a(i,j) is to be stored in R(I-N+MP,J) for
C          N-MP+1 .LE. I .LE. N  and  1 .LE. J .LE. N-MP.
C C    = LDC by MP array with right edge elements (LDC .GE. N).
C        a(i,j) is to be stored in C(I,J-N+MP) for
C          1 .LE. I .LE. N  and  N-MP+1 .LE. J .LE. N.
C LDB  = declared first dimension of B, must be .GE. ML+MU+1.
C LDR  = declared first dimension of R, must be .GE. MP.
C LDC  = declared first dimension of C, must be .GE. N.
C
C
C Output arguments:
C
C IER  = error flag.
C        IER = -1 means integer input parameters were illegal.
C        IER = 0 means the LU factorization was successful.
C        IER = K .GT. 1 means the K-th pivot was found to be zero.
C B, R, C arrays contain a representation of the L and U factors.
C
C Routines called: DSCAL, DAXPY (from the BLAS collection)
C-----------------------------------------------------------------------
      DOUBLE PRECISION ONE, T, ZERO
      DATA ONE/1.0D0/, ZERO/0.0D0/
C
C Test inputs.
      IF (N .LE. 0) GO TO 500
      IF (ML .GE. N .OR. MU .GE. N .OR. MP .GT. N) GO TO 500
      IF (LDB .LT. ML+MU+1 .OR. LDR .LT. MP .OR. LDC .LT. N) GO TO 500
C
C Set auxiliary constants.
      NM1 = N - 1
      NMP = N - MP
      NMPP1 = NMP + 1
      MU1 = MU + 1
      MU2 = MU + 2
      IER = 0
C
C Generate LU factors of band and lower border (stages 1 .. N-MP).
      DO 160 K = 1,NMP
        KP1 = K + 1
C Form K-th pivot.
        T = B(MU1,K)
        IF (T .EQ. ZERO) GO TO 550
        T = ONE/T
        B(MU1,K) = T
C Form L elements in band.
         LEN1 = MIN(NMP-K,ML)
         T = -T
         CALL DSCAL (LEN1, T, B(MU2,K), 1)
C Form L elements in lower border.
         CALL DSCAL (MP, T, R(1,K), 1)
C Update matrix elements in band and lower border.
        JHI = MIN(NMP,K+MU)
        DO 150 J = KP1,JHI
          T = B(K-J+MU1,J)
          CALL DAXPY (LEN1, T, B(MU2,K), 1, B(KP1-J+MU1,J), 1)
          CALL DAXPY (MP, T, R(1,K), 1, R(1,J), 1)
 150      CONTINUE
 160    CONTINUE
C
C Update matrix elements in right border from stages 1 .. N-MP.
      DO 240 K = 1,NMP
        KP1 = K + 1
        DO 230 J = NMPP1,N
          T = C(K,J-NMP)
          LEN1 = MIN(NMP-K,ML)
          CALL DAXPY (LEN1, T, B(MU2,K), 1, C(KP1,J-NMP), 1)
          CALL DAXPY (MP, T, R(1,K), 1, C(NMPP1,J-NMP), 1)
 230      CONTINUE
 240    CONTINUE
C
C Generate LU factors in lower right block (stages N-MP+1 .. N-1).
C
      DO 340 K = NMPP1,NM1
        KP1 = K + 1
C Form K-th pivot.
        T = C(K,K-NMP)
        IF (T .EQ. ZERO) GO TO 550
        T = ONE/T
        C(K,K-NMP) = T
C Form L elements.
        T = -T
        CALL DSCAL (N-K, T, C(KP1,K-NMP), 1)
C Update matrix elements.
        DO 330 J = KP1,N
          T = C(K,J-NMP)
          CALL DAXPY (N-K, T, C(KP1,K-NMP), 1, C(KP1,J-NMP), 1)
 330      CONTINUE
 340    CONTINUE
C
C Form last pivot and return
      IF (MP .EQ. 0) RETURN
      K = N
      T = C(N,MP)
      IF (T .EQ. ZERO) GO TO 550
      C(N,MP) = ONE/T
      RETURN
C
C Error return on illegal input.
 500  IER = -1
      RETURN
C Error return on zero pivot.
 550  IER = K
      RETURN
      END
C
      SUBROUTINE DBBSL (N, ML, MU, MP, B, R, C, LDB, LDR, LDC, X)
      DOUBLE PRECISION B(LDB,*), R(LDR,*), C(LDC,*), X(*)
C-----------------------------------------------------------------------
C Solution of border-banded matrix without pivoting, given the LU
C factorization by DBBFA.  Assumes factorization was successful and
C factor information has not been changed.  For systems Ax = b,
C with A factored by DBBFA, call DBBSL once for each RHS vector b.
C
C Input arguments:
C
C N    = order of matrix
C ML   = lower half-bandwidth
C MU   = upper half-bandwidth
C MP   = width of border at right and lower edges
C B, R, C = arrays as output by factorization
C LDB, LDR, LDC  = declared first dimension of B, R, C, respectively
C          (assumed to be the same as in factorization)
C X    = array of length .GE. N containing right-hand side vector b
C
C
C Output arguments:
C
C X    = solution vector
C
C Routines called: DSCAL, DAXPY (from the BLAS collection)
C-----------------------------------------------------------------------
      DOUBLE PRECISION T
C Set auxiliary constants.
      NM1 = N - 1
      NMP = N - MP
      NMPP1 = NMP + 1
      MU1 = MU + 1
      MU2 = MU + 2
C
C Apply inverse of L (columns 1 .. N-MP).
      DO 130 K = 1,NMP
        KP1 = K + 1
        LEN1 = MIN(NMP-K,ML)
        CALL DAXPY (LEN1, X(K), B(MU2,K), 1, X(KP1), 1)
        CALL DAXPY (MP, X(K), R(1,K), 1, X(NMPP1), 1)
 130    CONTINUE
C
C Apply inverse of L (columns N-MP+1 .. N-1).
      DO 150 K = NMPP1,NM1
        KP1 = K + 1
        CALL DAXPY (N-K, X(K), C(KP1,K-NMP), 1, X(KP1), 1)
 150    CONTINUE
C
C Apply inverse of U (columns N .. N-MP+1).
      DO 220 K = N,NMPP1,-1
        X(K) = X(K)*C(K,K-NMP)
        T = -X(K)
        CALL DAXPY (K-1, T, C(1,K-NMP), 1, X(1), 1)
 220    CONTINUE
C
C Apply inverse of U (columns N-MP .. 1).
      DO 240 K = NMP,1,-1
        X(K) = X(K)*B(MU1,K)
        ILO = MAX(1,K-MU)
        IBLO = ILO - K + MU1
        T = -X(K)
        CALL DAXPY (K-ILO, T, B(IBLO,K), 1, X(ILO), 1)
 240    CONTINUE
C
      RETURN
      END
C
      subroutine daxpy(n,da,dx,incx,dy,incy)
c
c     constant times a vector plus a vector.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c      double precision dx(1),dy(1),da **********************
      double precision dx(n),dy(n),da
      integer i,incx,incy,ix,iy,m,mp1,n
c
      if(n.le.0)return
      if (da .eq. 0.0d0) return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        dy(iy) = dy(iy) + da*dx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return
c
c        code for both increments equal to 1
c
c
c        clean-up loop
c
   20 m = mod(n,4)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dy(i) = dy(i) + da*dx(i)
   30 continue
      if( n .lt. 4 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,4
        dy(i) = dy(i) + da*dx(i)
        dy(i + 1) = dy(i + 1) + da*dx(i + 1)
        dy(i + 2) = dy(i + 2) + da*dx(i + 2)
        dy(i + 3) = dy(i + 3) + da*dx(i + 3)
   50 continue
      return
      end
C
      subroutine dscal(n,da,dx,incx)
c
c     scales a vector by a constant.
```

```
c     uses unrolled loops for increment equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c      double precision da,dx(1) **************************
      double precision da,dx(n)

      integer i,incx,m,mp1,n,nincx
c
      if(n.le.0)return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        dx(i) = da*dx(i)
   10 continue
      return
c
c        code for increment equal to 1
c
c        clean-up loop
c
   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dx(i) = da*dx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        dx(i) = da*dx(i)
        dx(i + 1) = da*dx(i + 1)
        dx(i + 2) = da*dx(i + 2)
        dx(i + 3) = da*dx(i + 3)
        dx(i + 4) = da*dx(i + 4)
   50 continue
      return
      end
```

B.1.4 Runge-Kutta subroutines

The time integration is accomplished by fourth-order Runge-Kutta.

```
C-----------------------------------------------------------------X
C
      SUBROUTINE RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
      INCLUDE 'rbcu.h'
      DIMENSION rhonew(Nx,Ny+1),Xmnew(Nx,Ny+1)
      DIMENSION Ymnew(Nx,Ny+1),Tenew(Nx,Ny+1)
      DIMENSION f1(Nx), f2(Ny+1), f1b(Nx),f2b(Ny+1), fx(Nx), fy(Ny+1)
C     interpolate new density to other grids
C     rhonew -> G4rho
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=rhonew(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4rho(i,j)=f2b(j-1)
         END DO
      END DO
C     rhonew -> G3rho
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=rhonew(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3rho(1,j)=f1b(iNx)
         DO i=2,iNx
            G3rho(i,j)=f1b(i-1)
         END DO
      END DO
C     G3rho -> G1rho
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G3rho(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G1rho(i,j)=f2b(j-1)
         END DO
      END DO
C     determine state variables u,v,e from the "new" ones
      DO j=2,Ny
         DO i = 1,iNx
            u(i,j) = Xmnew(i,j)/G1rho(i,j)
            v(i,j) = Ymnew(i,j)/G1rho(i,j)
            e(i,j) = Tenew(i,j)/G1rho(i,j)
     2          -(u(i,j)*u(i,j)+v(i,j)*v(i,j))/2.0d0
         END DO
      END DO
C     interpolate state variables
C     u -> G4u
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=u(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4u(i,j)=f1b(i)
         END DO
      END DO
C     v -> G4v
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=v(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4v(i,j)=f1b(i)
         END DO
      END DO
C     e -> G4e
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=e(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4e(i,j)=f1b(i)
         END DO
      END DO
C     u -> G3u
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=u(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3u(i,j)=f2b(j)
         END DO
      END DO
C     e -> G3e   CUSP
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=e(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3e(i,j)=f2b(j)
         END DO
      END DO
C     v -> G3v
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=v(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3v(i,j)=f2b(j)
         END DO
      END DO
C     determine gradients of state variables
C     u->ux (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=u(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            ux(i,j)=fx(i)
         END DO
      END DO
C     v -> vx (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=v(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            vx(i,j)=fx(i)
         END DO
      END DO
C     e -> ex (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=e(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
```

```
         DO i=1,iNx
            ex(i,j)=fx(i)
         END DO
      END DO
C     u -> uy (even 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=u(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,2)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            uy(i,j)=fy(j)
         END DO
      END DO
C     e -> ey (even 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=e(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,2)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            ey(i,j)=fy(j)
         END DO
      END DO
C     v -> vy (odd 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=v(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,1)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            vy(i,j)=fy(j)
         END DO
      END DO
C     interpolate velocity gradients to opposite grid
C     ux->G2ux
      DO i=1,iNx
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=ux(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G2ux(i,j)=f2b(j)
         END DO
      END DO
C     vx->G2vx
      DO i=1,iNx
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=vx(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G2vx(i,j)=f2b(j)
         END DO
      END DO
C     uy -> G2uy
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=uy(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G2uy(i,j)=f1b(i)
         END DO
      END DO
C     vy -> G2vy
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=vy(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G2vy(i,j)=f1b(i)
         END DO
      END DO
C     G2ux->G3ux
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=G2ux(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3ux(1,j)=f1b(iNx)
         DO i=2,iNx
            G3ux(i,j)=f1b(i-1)
         END DO
      END DO
C     G2vx->G3vx
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=G2vx(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3vx(1,j)=f1b(iNx)
         DO i=2,iNx
            G3vx(i,j)=f1b(i-1)
         END DO
      END DO
C     G2uy->G4uy
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G2uy(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4uy(i,j)=f2b(j-1)
         END DO
      END DO
C     G2vy->G4vy
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G2vy(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4vy(i,j)=f2b(j-1)
         END DO
      END DO
C
C     determine FLUX values
      DO i=1,iNx
         Grho(i,1)=0.0d0
         Grho(i,Ny+1)=0.0d0
         DO j=2,Ny
            qx=-rkappa*ex(i,j)
            Sxx=(bulk+eta)*ux(i,j)+(bulk-eta)*G4vy(i,j)
     1          -G4rho(i,j)*G4e(i,j)
            Sxy=eta*(vx(i,j)+G4uy(i,j))
            G4Te=G4e(i,j)+(G4u(i,j)*G4u(i,j)+G4v(i,j)*G4v(i,j))/2.0d0
            Grho(i,j)=G4rho(i,j)*G4v(i,j)
            Fxm(i,j)=G4rho(i,j)*G4u(i,j)*G4u(i,j)-Sxx
            Fym(i,j)=G4rho(i,j)*G4u(i,j)*G4v(i,j)-Sxy
            Fte(i,j)=G4rho(i,j)*G4u(i,j)*G4Te +
     1          qx - G4u(i,j)*Sxx - G4v(i,j)*Sxy
         END DO
      END DO
      DO i=1,iNx
         DO j=1,Ny
            qy=-rkappa*ey(i,j)
            Syy=(bulk+eta)*vy(i,j)+(bulk-eta)*G3ux(i,j)
     1          -G3rho(i,j)*G3e(i,j)
            Sxy=eta*(G3vx(i,j)+uy(i,j))
            G3Te=G3e(i,j)+(G3u(i,j)*G3u(i,j)+G3v(i,j)*G3v(i,j))/2.0d0
            Frho(i,j)=G3rho(i,j)*G3u(i,j)
            Gxm(i,j)=G3rho(i,j)*G3u(i,j)*G3v(i,j)-Sxy
            Gym(i,j)=G3rho(i,j)*G3v(i,j)*G3v(i,j)-Syy
            Gte(i,j)=G3rho(i,j)*G3v(i,j)*G3Te +
     1          qy - G3u(i,j)*Sxy - G3v(i,j)*Syy
         END DO
      END DO
C     Compute Sources (rho has no source)
      DO i=1,iNx
         DO j=1,Ny
            rhodot(i,j)=0.0d0
            Xmdot(i,j)=0.0d0
            Ymdot(i,j)=G1rho(i,j)*g
            Tedot(i,j)=G1rho(i,j)*v(i,j)*g
         END DO
      END DO
C     Compute divergence of the fluxes
C     Frho->rhodot  (periodic 1->2)
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=Frho(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
C         CALL CubePbDx(f1,fx,delta,iNx)
         CALL CDPbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            RhoDot(i,j)=RhoDot(i,j)-fx(i)
         END DO
      END DO
C     Grho->rhodot (odd 1->2)
      DO i=1,iNx
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=Grho(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL CubeFbDxV1(f2,fy,delta,Ny,1)
C         CALL CubeFbDx(f2,fy,delta,Ny)
         CALL CDFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            RhoDot(i,j)=RhoDot(i,j)-fy(j)
         END DO
      END DO
C     Fxm-> Xmdot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fxm(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
C         CALL CubePbDx(f1,fx,delta,iNx)
         CALL CDPbDx(f1,fx,delta,iNx)
         XmDot(1,j)=XmDot(1,j)-fx(iNx)
         DO i=2,iNx
            XmDot(i,j)=XmDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gxm-> Xmdot  (odd 2->1)
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=Gxm(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,1)
C         CALL CubeFbDx(f2,fy,delta,Ny-1)
         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            XmDot(i,j)=XmDot(i,j)-fy(j-1)
         END DO
      END DO
C     Fym-> Ymdot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fym(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
C         CALL CubePbDx(f1,fx,delta,iNx)
         CALL CDPbDx(f1,fx,delta,iNx)
         YmDot(1,j)=YmDot(1,j)-fx(iNx)
         DO i=2,iNx
            YmDot(i,j)=YmDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gym-> Ymdot  (even 2->1)
      DO i=1,iNx
```

```
         DO j=1,Ny
            f2(j)=Gym(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,2)
C         CALL CubeFbDx(f2,fy,delta,Ny-1)
         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            YmDot(i,j)=YmDot(i,j)-fy(j-1)
         END DO
      END DO
C     Fte-> Tedot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fte(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
C         CALL CubePbDx(f1,fx,delta,iNx)
         CALL CDPbDx(f1,fx,delta,iNx)
         TeDot(1,j)=TeDot(i,j)-fx(iNx)
         DO i=2,iNx
            TeDot(i,j)=TeDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gte-> Tedot  (odd 2->1)
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=Gte(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,1)
C         CALL CubeFbDx(f2,fy,delta,Ny-1)
         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            TeDot(i,j)=TeDot(i,j)-fy(j-1)
         END DO
      END DO
      RETURN
      END
C
C------------------------------------------------------------ -----------X
C
      SUBROUTINE RK(dt)
      INCLUDE 'rbcu.h'
      DIMENSION rhonew(Nx,Ny), Xmnew(Nx,Nn), Ymnew(Nx,Nn), Tenew(Nx,Nn)
      DIMENSION rhodot1(Nx,Ny),Xmdot1(Nx,Ny),Ymdot1(Nx,Ny),Tedot1(Nx,Ny)
      DIMENSION rhodot2(Nx,Ny),Xmdot2(Nx,Ny),Ymdot2(Nx,Ny),Tedot2(Nx,Ny)
      DIMENSION rhodot3(Nx,Ny),Xmdot3(Nx,Ny),Ymdot3(Nx,Ny),Tedot3(Nx,Ny)
      DIMENSION rhodot4(Nx,Ny),Xmdot4(Nx,Ny),Ymdot4(Nx,Ny),Tedot4(Nx,Ny)
c      write(9,'(A)') "at RK"
      DO i=1,Nx
         DO j=1,Ny
            rhonew(i,j)=rho(i,j)
            Xmnew(i,j) = Xm(i,j)
            Ymnew(i,j) = Ym(i,j)
            Tenew(i,j) = Te(i,j)
         END DO
         Xmnew(i,Ny+1) = Xm(i,Ny+1)
         Ymnew(i,Ny+1) = Ym(i,Ny+1)
         Tenew(i,Ny+1) = Te(i,Ny+1)
      END DO
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot1(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot1(i,j)=rhodot(i,j)
            Xmdot1 (i,j) =Xmdot(i,j)
            Ymdot1 (i,j) =Ymdot(i,j)
            Tedot1 (i,j) =Tedot(i,j)
         END DO
      END DO
 1    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot1(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot1(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j)+ dt*Xmdot1(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j)+ dt*Ymdot1(i,j)/2.0d0
            Tenew(i,j) = Te(i,j)+ dt*Tedot1(i,j)/2.0d0
         END DO
      END DO
 2    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot2(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot2(i,j)=rhodot(i,j)
            Xmdot2(i,j) = Xmdot(i,j)
            Ymdot2(i,j) = Ymdot(i,j)
            Tedot2(i,j) = Tedot(i,j)
         END DO
      END DO
 3    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot2(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot2(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot2(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot2(i,j)/2.0d0
            Tenew(i,j) = Te(i,j) +dt*Tedot2(i,j)/2.0d0
         END DO
      END DO
 4    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot3(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot3(i,j)=rhodot(i,j)
            Xmdot3(i,j) = Xmdot(i,j)
            Ymdot3(i,j) = Ymdot(i,j)
            Tedot3(i,j) = Tedot(i,j)
         END DO
      END DO
 5    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot3(i,1)
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot3(i,j)
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot3(i,j)
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot3(i,j)
            Tenew(i,j) = Te(i,j) +dt*Tedot3(i,j)
         END DO
      END DO
 6    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot4(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot4(i,j)=rhodot(i,j)
            Xmdot4(i,j) = Xmdot(i,j)
            Ymdot4(i,j) = Ymdot(i,j)
            Tedot4(i,j) = Tedot(i,j)
         END DO
      END DO
 7    continue
      DO i=1,Nx
         rho(i,1)=rho(i,1)+dt*(rhodot1(i,1)+
     2      2.0d0*(rhodot2(i,1)+rhodot3(i,1))+rhodot4(i,1))/6.0d0
         DO j=2,Ny
            rho(i,j)=rho(i,j)+dt*(rhodot1(i,j)+
     2           2.0d0*(rhodot2(i,j)+rhodot3(i,j))+rhodot4(i,j))/6.0d0
            Xm(i,j)=Xm(i,j)+dt*(Xmdot1(i,j)+
     2           2.0d0*(Xmdot2(i,j)+Xmdot3(i,j))+Xmdot4(i,j))/6.0d0
            Ym(i,j)=Ym(i,j)+dt*(Ymdot1(i,j)+
     2           2.0d0*(Ymdot2(i,j)+Ymdot3(i,j))+Ymdot4(i,j))/6.0d0
            Te(i,j)=Te(i,j)+dt*(Tedot1(i,j)+
     2           2.0d0*(Tedot2(i,j)+Tedot3(i,j))+Tedot4(i,j))/6.0d0
         END DO
      END DO
 8    continue
      RETURN
      END
C
C------------------------------------------------------------ -----------X
C
      SUBROUTINE RKI(dt)
      INCLUDE 'rbcu.h'
      DIMENSION rhonew(Nx,Ny), Xmnew(Nx,Nn), Ymnew(Nx,Nn), Tenew(Nx,Nn)
      DIMENSION rhodot1(Nx,Ny),Xmdot1(Nx,Ny),Ymdot1(Nx,Ny),Tedot1(Nx,Ny)
      DIMENSION rhodot2(Nx,Ny),Xmdot2(Nx,Ny),Ymdot2(Nx,Ny),Tedot2(Nx,Ny)
      DIMENSION rhodot3(Nx,Ny),Xmdot3(Nx,Ny),Ymdot3(Nx,Ny),Tedot3(Nx,Ny)
      DIMENSION rhodot4(Nx,Ny),Xmdot4(Nx,Ny),Ymdot4(Nx,Ny),Tedot4(Nx,Ny)

      Ni=INT(rolls/2.0)
c      if(rolls.eq.4.0) Ni=1
c      if(rolls.eq.4.0) Ni=2
c      if(rolls.eq.6.0) Ni=3
      iNx=INT(Nx/Ni)
      DO i=1,iNx
         DO j=1,Ny
            rhonew(i,j)=rho(i,j)
            Xmnew(i,j) = Xm(i,j)
            Ymnew(i,j) = Ym(i,j)
            Tenew(i,j) = Te(i,j)
         END DO
         Xmnew(i,Ny+1) = Xm(i,Ny+1)
         Ymnew(i,Ny+1) = Ym(i,Ny+1)
         Tenew(i,Ny+1) = Te(i,Ny+1)
      END DO
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
C
      DO i=1,iNx
         rhodot1(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot1(i,j)=rhodot(i,j)
            Xmdot1 (i,j) =Xmdot(i,j)
            Ymdot1 (i,j) =Ymdot(i,j)
            Tedot1 (i,j) =Tedot(i,j)
         END DO
      END DO
 1    continue
      DO i=1,iNx
         rhonew(i,1)=rho(i,1)+dt*rhodot1(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot1(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j)+ dt*Xmdot1(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j)+ dt*Ymdot1(i,j)/2.0d0
            Tenew(i,j) = Te(i,j)+ dt*Tedot1(i,j)/2.0d0
         END DO
      END DO
 2    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
C
      DO i=1,iNx
         rhodot2(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot2(i,j)=rhodot(i,j)
            Xmdot2(i,j) = Xmdot(i,j)
            Ymdot2(i,j) = Ymdot(i,j)
            Tedot2(i,j) = Tedot(i,j)
         END DO
      END DO
 3    continue
      DO i=1,iNx
         rhonew(i,1)=rho(i,1)+dt*rhodot2(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot2(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot2(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot2(i,j)/2.0d0
            Tenew(i,j) = Te(i,j) +dt*Tedot2(i,j)/2.0d0
         END DO
      END DO
 4    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
C
      DO i=1,iNx
         rhodot3(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot3(i,j)=rhodot(i,j)
            Xmdot3(i,j) = Xmdot(i,j)
            Ymdot3(i,j) = Ymdot(i,j)
            Tedot3(i,j) = Tedot(i,j)
         END DO
      END DO
 5    continue
      DO i=1,iNx
         rhonew(i,1)=rho(i,1)+dt*rhodot3(i,1)
         DO j=2,Ny
```

```
            rhonew(i,j)=rho(i,j)+dt*rhodot3(i,j)
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot3(i,j)
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot3(i,j)
            Tenew(i,j) = Te(i,j) +dt*Tedot3(i,j)
         END DO
      END DO
 6    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
C
      DO i=1,iNx
         rhodot4(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot4(i,j)=rhodot(i,j)
            Xmdot4(i,j) = Xmdot(i,j)
            Ymdot4(i,j) = Ymdot(i,j)
            Tedot4(i,j) = Tedot(i,j)
         END DO
      END DO
 7    continue
      DO i=1,iNx
         rho(i,1)=rho(i,1)+dt*(rhodot1(i,1)+
     2        2.0d0*(rhodot2(i,1)+rhodot3(i,1))+rhodot4(i,1))/6.0d0
         DO j=2,Ny
            rho(i,j)=rho(i,j)+dt*(rhodot1(i,j)+
     2          2.0d0*(rhodot2(i,j)+rhodot3(i,j))+rhodot4(i,j))/6.0d0
            Xm(i,j)=Xm(i,j)+dt*(Xmdot1(i,j)+
     2          2.0d0*(Xmdot2(i,j)+Xmdot3(i,j))+Xmdot4(i,j))/6.0d0
            Ym(i,j)=Ym(i,j)+dt*(Ymdot1(i,j)+
     2          2.0d0*(Ymdot2(i,j)+Ymdot3(i,j))+Ymdot4(i,j))/6.0d0
            Te(i,j)=Te(i,j)+dt*(Tedot1(i,j)+
     2          2.0d0*(Tedot2(i,j)+Tedot3(i,j))+Tedot4(i,j))/6.0d0
         END DO
      END DO
c
      iNx2=iNx/2
      DO i=1,iNx2-1
         DO j=1,Ny
            rho(iNx2+i,j)=rho(iNx2-i,j)
         END DO
      END DO
      DO i=1,iNx2
         DO j=2,Ny
            Xm(iNx2+i,j)=-Xm(iNx2-i+1,j)
            Ym(iNx2+i,j)=Ym(iNx2-i+1,j)
            Te(iNx2+i,j)=Te(iNx2-i+1,j)
         END DO
      END DO
 8    continue
      RETURN
      END
```

B.2 Calculation of maximum Lyapunov exponent

The maximum Lyapunov exponent is obtained by determining the Lagrange multiplier needed to readjust the phase-space distance between a reference trajectory and a perturbed one.

```
Cubic Spline on staggered grid for
Continuum mechanics are used on a rectangular grid to
Calculate the Rayleigh-Be'nard flow patterns.
Conservative form of differential equations are used.
c$SAVE_LOCALS ON
c$SYMDEBUG
c$WARNINGS ON
c$OPTION RANGE ON
      PROGRAM rbcufd
      INCLUDE 'rbculy.h'
      CHARACTER*50 LY_FILE     /'Rax5/cufd500M1c.ly'/
      CHARACTER*50 STATE_FILE  /'Rax5/cufd500M1c.state'/
C                        f77 -O -r8 -o xly500 rbly.f

      Ra = 500 000.0d0
      dr=0.1d0
      Npsd=4*Nx*(Ny-1)
      factor=dr*Sqrt(1.0d0/(Npsd))
      IX=0
      IY=0
      cn=0.9d0
      delta=1.0d0
      rKb= 1.0d0
      bulk=0.0d0
      Th = 1.5d0
      Tl = 0.5d0
      rLy= Ny*delta
      eta=rLy/Sqrt(Ra)
      rkappa=eta
      g= -rKb*(Th-Tl)/rLy
      dt=delta*cn/Sqrt(2.0*Th)
      tottime=20000.0d0
      maxiter=INT(tottime/dt)
      iprfreq=maxiter/5
c$IF(THREE_D_STOKES_FLUID)
      bulk=eta/3.0d0
c$ENDIF
C
C     initialize variables
      CALL INITIAL(STATE_FILE)
CCCCCC
CCCCCC
C     START ITERATION
      sum=0.0d0
      time=0.0d0
      DO iter=1,INT(maxiter)
c     follow first trajectory
         CALL SUBS(rho1,u1,v1,e1,rho,u,v,e)
         CALL CONSV()
         CALL RK(dt)
         CALL SUBS(rho,u,v,e,rho1,u1,v1,e1)
c     follow second trajectory
         CALL SUBS(rho1,u1,v1,e1,rho,u,v,e)
         CALL CONSV()
         CALL RK(dt)
         CALL SUBS(rho,u,v,e,rho1,u1,v1,e1)
c     calculate distance
         CALL DIST(dis)
         sum=sum+Log(dis/dr)
c     rescale distance
         CALL RESC(dis)
         time=time+dt
         if (jmod(iter,100).eq.0) then
            exp=(sum)/time
            open(14,file=LY_FILE,form='formatted',ACCESS='APPEND',
     2        STATUS= 'UNKNOWN')
            write(14,'(f10.3,3x,f18.14)')  time,exp
            close(14)
         end if
      END DO
CCCCCCCCCCCCCCCCCCCC SECOND HALF CCCCCCCCCCCCCCCCCCCCC
c      sum=0.0d0
c      time=0.0d0
c      DO iter=1,INT(maxiter/2)
cc     follow first trajectory
c         CALL SUBS(rho1,u1,v1,e1,rho,u,v,e)
c         CALL CONSV()
c         CALL RK(dt)
c         CALL SUBS(rho,u,v,e,rho1,u1,v1,e1)
cc     follow second trajectory
c         CALL SUBS(rho1,u1,v1,e1,rho,u,v,e)
c         CALL CONSV()
c         CALL RK(dt)
c         CALL SUBS(rho,u,v,e,rho1,u1,v1,e1)
cc     calculate distance
c         CALL DIST(dis)
c         sum=sum+Log(dis/dr)
cc     rescale distance
c         CALL RESC(dis)
c         time=time+dt
c         if (jmod(iter,100).eq.0) then
c            exp=(sum)/time
c            open(14,file=LY2_FILE,form='formatted',ACCESS='APPEND',
c     2        STATUS= 'UNKNOWN')
c            write(14,'(f10.3,3x,f18.14)')  time,exp
c            close(14)
c         end if
c      END DO


      STOP
      END
C
C-----------------------------------------------------------------------X
C     Sets initial and boundary values
      SUBROUTINE INITIAL(STATE_FILE)
      INCLUDE 'rbculy.h'
      CHARACTER*50 STATE_FILE
C     initialize grid point values (note period bc in x-direction)
      open(22,file=STATE_FILE,form='formatted',status='OLD')
```

```
      DO i=1,Nx
         DO j=1,Ny
            read (22,'(2i5,4f24.18)',END=99)ii,jj,rho(i,j),
     2           u(i,j),v(i,j),e(i,j)
            if(i.ne.ii) write(*,*) "error"
            if(j.ne.jj) write(*,*) "error"
         END DO
         j=Ny+1
         read (22,'(2i5,4f24.18)',END=99)ii,jj,
     2        u(i,j),v(i,j),e(i,j)
      END DO
 99   close(22,STATUS='KEEP')
C     enforce boundary conditions
      DO i=1,Nx
         e(i,1)=Th
         u(i,1)=0.0d0
         v(i,1)=0.0d0
         u(i,Ny+1)=0.0d0
         v(i,Ny+1)=0.0d0
         e(i,Ny+1)=Tl
      END DO
      CALL SUBS(rho,u,v,e,rho1,u1,v1,e1)

      DO i=1,Nx
         DO j=1,Ny
            rho2(i,j)=rho1(i,j)+factor
         END DO
      END DO
      DO i=1,Nx
         DO j=1,Nn
            u2(i,j)=u1(i,j)+factor
            v2(i,j)=v1(i,j)+factor
            e2(i,j)=e1(i,j)+factor
         END DO
      END DO
      RETURN
      END

      SUBROUTINE DIST(dis)
      INCLUDE 'rbculy.h'
      dis=0.0d0
      DO i=1,Nx
         DO j=1,Ny
            dis=dis+(rho2(i,j)-rho1(i,j))**2.0d0
         END DO
      END DO
      DO i=1,Nx
         DO j=1,Nn
            dis=dis+(u2(i,j)-u1(i,j))**2.0d0
            dis=dis+(v2(i,j)-v1(i,j))**2.0d0
            dis=dis+(e2(i,j)-e1(i,j))**2.0d0
         END DO
      END DO
      dis=Sqrt(dis)
      RETURN
      END

      SUBROUTINE RESC(dis)
      INCLUDE 'rbculy.h'
      DO i=1,Nx
         DO j=1,Ny
            rho2(i,j)=rho1(i,j)+(rho2(i,j)-rho1(i,j))*dr/dis
         END DO
      END DO
      DO i=1,Nx
         DO j=1,Nn
            u2(i,j)=u1(i,j)+(u2(i,j)-u1(i,j))*dr/dis
            v2(i,j)=v1(i,j)+(v2(i,j)-v1(i,j))*dr/dis
            e2(i,j)=e1(i,j)+(e2(i,j)-e1(i,j))*dr/dis
         END DO
      END DO
      RETURN
      END


      SUBROUTINE SUBS(rhoa,ua,va,ea,rhob,ub,vb,eb)
      INCLUDE 'rbculy.h'
      DIMENSION Rhoa(Nx,Nn),Ua(Nx,Nn),Va(Nx,Nn),Ea(Nx,Nn)
      DIMENSION Rhob(Nx,Nn),Ub(Nx,Nn),Vb(Nx,Nn),Eb(Nx,Nn)
      DO i=1,Nx
         DO j=1,Ny
            rhob(i,j)=rhoa(i,j)
         END DO
      END DO
      DO i=1,Nx
         DO j=1,Nn
            ub(i,j)=ua(i,j)
            vb(i,j)=va(i,j)
            eb(i,j)=ea(i,j)
         END DO
      END DO
      RETURN
      END


      SUBROUTINE CONSV()
      INCLUDE 'rbculy.h'
      DIMENSION f1(Nx), f2(Ny+1), f1b(Nx), f2b(Ny+1)
C     interpolate density value to all grids
C     interpolate new density to other grids
C     rho -> G4rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=rho(i,j)
         END DO
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4rho(i,j)=f2b(j-1)
         END DO
      END DO
C     rho -> G3rho
      DO j=1,Ny
         DO i=1,Nx
            f1(i)=rho(i,j)
         END DO
         CALL CubePbINT(f1,f1b,delta,Nx)
         G3rho(1,j)=f1b(Nx)
         DO i=2,Nx
            G3rho(i,j)=f1b(i-1)
         END DO
      END DO
C     G3rho -> G1rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=G3rho(i,j)
         END DO
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G1rho(i,j)=f2b(j-1)
         END DO
      END DO
C
C     initialize "conserved" variables on grid
      DO j=1,Ny+1
         DO i=1,Nx
            Xm(i,j) = G1rho(i,j)*u(i,j)
            Ym(i,j) = G1rho(i,j)*v(i,j)
            Te(i,j) = G1rho(i,j)*(e(i,j)
     &            +(u(i,j)*u(i,j)+v(i,j)*v(i,j))/2.0d0)
         END DO
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      SUBROUTINE RHS(rhonew,Xmnew,Ymnew,Tenew,iNx)
      INCLUDE 'rbculy.h'
      DIMENSION rhonew(Nx,Ny+1),Xmnew(Nx,Ny+1)
      DIMENSION Ymnew(Nx,Ny+1),Tenew(Nx,Ny+1)
      DIMENSION f1(Nx), f2(Ny+1), f1b(Nx),f2b(Ny+1), fx(Nx), fy(Ny+1)
C     interpolate new density to other grids
C     rhonew -> G4rho
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=rhonew(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4rho(i,j)=f2b(j-1)
         END DO
      END DO
C     rhonew -> G3rho
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=rhonew(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3rho(1,j)=f1b(iNx)
         DO i=2,iNx
            G3rho(i,j)=f1b(i-1)
         END DO
      END DO
C     G3rho -> G1rho
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G3rho(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G1rho(i,j)=f2b(j-1)
         END DO
      END DO
C     determine state variables u,v,e from the "new" conserved ones
      DO j=2,Ny
         DO i = 1,iNx
            u(i,j) = Xmnew(i,j)/G1rho(i,j)
            v(i,j) = Ymnew(i,j)/G1rho(i,j)
            e(i,j) = Tenew(i,j)/G1rho(i,j)
     &          -(u(i,j)*u(i,j)+v(i,j)*v(i,j))/2.0d0
         END DO
      END DO
C     interpolate state variables
C     u -> G4u
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=u(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4u(i,j)=f1b(i)
         END DO
      END DO
C     v -> G4v
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=v(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4v(i,j)=f1b(i)
         END DO
      END DO
C     e -> G4e
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=e(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G4e(i,j)=f1b(i)
         END DO
      END DO
C     u -> G3u
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=u(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3u(i,j)=f2b(j)
         END DO
      END DO
```

```
C     e -> G3e   CUSP
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=e(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3e(i,j)=f2b(j)
         END DO
      END DO
C     v -> G3v
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=v(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G3v(i,j)=f2b(j)
         END DO
      END DO
C     determine gradients of state variables
C     u->ux (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=u(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            ux(i,j)=fx(i)
         END DO
      END DO
C     v -> vx (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=v(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            vx(i,j)=fx(i)
         END DO
      END DO
C     e -> ex (periodic 1->2)
      DO j=1,Ny+1
         DO i=1,iNx
            f1(i)=e(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            ex(i,j)=fx(i)
         END DO
      END DO
C     u -> uy (even 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=u(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,2)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            uy(i,j)=fy(j)
         END DO
      END DO
C     e -> ey (even 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=e(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,2)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            ey(i,j)=fy(j)
         END DO
      END DO
C     v -> vy (odd 1->2)
      DO i=1,iNx
         DO j=1,Ny+1
            f2(j)=v(i,j)
         END DO
C         CALL CubeFbDxV1(f2,fy,delta,Ny,1)
         CALL CubeFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            vy(i,j)=fy(j)
         END DO
      END DO
C     interpolate velocity gradients to opposite grid
C     ux->G2ux
      DO i=1,iNx
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=ux(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G2ux(i,j)=f2b(j)
         END DO
      END DO
C     vx->G2vx
      DO i=1,iNx
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=vx(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL PADEFbINTV(f2,f2b,Ny)
         CALL CubeFbINT(f2,f2b,delta,Ny)
         DO j=1,Ny
            G2vx(i,j)=f2b(j)
         END DO
      END DO
C     uy -> G2uy
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=uy(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G2uy(i,j)=f1b(i)
         END DO
      END DO
C     vy -> G2vy
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=vy(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         DO i=1,iNx
            G2vy(i,j)=f1b(i)
         END DO
      END DO
C     G2ux->G3ux
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=G2ux(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3ux(1,j)=f1b(iNx)
         DO i=2,iNx
            G3ux(i,j)=f1b(i-1)
         END DO
      END DO
C     G2vx->G3vx
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=G2vx(i,j)
         END DO
C         CALL PadePbINTV(f1,f1b,iNx)
         CALL CubePbINT(f1,f1b,delta,iNx)
         G3vx(1,j)=f1b(iNx)
         DO i=2,iNx
            G3vx(i,j)=f1b(i-1)
         END DO
      END DO
C     G2uy->G4uy
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G2uy(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4uy(i,j)=f2b(j-1)
         END DO
      END DO
C     G2vy->G4vy
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=G2vy(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4vy(i,j)=f2b(j-1)
         END DO
      END DO
C
C     determine FLUX values
      DO i=1,iNx
         Grho(i,1)=0.0d0
         Grho(i,Ny+1)=0.0d0
         DO j=2,Ny
            qx=-rkappa*ex(i,j)
            Sxx=(bulk+eta)*ux(i,j)+(bulk-eta)*G4vy(i,j)
     1          -G4rho(i,j)*G4e(i,j)
            Sxy=eta*(vx(i,j)+G4uy(i,j))
            G4Te=G4e(i,j)+(G4u(i,j)*G4u(i,j)+G4v(i,j)*G4v(i,j))/2.0d0
            Grho(i,j)=G4rho(i,j)*G4v(i,j)
            Fxm(i,j)=G4rho(i,j)*G4u(i,j)*G4u(i,j)-Sxx
            Fym(i,j)=G4rho(i,j)*G4u(i,j)*G4v(i,j)-Sxy
            Fte(i,j)=G4rho(i,j)*G4u(i,j)*G4Te +
     1          qx - G4u(i,j)*Sxx - G4v(i,j)*Sxy
         END DO
      END DO
      DO i=1,iNx
         DO j=1,Ny
            qy=-rkappa*ey(i,j)
            Syy=(bulk+eta)*vy(i,j)+(bulk-eta)*G3ux(i,j)
     1          -G3rho(i,j)*G3e(i,j)
            Sxy=eta*(G3vx(i,j)+uy(i,j))
            G3Te=G3e(i,j)+(G3u(i,j)*G3u(i,j)+G3v(i,j)*G3v(i,j))/2.0d0
            Frho(i,j)=G3rho(i,j)*G3u(i,j)
            Gxm(i,j)=G3rho(i,j)*G3u(i,j)*G3v(i,j)-Sxy
            Gym(i,j)=G3rho(i,j)*G3v(i,j)*G3v(i,j)-Syy
            Gte(i,j)=G3rho(i,j)*G3v(i,j)*G3Te +
     1          qy - G3u(i,j)*Sxy - G3v(i,j)*Syy
         END DO
      END DO
C     Compute Sources (rho has no source)
      DO i=1,iNx
         DO j=1,Ny
            rhodot(i,j)=0.0d0
            Xmdot(i,j)=0.0d0
            Ymdot(i,j)=G1rho(i,j)*g
            Tedot(i,j)=G1rho(i,j)*v(i,j)*g
         END DO
      END DO
C     Compute divergence of the fluxes
C     Frho->rhodot  (periodic 1->2)
      DO j=1,Ny
         DO i=1,iNx
            f1(i)=Frho(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
C         CALL CDPbDx(f1,fx,delta,iNx)
         DO i=1,iNx
            RhoDot(i,j)=RhoDot(i,j)-fx(i)
         END DO
      END DO
C     Grho->rhodot (odd 1->2)
      DO i=1,iNx
```

```
         f2(1)=0.0d0
         DO j=2,Ny
            f2(j)=Grho(i,j)
         END DO
         f2(Ny+1)=0.0d0
C         CALL CubeFbDxV1(f2,fy,delta,Ny,1)
         CALL CubeFbDx(f2,fy,delta,Ny)
C         CALL CDFbDx(f2,fy,delta,Ny)
         DO j=1,Ny
            RhoDot(i,j)=RhoDot(i,j)-fy(j)
         END DO
      END DO
C     Fxm-> Xmdot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fxm(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
C         CALL CDPbDx(f1,fx,delta,iNx)
         XmDot(1,j)=XmDot(1,j)-fx(iNx)
         DO i=2,iNx
            XmDot(i,j)=XmDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gxm-> Xmdot  (odd 2->1)
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=Gxm(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,1)
         CALL CubeFbDx(f2,fy,delta,Ny-1)
C         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            XmDot(i,j)=XmDot(i,j)-fy(j-1)
         END DO
      END DO
C     Fym-> Ymdot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fym(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
C         CALL CDPbDx(f1,fx,delta,iNx)
         YmDot(1,j)=YmDot(1,j)-fx(iNx)
         DO i=2,iNx
            YmDot(i,j)=YmDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gym-> Ymdot  (even 2->1)
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=Gym(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,2)
         CALL CubeFbDx(f2,fy,delta,Ny-1)
C         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            YmDot(i,j)=YmDot(i,j)-fy(j-1)
         END DO
      END DO
C     Fte-> Tedot  (periodic 2->1)
      DO j=2,Ny
         DO i=1,iNx
            f1(i)=Fte(i,j)
         END DO
C         CALL CubePbDxV(f1,fx,delta,iNx)
         CALL CubePbDx(f1,fx,delta,iNx)
C         CALL CDPbDx(f1,fx,delta,iNx)
         TeDot(1,j)=TeDot(1,j)-fx(iNx)
         DO i=2,iNx
            TeDot(i,j)=TeDot(i,j)-fx(i-1)
         END DO
      END DO
C     Gte-> Tedot  (odd 2->1)
      DO i=1,iNx
         DO j=1,Ny
            f2(j)=Gte(i,j)
         END DO
C         CALL CubeFbDxV2(f2,fy,delta,Ny,1)
         CALL CubeFbDx(f2,fy,delta,Ny-1)
C         CALL CDFbDx(f2,fy,delta,Ny-1)
         DO j=2,Ny
            TeDot(i,j)=TeDot(i,j)-fy(j-1)
         END DO
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      SUBROUTINE RK(dt)
      INCLUDE 'rbculy.h'
      DIMENSION rhonew(Nx,Ny), Xmnew(Nx,Nn), Ymnew(Nx,Nn), Tenew(Nx,Nn)
      DIMENSION rhodot1(Nx,Ny),Xmdot1(Nx,Ny),Ymdot1(Nx,Ny),Tedot1(Nx,Ny)
      DIMENSION rhodot2(Nx,Ny),Xmdot2(Nx,Ny),Ymdot2(Nx,Ny),Tedot2(Nx,Ny)
      DIMENSION rhodot3(Nx,Ny),Xmdot3(Nx,Ny),Ymdot3(Nx,Ny),Tedot3(Nx,Ny)
      DIMENSION rhodot4(Nx,Ny),Xmdot4(Nx,Ny),Ymdot4(Nx,Ny),Tedot4(Nx,Ny)
      DO i=1,Nx
         DO j=1,Ny
            rhonew(i,j)=rho(i,j)
            Xmnew(i,j) = Xm(i,j)
            Ymnew(i,j) = Ym(i,j)
            Tenew(i,j) = Te(i,j)
         END DO
         Xmnew(i,Ny+1) = Xm(i,Ny+1)
         Ymnew(i,Ny+1) = Ym(i,Ny+1)
         Tenew(i,Ny+1) = Te(i,Ny+1)
      END DO
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot1(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot1(i,j)=rhodot(i,j)
            Xmdot1 (i,j) =Xmdot(i,j)
            Ymdot1 (i,j) =Ymdot(i,j)
            Tedot1 (i,j) =Tedot(i,j)
         END DO
      END DO
 1    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot1(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot1(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j)+ dt*Xmdot1(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j)+ dt*Ymdot1(i,j)/2.0d0
            Tenew(i,j) = Te(i,j)+ dt*Tedot1(i,j)/2.0d0
         END DO
      END DO
 2    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot2(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot2(i,j)=rhodot(i,j)
            Xmdot2(i,j) = Xmdot(i,j)
            Ymdot2(i,j) = Ymdot(i,j)
            Tedot2(i,j) = Tedot(i,j)
         END DO
      END DO
 3    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot2(i,1)/2.0d0
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot2(i,j)/2.0d0
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot2(i,j)/2.0d0
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot2(i,j)/2.0d0
            Tenew(i,j) = Te(i,j) +dt*Tedot2(i,j)/2.0d0
         END DO
      END DO
 4    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot3(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot3(i,j)=rhodot(i,j)
            Xmdot3(i,j) = Xmdot(i,j)
            Ymdot3(i,j) = Ymdot(i,j)
            Tedot3(i,j) = Tedot(i,j)
         END DO
      END DO
 5    continue
      DO i=1,Nx
         rhonew(i,1)=rho(i,1)+dt*rhodot3(i,1)
         DO j=2,Ny
            rhonew(i,j)=rho(i,j)+dt*rhodot3(i,j)
            Xmnew(i,j) = Xm(i,j) +dt*Xmdot3(i,j)
            Ymnew(i,j) = Ym(i,j) +dt*Ymdot3(i,j)
            Tenew(i,j) = Te(i,j) +dt*Tedot3(i,j)
         END DO
      END DO
 6    continue
      CALL RHS(rhonew,Xmnew,Ymnew,Tenew,Nx)
C
      DO i=1,Nx
         rhodot4(i,1)=rhodot(i,1)
         DO j=2,Ny
            rhodot4(i,j)=rhodot(i,j)
            Xmdot4(i,j) = Xmdot(i,j)
            Ymdot4(i,j) = Ymdot(i,j)
            Tedot4(i,j) = Tedot(i,j)
         END DO
      END DO
 7    continue
      DO i=1,Nx
         rho(i,1)=rho(i,1)+dt*(rhodot1(i,1)+
     &        2.0d0*(rhodot2(i,1)+rhodot3(i,1))+rhodot4(i,1))/6.0d0
         DO j=2,Ny
            rho(i,j)=rho(i,j)+dt*(rhodot1(i,j)+
     &           2.0d0*(rhodot2(i,j)+rhodot3(i,j))+rhodot4(i,j))/6.0d0
            Xm(i,j)=Xm(i,j)+dt*(Xmdot1(i,j)+
     &           2.0d0*(Xmdot2(i,j)+Xmdot3(i,j))+Xmdot4(i,j))/6.0d0
            Ym(i,j)=Ym(i,j)+dt*(Ymdot1(i,j)+
     &           2.0d0*(Ymdot2(i,j)+Ymdot3(i,j))+Ymdot4(i,j))/6.0d0
            Te(i,j)=Te(i,j)+dt*(Tedot1(i,j)+
     &           2.0d0*(Tedot2(i,j)+Tedot3(i,j))+Tedot4(i,j))/6.0d0
         END DO
      END DO
 8    continue
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine TRIDIA(a,b,c,r,u,n)
C     subroutine tridia solves system Au=r where A is tridiagonal
C     with major and minor diags c and a resp. and main diag b.
C     b is diagonal, a/c are lower/upper off-diagonals
      IMPLICIT REAL*8 (A-H,O-Z)
      parameter (nmax=1000)
      dimension gam(nmax),a(n),b(n),c(n),r(n),u(n)
      if(b(1).eq.0.0d0) pause
      bet=b(1)
      u(1) = r(1)/bet
      do 11 j=2,n
         gam(j)=c(j-1)/bet
         bet=b(j)-a(j)*gam(j)
         if(bet.eq.0.0d0) then
            write(*,*) j,a(j),b(j),c(j)
            pause
         end if
         u(j)=(r(j)-a(j)*u(j-1))/bet
 11   continue
      do 12 j=n-1,1,-1
         u(j)=u(j)-gam(j+1)*u(j+1)
 12   continue
      return
      end
C
C-----------------------------------------------------------------------X
C*****Random Number Generator
      FUNCTION RANDOM(IX,IY)
      IMPLICIT REAL*8 (A-H,O-Z)
      IMPLICIT INTEGER (I-N)
      I = 1029*IX+1731
      J = I + 1029*IY + 507*IX - 1731
```

```
      IX = MOD(I,2048)
      J = J + (I-IX)/2048
      IY = MOD(J,2048)
      RANDOM = (IX+2048*IY)/4194304.0d0
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CubePbDX(f,fp,h,n)
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
C     ASSUMES M=0 at endpoints
C     USES DBTDLU and DBTDSOL
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fp(n)
      dimension am(n,n), bm(3,n-1), rm(1,n-1), cm(n,1), xm(n)
      data ldb/3/, ldr/1/, ml/1/, mu/1/, mp/1/
      ldc=n
c     Set elements of matrix A in normal (full) form
      do 20 j = 1,n
         do 10 i = 1,n
            if (j-i .le. mu .and. i-j .le. ml) then
               am(i,j) = 1.0d0
            else
               am(i,j) = 0.0d0
            endif
 10      continue
         am(j,j) = 4.0d0
 20   continue
      am(1,n)=1.0d0
      am(n,1)=1.0d0
c     Load b, r, c from A
      m = ml + mu + 1
      nmp = n - mp
      do 50 j = 1,nmp
         do 30 k = 1,m
            i = k + j - mu - 1
 30         if (i .ge. 1 .and. i .le. nmp) bm(k,j) = am(i,j)
            do 40 k = 1,mp
 40            rm(k,j) = am(k+nmp,j)
 50         continue
            do 70 k = 1,mp
               do 60 i = 1,n
 60               cm(i,k) = am(i,k+nmp)
 70   continue
c
C     SETUP RHS:
      xm(1)=(6.0d0/h/h)*(f(2)-2.0d0*f(1)+f(N))
      DO i=2,N-1
         xm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      xm(N)=(6.0d0/h/h)*(f(1)-2.0d0*f(N)+f(N-1))
c
c Call solver routines and print results
      call dbbfa (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, ier)
      if (ier .ne. 0) then
        write(6,130) ier
 130    format(//'Error return from dbbFA.  ier =',i4)
        stop
      endif
      call dbbsl (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, xm)
      DO i=1,n-1
         fp(i)=(f(i+1)-f(i))/h - (h/24.0d0)*(xm(i+1)-xm(i))
      END DO
      fp(n)=(f(1)-f(n))/h - (h/24.0d0)*(xm(1)-xm(n))
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CDPbDX(f,fp,h,n)
C     CENTERED DIFFERENCE DIVERGENCE FOR PERIODIC BOUNDARIES
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fp(n)
      DO i=1,n-1
         fp(i)=(f(i+1)-f(i))/h
      END DO
      fp(n)=(f(1)-f(n))/h
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CubePbINT(f,fbar,h,n)
C     subroutine that determines derivatives with periodic b.c's
C     assumes function is on grid and derivative is on cell center
C     otherwise the index must be adjusted
C     ASSUMES M=0 at endpoints
C     USES DBTDLU and DBTDSOL
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n),fbar(n)
      dimension am(n,n), bm(3,n-1), rm(1,n-1), cm(n,1), xm(n)
      data ldb/3/, ldr/1/, ml/1/, mu/1/, mp/1/
      ldc=n
c     Set elements of matrix A in normal (full) form
      do 20 j = 1,n
         do 10 i = 1,n
            if (j-i .le. mu .and. i-j .le. ml) then
               am(i,j) = 1.0d0
            else
               am(i,j) = 0.0d0
            endif
 10      continue
         am(j,j) = 4.0d0
 20   continue
      am(1,n)=1.0d0
      am(n,1)=1.0d0
c     Load b, r, c from A
      m = ml + mu + 1
      nmp = n - mp
      do 50 j = 1,nmp
         do 30 k = 1,m
            i = k + j - mu - 1
 30         if (i .ge. 1 .and. i .le. nmp) bm(k,j) = am(i,j)
            do 40 k = 1,mp
 40            rm(k,j) = am(k+nmp,j)
 50         continue
            do 70 k = 1,mp
               do 60 i = 1,n
 60               cm(i,k) = am(i,k+nmp)
 70   continue
c
C     SETUP RHS:
      xm(1)=(6.0d0/h/h)*(f(2)-2.0d0*f(1)+f(N))
      DO i=2,N-1
         xm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      xm(N)=(6.0d0/h/h)*(f(1)-2.0d0*f(N)+f(N-1))
c
c Call solver routines and print results
      call dbbfa (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, ier)
      if (ier .ne. 0) then
        write(6,130) ier
 130    format(//'Error return from dbbFA.  ier =',i4)
        stop
      endif
      call dbbsl (n, ml, mu, mp, bm, rm, cm, ldb, ldr, ldc, xm)
      DO i=1,n-1
         fbar(i)=(f(i+1)+f(i))/2.0d0 - (h*h/16.0d0)*(xm(i+1)+xm(i))
      END DO
      fbar(n)=(f(1)+f(n))/2.0d0 - (h*h/16.0d0)*(xm(1)+xm(n))
      RETURN
      end
C
C-----------------------------------------------------------------------X
C
      subroutine CubeFbDx(f,fy,h,n)
C     differentiates with fixed b.c's
C     ip=1 -> odd function, ip=2 -> even function
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fy(n), rMom(n+1)
      dimension am(n+1),bm(n+1),cm(n+1),rm(n+1)
C     SETUP RHS:
      rm(1)=0.0d0
      DO i=2,N
         rm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      rm(N+1)=0.0d0
C     SETUP LHS:
      am(1)=0.0d0
      bm(1)=4.0d0
      cm(1)=0.0d0
      DO i=2,N
         am(i)=1.0d0
         bm(i)=4.0d0
         cm(i)=1.0d0
      END DO
      am(N+1)=0.0d0
      bm(N+1)=4.0d0
      cm(N+1)=0.0d0
C     SOLVE SYSTEM:
      call tridia(am,bm,cm,rm,rMom,N+1)
      DO i=1,N
         fy(i)=(f(i+1)-f(i))/h - (h/24.0d0)*(rMom(i+1)-rMom(i))
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CDFbDx(f,fy,h,n)
C     CENTERED DIFFERENCE DIVERGENCE FOR FIXED BOUNDARIES
C     differentiates with fixed b.c's
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fy(n)
      DO i=1,N
         fy(i)=(f(i+1)-f(i))/h
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C
      subroutine CubeFbInt(f,fbar,h,n)
C     interpolation with fixed b.c's
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension f(n+1),fbar(n), rMom(n+1)
      dimension am(n+1),bm(n+1),cm(n+1),rm(n+1)
C     SETUP RHS:
      rm(1)=0.0d0
      DO i=2,N
         rm(i)=(6.0d0/h/h)*(f(i+1)-2.0d0*f(i)+f(i-1))
      END DO
      rm(N+1)=0.0d0
C     SETUP LHS:
      am(1)=0.0d0
      bm(1)=4.0d0
      cm(1)=0.0d0
      DO i=2,N
         am(i)=1.0d0
         bm(i)=4.0d0
         cm(i)=1.0d0
      END DO
      am(N+1)=0.0d0
      bm(N+1)=4.0d0
      cm(N+1)=0.0d0
c$IF(FD)
c      DO i=1,N
c         fbar(i)=(f(i+1)+f(i))/2.0d0
c      END DO
c$ENDIF
c$IF(CUBIC)
C     SOLVE SYSTEM:
      call tridia(am,bm,cm,rm,rMom,N+1)
      DO i=1,N
         fbar(i)=(f(i+1)+f(i))/2.0d0 - (h*h/16.0d0)*(rMom(i+1)+rMom(i))
      END DO
c$ENDIF
      RETURN
      END
C
C-----------------------------------------------------------------------
C
      SUBROUTINE DBBFA (N, ML, MU, MP, B, R, C, LDB, LDR, LDC, IER)
      DOUBLE PRECISION B(LDB,*), R(LDR,*), C(LDC,*)
```

```
C-----------------------------------------------------------------------
C LU factorization of border-banded matrix without pivoting.
C Double precision version.
C Call DBBFA to factor a given matrix A, then call DBBSL to solve
C a system Ax = b with given vector b.
C
C Input arguments:
C
C N    = order of matrix
C ML   = lower half-bandwidth (.LT. N)
C MU   = upper half-bandwidth (.LT. N)
C MP   = width of border at right and lower edges  (.LE. N).
C B    = LDB by N-MP array with band elements (LDB .ge. ML+MU+1).
C        Element a(i,j) is to be stored in B(I-J+MU+1,J) for
C          1 .LE. I,J .LE. N-MP  and  -ML .LE. J-I .LE. MU.
C R    = LDR by N-MP array with lower edge elements (LDR .GE. MP).
C        a(i,j) is to be stored in R(I-N+MP,J) for
C          N-MP+1 .LE. I .LE. N  and  1 .LE. J .LE. N-MP.
C C    = LDC by MP array with right edge elements (LDC .GE. N).
C        a(i,j) is to be stored in C(I,J-N+MP) for
C          1 .LE. I .LE. N  and  N-MP+1 .LE. J .LE. N.
C LDB  = declared first dimension of B, must be .GE. ML+MU+1.
C LDR  = declared first dimension of R, must be .GE. MP.
C LDC  = declared first dimension of C, must be .GE. N.
C
C
C Output arguments:
C
C IER  = error flag.
C        IER = -1 means integer input parameters were illegal.
C        IER = 0 means the LU factorization was successful.
C        IER = K .GT. 1 means the K-th pivot was found to be zero.
C B, R, C arrays contain a representation of the L and U factors.
C
C Routines called: DSCAL, DAXPY (from the BLAS collection)
C-----------------------------------------------------------------------
      DOUBLE PRECISION ONE, T, ZERO
      DATA ONE/1.0D0/, ZERO/0.0D0/
C
C Test inputs.
      IF (N .LE. 0) GO TO 500
      IF (ML .GE. N .OR. MU .GE. N .OR. MP .GT. N) GO TO 500
      IF (LDB .LT. ML+MU+1 .OR. LDR .LT. MP .OR. LDC .LT. N) GO TO 500
C
C Set auxiliary constants.
      NM1 = N - 1
      NMP = N - MP
      NMPP1 = NMP + 1
      MU1 = MU + 1
      MU2 = MU + 2
      IER = 0
C
C Generate LU factors of band and lower border (stages 1 .. N-MP).
      DO 160 K = 1,NMP
        KP1 = K + 1
C Form K-th pivot.
        T = B(MU1,K)
        IF (T .EQ. ZERO) GO TO 550
        T = ONE/T
        B(MU1,K) = T
C Form L elements in band.
         LEN1 = MIN(NMP-K,ML)
         T = -T
         CALL DSCAL (LEN1, T, B(MU2,K), 1)
C Form L elements in lower border.
         CALL DSCAL (MP, T, R(1,K), 1)
C Update matrix elements in band and lower border.
        JHI = MIN(NMP,K+MU)
        DO 150 J = KP1,JHI
          T = B(K-J+MU1,J)
          CALL DAXPY (LEN1, T, B(MU2,K), 1, B(KP1-J+MU1,J), 1)
          CALL DAXPY (MP, T, R(1,K), 1, R(1,J), 1)
 150      CONTINUE
 160    CONTINUE
C
C Update matrix elements in right border from stages 1 .. N-MP.
      DO 240 K = 1,NMP
        KP1 = K + 1
        DO 230 J = NMPP1,N
          T = C(K,J-NMP)
          LEN1 = MIN(NMP-K,ML)
          CALL DAXPY (LEN1, T, B(MU2,K), 1, C(KP1,J-NMP), 1)
          CALL DAXPY (MP, T, R(1,K), 1, C(NMPP1,J-NMP), 1)
 230      CONTINUE
 240    CONTINUE
C
C Generate LU factors in lower right block (stages N-MP+1 .. N-1).
C
      DO 340 K = NMPP1,NM1
        KP1 = K + 1
C Form K-th pivot.
        T = C(K,K-NMP)
        IF (T .EQ. ZERO) GO TO 550
        T = ONE/T
        C(K,K-NMP) = T
C Form L elements.
        T = -T
        CALL DSCAL (N-K, T, C(KP1,K-NMP), 1)
C Update matrix elements.
        DO 330 J = KP1,N
          T = C(K,J-NMP)
          CALL DAXPY (N-K, T, C(KP1,K-NMP), 1, C(KP1,J-NMP), 1)
 330      CONTINUE
 340    CONTINUE
C
C Form last pivot and return
      IF (MP .EQ. 0) RETURN
      K = N
      T = C(N,MP)
      IF (T .EQ. ZERO) GO TO 550
      C(N,MP) = ONE/T
      RETURN
C
C Error return on illegal input.
 500  IER = -1
      RETURN
C Error return on zero pivot.
 550  IER = K
      RETURN
      END
C
      SUBROUTINE DBBSL (N, ML, MU, MP, B, R, C, LDB, LDR, LDC, X)
      DOUBLE PRECISION B(LDB,*), R(LDR,*), C(LDC,*), X(*)
C-----------------------------------------------------------------------
C Solution of border-banded matrix without pivoting, given the LU
C factorization by DBBFA.  Assumes factorization was successful and
C factor information has not been changed.  For systems Ax = b,
C with A factored by DBBFA, call DBBSL once for each RHS vector b.
C
C Input arguments:
C
C N    = order of matrix
C ML   = lower half-bandwidth
C MU   = upper half-bandwidth
C MP   = width of border at right and lower edges
C B, R, C = arrays as output by factorization
C LDB, LDR, LDC  = declared first dimension of B, R, C, respectively
C          (assumed to be the same as in factorization)
C X    = array of length .GE. N containing right-hand side vector b
C
C
C Output arguments:
C
C X    = solution vector
C
C Routines called: DSCAL, DAXPY (from the BLAS collection)
C-----------------------------------------------------------------------
      DOUBLE PRECISION T
C Set auxiliary constants.
      NM1 = N - 1
      NMP = N - MP
      NMPP1 = NMP + 1
      MU1 = MU + 1
      MU2 = MU + 2
C
C Apply inverse of L (columns 1 .. N-MP).
      DO 130 K = 1,NMP
        KP1 = K + 1
        LEN1 = MIN(NMP-K,ML)
        CALL DAXPY (LEN1, X(K), B(MU2,K), 1, X(KP1), 1)
        CALL DAXPY (MP, X(K), R(1,K), 1, X(NMPP1), 1)
 130    CONTINUE
C
C Apply inverse of L (columns N-MP+1 .. N-1).
      DO 150 K = NMPP1,NM1
        KP1 = K + 1
        CALL DAXPY (N-K, X(K), C(KP1,K-NMP), 1, X(KP1), 1)
 150    CONTINUE
C
C Apply inverse of U (columns N .. N-MP+1).
      DO 220 K = N,NMPP1,-1
        X(K) = X(K)*C(K,K-NMP)
        T = -X(K)
        CALL DAXPY (K-1, T, C(1,K-NMP), 1, X(1), 1)
 220    CONTINUE
C
C Apply inverse of U (columns N-MP .. 1).
      DO 240 K = NMP,1,-1
        X(K) = X(K)*B(MU1,K)
        ILO = MAX(1,K-MU)
        IBLO = ILO - K + MU1
        T = -X(K)
        CALL DAXPY (K-ILO, T, B(IBLO,K), 1, X(ILO), 1)
 240    CONTINUE
C
      RETURN
      END
C
      subroutine daxpy(n,da,dx,incx,dy,incy)
c
c     constant times a vector plus a vector.
c     uses unrolled loops for increments equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c      double precision dx(1),dy(1),da **********************
      double precision dx(n),dy(n),da
      integer i,incx,incy,ix,iy,m,mp1,n
c
      if(n.le.0)return
      if (da .eq. 0.0d0) return
      if(incx.eq.1.and.incy.eq.1)go to 20
c
c        code for unequal increments or equal increments
c          not equal to 1
c
      ix = 1
      iy = 1
      if(incx.lt.0)ix = (-n+1)*incx + 1
      if(incy.lt.0)iy = (-n+1)*incy + 1
      do 10 i = 1,n
        dy(iy) = dy(iy) + da*dx(ix)
        ix = ix + incx
        iy = iy + incy
   10 continue
      return
c
c        code for both increments equal to 1
c
c
c        clean-up loop
c
   20 m = mod(n,4)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dy(i) = dy(i) + da*dx(i)
   30 continue
      if( n .lt. 4 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,4
        dy(i) = dy(i) + da*dx(i)
        dy(i + 1) = dy(i + 1) + da*dx(i + 1)
        dy(i + 2) = dy(i + 2) + da*dx(i + 2)
        dy(i + 3) = dy(i + 3) + da*dx(i + 3)
   50 continue
      return
      end
C
      subroutine dscal(n,da,dx,incx)
c
c     scales a vector by a constant.
```

```
c     uses unrolled loops for increment equal to one.
c     jack dongarra, linpack, 3/11/78.
c
c      double precision da,dx(1) **************************
      double precision da,dx(n)

      integer i,incx,m,mp1,n,nincx
c
      if(n.le.0)return
      if(incx.eq.1)go to 20
c
c        code for increment not equal to 1
c
      nincx = n*incx
      do 10 i = 1,nincx,incx
        dx(i) = da*dx(i)
   10 continue
      return
c
c        code for increment equal to 1
c
c        clean-up loop
c
   20 m = mod(n,5)
      if( m .eq. 0 ) go to 40
      do 30 i = 1,m
        dx(i) = da*dx(i)
   30 continue
      if( n .lt. 5 ) return
   40 mp1 = m + 1
      do 50 i = mp1,n,5
        dx(i) = da*dx(i)
        dx(i + 1) = da*dx(i + 1)
        dx(i + 2) = da*dx(i + 2)
        dx(i + 3) = da*dx(i + 3)
        dx(i + 4) = da*dx(i + 4)
   50 continue
      return
      end
```

B.3 Calculation of internal entropy production

The local internal entropy production is calculated and integrated over the entire volume.

```
Cubic Spline on staggered grid for
Continuum mechanics are used on a rectangular grid to
Calculate the Rayleigh-Be'nard flow patterns.
Conservative form of differential equations are used.
      PROGRAM entro
      INCLUDE 'rbcu.h'
C     set up output data files
C     name output files
c
      CHARACTER*50 IN_STATE_FILE  /'Rax5/cufd500M1c.state'/
      CHARACTER*50 E_FILE         /'Rax5/cufd500M1c.e'/
C           >>>>>>>>>>>>>>> f77 -O -r8 -o xen500 entro.f

      Ra = 500 000.0d0
      IX=0
      IY=0
      cn=0.9d0
      delta=1.0d0
      rKb= 1.0d0
      bulk=0.0d0
      Th = 1.5d0
      Tl = 0.5d0
      rLy= Ny*delta
      rLx= 2.0d0*rLy
      Pr = 1.0d0
      eta= rLy*Sqrt(Pr/Ra)
      rkappa=eta/Pr
      g = -rKb*(Th-Tl)/rLy
      dt=delta*cn/Sqrt(2.0*Th)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      tottime=20000.0d0
      maxiter=INT(tottime/dt)
      iprfreq=maxiter/1000
c$IF(THREE_D_STOKES_FLUID)
      bulk=eta/3.0d0
c$ENDIF
C
C     initialize variables
      CALL INITIAL(IN_STATE_FILE,dt)
CCCCCC
CCCCCC
C     START ITERATION
      DO iter=1,maxiter
         CALL RK(dt)
         time=time+dt
c$IF(PRINT_E_FILE)
C     output to ke grid center data file
         if (jmod(iter,iprfreq).eq.0) then
C*****************************************************************
      totENT=0.0d0
      totENM=0.0d0
      DO i=1,Nx
         ip1=i+1
         if(i.eq.Nx) ip1=1
         DO j=1,Ny
            G2T=(e(i,j)+e(i,j+1)+e(ip1,j)+e(ip1,j+1))/4.0d0
            G2T2=G2T*G2T
            totENT=totENT- (Fte(i,j)*ex(i,j) + Fte(i,j+1)*ex(i,j+1)
     2           + Gte(i,j)*ey(i,j) + Gte(ip1,j)*ey(ip1,j))
     3           /(2.0d0*G2T2)
            totENM=totENM+ (eta*(G2ux(i,j)-G2vy(i,j))**2
     2           + eta*(G2uy(i,j)+G2vx(i,j))**2
     3           + bulk*(G2ux(i,j)+G2vy(i,j))**2
     4           )/(G2T)

c            totENM=totENM+ (2.0d0*eta*(G2ux(i,j)**2+G2vy(i,j)**2)
c     2           + eta*(G2uy(i,j)+G2vx(i,j))**2
c     3           + bulk*(G2ux(i,j)+G2vy(i,j))**2
c     4           )/G2T
c     4           - Rho(i,j)*G2T*(G2ux(i,j)+G2vy(i,j)))/G2T
c            totENM=totENM+ (eta*(G2ux(i,j)-G2vy(i,j))**2
c     2           + eta*(G2uy(i,j)-G2vx(i,j))**2
c     3           + bulk*(G2ux(i,j)+G2vy(i,j))**2
c     4           - Rho(i,j)*G2T*(G2ux(i,j)+G2vy(i,j)))/G2T
         END DO
      END DO
C*****************************************************************
      open(14,file=E_FILE,form='formatted',ACCESS='APPEND'
     2     ,STATUS= 'UNKNOWN')
      write(14,'(f10.3,2(3x,f18.14))') time, totENT, totENM
      close(14)
         end if
c$ENDIF
      END DO
      STOP
      END
C
C-----------------------------------------------------------------X
C     Sets initial and boundary values
      SUBROUTINE INITIAL(IN_STATE_FILE,dt)
      INCLUDE 'rbcu.h'
      CHARACTER*50 IN_STATE_FILE
      DIMENSION f1(Nx), f2(Ny+1), f1b(Nx), f2b(Ny+1)
C     initialize grid point values (note period bc in x-direction)
c$IF(READ_STATE_FILE)
      open(22,file=IN_STATE_FILE,form='formatted',status='OLD')
      DO i=1,Nx
         DO j=1,Ny
            read (22,'(2i5,4f24.18)',END=99)ii,jj,rho(i,j),
     2           u(i,j),v(i,j),e(i,j)
            if(i.ne.ii) write(*,*) "error"
            if(j.ne.jj) write(*,*) "error"
         END DO
         j=Ny+1
         read (22,'(2i5,4f24.18)',END=99)ii,jj,
     2           u(i,j),v(i,j),e(i,j)
      END DO
 99   close(22,STATUS='KEEP')
C     enforce boundary conditions
      DO i=1,Nx
         e(i,1)=Th
         u(i,1)=0.0d0
         v(i,1)=0.0d0
         u(i,Ny+1)=0.0d0
         v(i,Ny+1)=0.0d0
         e(i,Ny+1)=Tl
      END DO
c$ENDIF
C     interpolate density value to all grids
C     interpolate new density to other grids
C     rho -> G4rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=rho(i,j)
         END DO
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G4rho(i,j)=f2b(j-1)
         END DO
      END DO
C     rho -> G3rho
      DO j=1,Ny
```

```
         DO i=1,Nx
            f1(i)=rho(i,j)
         END DO
         CALL CubePbINT(f1,f1b,delta,Nx)
         G3rho(1,j)=f1b(Nx)
         DO i=2,Nx
            G3rho(i,j)=f1b(i-1)
         END DO
      END DO
C     G3rho -> G1rho
      DO i=1,Nx
         DO j=1,Ny
            f2(j)=G3rho(i,j)
         END DO
C         CALL PADEFbINTV(f2,f2b,Ny-1)
         CALL CubeFbINT(f2,f2b,delta,Ny-1)
         DO j=2,Ny
            G1rho(i,j)=f2b(j-1)
         END DO
      END DO
C
C     initialize "conserved" variables on grid
      DO j=1,Ny+1
         DO i=1,Nx
            Xm(i,j) = G1rho(i,j)*u(i,j)
            Ym(i,j) = G1rho(i,j)*v(i,j)
            Te(i,j) = G1rho(i,j)*(e(i,j)
     &            +(u(i,j)*u(i,j)+v(i,j)*v(i,j))/2.0d0)
         END DO
      END DO
      RETURN
      END
C
C-----------------------------------------------------------------------X
C

      INCLUDE 'rk.sub.f'

      INCLUDE 'cubic.sub.f'
```

Appendix C

MATHEMATICA scripts

C.1 Velocity and temperature field plots

This MATHEMATICA script generates a plot of the temperature and velocity fields. The velocity is represented as black arrows whose length is proportional to the normalized velocity ($\max|v| \equiv 1.0$. The color is represented as a background color from red (hot) to blue (cold).

```
Needs["Graphics`PlotField`"]
hot[val_] = RGBColor[Sin[val*Pi/2.0],0.5*Sin[val*Pi],Cos[val*Pi/2.0]];

<<new/cufd15OMM6hc.dat

dm=Max[Abs[dat]]; dat=dat/dm;

cm=Max[Abs[cdat]]; cdat=cdat/cm;

fg=ListPlotVectorField[dat,DisplayFunction->Identity];
bg=ListContourPlot[cdat, DisplayFunction->Identity,
        Contours->25,
        ContourLines->False,ColorFunction -> hot];
temp=Show[bg,fg];
optlist=temp[[2]];
fglist=temp[[1]][[2]];
bglist=temp[[1]][[1]];
nt=Dimensions[bglist][[1]];
newbg={};
For[i=1,i<=nt,i++,
        fp=bglist[[i]][[1]];
        sp=bglist[[i]][[2]];
        newbg=Append[newbg,{fp,sp}];
];
new=Drop[temp,1];
new=Prepend[new,{newbg,fglist}];

p=Show[new, DisplayFunction->$DisplayFunction,Frame->False,
        AspectRatio->2]

Display["gfile",p]
!psfix -land gfile>cufd20Msc.eps
```

C.2 Vorticity field plots

The vorticity of the velocity field is represented as a color scale from blue (counter-clockwise) to green (clockwise rotation).

```
(*The Color function returns a scale between
bluegreen[val_] = RGBColor[0.5,Cos[val*Pi/2.0],Sin[val*Pi/2.0]];

<<new/cufd150MM6hc.dat
dm=Max[Abs[dat]];
dat=dat/dm;

nn=Dimensions[dat];nx=nn[[1]];ny=nn[[2]];

(*note that dat file is pairs of (uy, -ux) *)
wdat={};
For[i=1,i<nx,i++,
        temp={};
        For[j=1,j<ny,j++,
           utop=-(dat[[i]][[j+1]][[2]]+dat[[i+1]][[j+1]][[2]])/2;
           ubottom=-(dat[[i]][[j]][[2]]+dat[[i+1]][[j]][[2]])/2;
           vleft=(dat[[i]][[j]][[1]]+dat[[i]][[j+1]][[1]])/2;
           vright=(dat[[i+1]][[j]][[1]]+dat[[i+1]][[j+1]][[1]])/2;
           w=(vright-vleft)-(utop-ubottom);
           temp=Append[temp,w];
        ];
        wdat=Append[wdat,temp];
];
wm=Max[Abs[wdat]];
wdat=-wdat/wm;
wdat=Transpose[wdat];

b=ListContourPlot[wdat,Contours ->25,DisplayFunction->Identity];
p=Show[b,AspectRatio -> 0.5,DisplayFunction->$DisplayFunction,
        ColorFunction->bluegreen,Frame -> False];
Display["gfile",p]
!psfix -epsf -width 4.75 -height 2 -stretch gfile>rb150MMw.eps
```

Bibliography

1. H. Bénard. Les tourbillons cellulaires dans une nappé liquide. *Rev. Gén. Sciences Pure Appl.*, 11:(1261, 1309), 1900.

2. H. Bénard. Les tourbillons cellulaires dans une nappé liquide transportant de la chaleur par convection en régime permanent. *Annls. Chim. Phys.*, 23:(62), 1901.

3. Lord Rayleigh. On convection currents in a horizontal layer of fluid when the higher temperature is on the under side. *Phil. Mag.*, 32:529, 1916.

4. A. Oberbeck. Uber die wärmeleitung der flüssigkeiten bei berücksichtigung der strömungen infolge von temperaturdifferenzen. *Ann. Phys. Chem.*, 7:271, 1879.

5. J. Boussinesq. *Théory Analytique de la Chaleur*. Gauthier-Villars, 1903.

6. H. Jeffreys. The stability of a layer of fluid heated from below. *Phil. Mag.*, 2:833, 1926.

7. H. Jeffreys. Some cases of instability in fluid motion. *Proc. Roy. Soc. London A*, 118:195, 1928.

8. O. G. Sutton. On the stability of a fluid heated from below. *Proc. Roy. Soc. London A*, 204:297, 1961.

9. S. Chandrasekhar. *Hydrodynamic and Hydromagnetic Stability*. Oxford Press, London, 1961.

10. A. Puhl, M. Malek, and M. Mareschal. Quantitative comparison of molecular dynamics with hydrodynamics in Rayleigh-Bénard convection. *Phys. Rev. A*, 40:1999, 1989.

11. A. Garcia. Nonequilibrium hydrodynamic fluctuations and computer simulations. In M. Mareschal, editor, *Microscopic Simulations of Complex Flows*. Plenum, New York, 1990.

12. D. C. Rapaport. Time-dependent patterns in atomistically simulated convection. *Phys. Rev. A*, 43:7046, 1991.

13. X. Shan. Simulation of Rayleigh-Bénard convection using a lattice Boltzmann method. *Phys. Rev. E*, 55:2780, 1997.

14. O. Kum. *Nonequilibrium Flows with Smooth Particle Applied Mechanics*. PhD thesis, U. C. Davis, 1995.

15. V. M. Castillo, Wm. G. Hoover, and C. G. Hoover. Coexisting attractors in compressible Rayleigh-Bénard flow. *Phys. Rev. E*, 55:5546, 1997.

16. W. V. R. Malkus. Discrete transitions in turbulent convection. *Proc. Roy. Soc. London A*, 225:185, 1954.

17. G. E. Willis and J. W. Deardorff. Confirmation and renumbering of thge discrete heat flux transitions of malkus. *Phys. Fluids*, 10:1861, 1967.

18. R. Krishnamurti. On the transition to turbulent convection. Part 1. The transition from two- to three-dimensional flow. *J. Fluid Mech.*, 42:295, 1970.

19. R. Krishnamurti. On the transition to turbulent convection. Part 2. The transition to time-dependent flow. *J. Fluid Mech.*, 42:309, 1970.

20. F. Heslot, B. Castaing, and A. Libchaber. Transitions to turbulence in helium gas. *Phys. Rev. A*, 36:5870, 1987.

21. F. H. Busse. The oscillatory instability of convection rolls in a low Prandtl number fluid. *J. Fluid Mech.*, 52:97, 1972.

22. R. M. Clever and F. H. Busse. Transition to time-dependent convection. *J. Fluid Mech.*, 65:625, 1974.

23. F. H. Busse. Non-linear properties of thermal convection. *Rep. Prog. Phys.*, 41:1929, 1978.

24. I. Goldhirsch, R. B. Pelz, and S. A. Orszag. Numerical simulation of thermal convection in a two-dimensional finite box. *J. Fluid Mech.*, 199:1, 1989.

25. E. E. DeLuca, J. Werne, R. Rosner, and F. Cattaneo. Numerical simulations of soft and hard turbulence: Preliminary results for two-dimensional convection. *Phys. Rev. Lett.*, 64:2370, 1990.

26. J. Werne, E. E. DeLuca, R. Rosner, and F. Cattaneo. Development of hard-turbulent convection in two dimensions: Numerical evidence. *Phys. Rev. Lett.*, 67:3519, 1991.

27. J. Werne. Structure of hard-turbulent convection in two dimensions: Numerical evidence. *Phys. Rev. E*, 48:1020, 1993.

28. V. M. Castillo and Wm. G. Hoover. Heat flux at the transition from harmonic to chaotic flow in thermal convection. *Phys. Rev. E*, 58:4016, 1998.

29. V. M. Castillo and Wm. G. Hoover. Entropy production and Lyapunov instability at the onset of turbulent convection. *Phys. Rev. E*, 58:7350, 1998.

30. L. Prandtl. Meteorologische anwendungen der strömunglsehre. *Beitr. Z. Phys. Atmos.*, 19:188, 1932.

31. C. H. B. Priestley. Convection from a large horizontal surface. *Austral. J. Phys.*, 7:176, 1954.

32. L. N. Howard. Heat transport by turbulent convection. *J. Fluid Mech.*, 17:405, 1963.

33. B. Castaing and *et alii*. Scaling of hard thermal turbulence in Rayleigh-Bénard convection. *J. Fluid Mech.*, 240:1, 1989.

34. Z.-S. She. On the scaling laws of thermal turbulent convection. *Phys. Fluids A*, 1:911, 1989.

35. B. I. Shraiman and E. D. Siggia. Heat transport in high-rayleigh-number convection. *Phys. Rev. A*, 42:3650, 1990.

36. Y. B. Xin and K. Q. Xia. Boundary layer length scales in convective turbulence. *Phys. Rev. E*, 56:3010, 1997.

37. A. Naert, T. Segawa, and M. Sano. High-Reynolds-number thermal turbulence in mercury. *Phys. Rev. E*, 56:R1302, 1997.

38. E. Ching. Heat flux and shear rate in turbulent convection. *Phys. Rev. E*, 55:1189, 1997.

39. X. Z. Wu and A. Libchaber. Non-Boussinesq effects in free thermal convection. *Phys. Rev. A*, 43:2833, 1991.

40. J. Zhang, S. Childress, and A. Libchaber. Non-Boussinesq effect: Thermal convection with broken symmetry. *Phys. Fluids*, 9:1034, 1997.

41. E. A. Spiegel. Convective instability in a compressible atmosphere. *Astrophys. J.*, 141:1068, 1964.

42. S. Gauthier. A spectral collocation method for two-dimensional compressible convection. *J. Comp. Phys.*, 75:217, 1988.

43. J. Frölich and R. Peyret. Calculations of non-Boussinesq convection by a pseudospectral method. *Computer Methods in Applied Mechanics and Engineering*, 80:425, 1990.

44. S. Gauthier, T. Desmarais, and G. Iooss. Steady compressible convection. *Europhys. Lett.*, 10:543, 1989.

45. N. H. Brummell, N. E. Hurlburt, and J. Toomre. Turbulent compressible convection with rotation. I. Flow structure and evolution. *The Astrophysical Journal*, 473:494, 1996.

46. F. Cattaneo, N. H. Brummell, J. Toomre, A. Malagoli, and N. E. Hurlburt. Turbulent compressible convection. *The Astrophys. J.*, 370:282, 1991.

47. J. Toomre, N. H. Brummell, F. Cattaneo, and N. E. Hurlburt. Three-dimensional compressible convection at low Prandtl numbers. *Computer Physics Communications*, 59:105, 1990.

48. K. A. Emanuel. *Atmospheric Convection*. Oxford Press, New York, 1994.

49. J. B. Klemp. Dynamics of tornadic thunderstorms. *Ann. Rev. Fluid Mech.*, 19:369, 1987.

50. J.B.Pollack, D. Rind, A. Lacis, J.E. Hansen, Mki. Sato, and R. Ruedy. Gcm simulations of volcanic aerosol forcing. Part I: Climate changes induced by steady-state perturbations. *J. Climate*, 6:1719, 1993.

51. P. van Keken. Evolution of starting mantle plumes: a comparison between numerical and laboratory models. *Planetary Science Letters*, 148:1–14, 1997.

52. G. F. Davies and M. A. Richards. Mantle convection. *J. of Geology*, 100:151, 1992.

53. L. H. Kellogg. personal communication.

54. L. H. Kellogg and D. L. Turcotte. Mixing and the distribution of heterogeneities in a chaotically convecting mantle. *Journal of Geophysical Research*, 95:421–43, 1990.

55. G. A. Bird. *Molecular Gas Dynamics*. Clarendon, Oxford, 1976.

56. G. A. Bird. *Molecular Dynamics and the Direct Simulation of Gas Flow*. Oxford Science Publications, Oxford, 1994.

57. U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier-Stokes equation. *Phys. Rev. Lett.*, 56:1505, 1986.

58. S. Wolfram. *Theory and applications of cellular automata*. World Scientific, Singapore, 1986.

59. W. G. Hoover. *Computational Statistical Mechanics*. Elsevier, Amsterdam, Amsterdam, 1991.

60. J. J. Monaghan. Smoothed particle hydrodynamics. *Rev. Astron. Astrophys.*, 30:543, 1992.

61. J. P. Boeuf and E. Marode. A monte carlo analysis of an electron swarm in a nonuniform field: The cathode region of a glow discharge in helium. *J. Phys. D*, 15:2169, 1982.

62. W. B. Bateson and D. W. Hewett. Grid and particle hydrodynamics: Beyond hydrodynamics via fluid element particle-in-cell. *J. Comp. Phys.*, 144:358, 1997.

63. S. K. Lele. Compact finite difference schemes with spectral-like resolution. *J. Comp. Phys*, 103:16, 1992.

64. G. Fairweather and D. Meade. A survey of spline collocation methods for the numerical solution of differential equations. In J. C. Diaz, editor, *Mathematics for Large Scale Computing*, page 297. Marcel Dekker, inc., New York, 1989.

65. C. A. J. Fletcher. *Computational Techniques for Fluid Dynamics*, volume I. Springer-Verlag, Berlin, 1991.

66. V. M. Castillo. Cubic-spline technique for solving the compressible navier-stokes equations for turbulent convection. Presented at 1997 International Conference on Computational Physics: PC'97 (Santa Cruz), August 1997.

67. V. M. Castillo and Wm. G. Hoover. Comment on 'Maximum of the local entropy production becomes minimal in stationary processes. *Phys. Rev. Lett.*, 81:5700, 1998.

68. A. C. Hindmarsh. personal communication.

69. C. Hirsch. *Numerical Computations of Internal and External Flows*, volume I. Wiley, New York, 1988.

70. H. R. Schwarz. *Numerical Analysis: A comprehensive Introduction*, chapter 9.3. Wiley & Sons, 1989.

71. A. Schluter, D. Lortz, and F. Busse. On the stability of steady finite amplitude convection. *J. Fluid Mech.*, 23:129, 1965.

72. H. E. Trease, M. J. Fitts, and W. P. Crowley, editors. *Advances in the Free-Lagrange Method*, volume 395 of *Lecture Notes in Physics*, Berlin, 1991. Springer-Verlag.

73. T. Watanabe, H. Kaburaki, M. Machida, and M. Yokokawa. Growth of long-range correlations in a transition between heat conduction and convection. *Phys. Rev. E*, 52:1601, 1995.

74. H. Struchtrup and W. Weiss. Maximum of the local entropy production becomes minimal in stationary processes. *Phys. Rev. Lett.*, 80:5048, 1998.

75. R. B. Bird, W. E. Stewart, and E. N. Lightfoot. *Transport Phenomena*. Wiley, New York, 1960.

76. S. Newhouse, D. Ruelle, and F. Takens. Occurrence of strange axiom-A attractors near quasiperiodic flow on $T^m, m \geq 3$. *Commun. Math. Phys.*, 64:35, 1978.

77. J. P Eckman and D. Ruelle. Ergodic theory of chaos and strange attractors. *Rev. Mod. Phys.*, 57:617, 1985.

78. F. Takens. *Lecture Notes in Mathematics*, volume 898. Springer-Verlag, Berlin, 1981.

79. H. A. Posch and W. G. Hoover. Equilibrium and nonequilibrium Lyapunov spectra for dense fluids and solids. *Phys. Rev. A*, 39:2175, 1989.

80. G. Benettin and J.-M. Strelcyn. Numerical experiments on the free motion of a point mass, moving in a plane convex region: Stochastic transition and entropy. *Phys. Rev. A*, 17:773, 1978.

81. R. H. Kraichnan. Turbulent thermal convection at arbitrary Prandtl number. *Phys. Fluids*, 5:1374, 1962.

82. K. R. Sreenivasan. Effect of intermittency on predicatbility estimates. *Physica D, Special issue on Predictability*, 1998.

83. V. M. Castillo. A convecting double pendulum as a simple model for the investigation of transitions in time-dependent Bénard convection. Presented at 18th Annual CNLS Conference: Predictability (Los Alamos), May 1998.