A computational framework for solving coupled equation systems using finite element method and introduction to a versatile fault-tolerant toolkit for high throughput batch processing

by

Yu Xie

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Mechanical Engineering

Program of Study Committee:

Baskar Ganapathysubramanian, Major Professor

Karin Dorman

Theodore Heindel

Sriram Sundararajan

Jue Yan

Iowa State University

Ames, Iowa

2015

DEDICATION

I would like to dedicate this thesis to my wife Minwen without whose encouragement and trust I would not have been able to complete this work.

I am also always feeling deeply grateful to my parents for their support to me during these years I spent at school, college and research lab.

Thanks to my friends for all their help.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES vii

ACKNOWLEDGEMENTS xi

CHAPTER 1. GENERAL INTRODUCTION 1

1.1 Introduction 1

1.2 My contributions 3

1.3 Outline of this thesis 4

CHAPTER 2. PHASE TRANSITIONS IN VORTEX SHEDDING IN THE WAKE OF A HEATED CIRCULAR CYLINDER AT LOW REYNOLDS NUMBER 6

2.1 Abstract 6

2.2 Introduction 7

2.3 Problem description and governing equations 10

2.4 Numerical Methods 13

2.5 Convergence tests and validation 21

2.6 3D flow transition 25

2.7 Vortex shedding pattern transition 29

2.8 Conclusions 33

CHAPTER 3. A DIFFUSE INTERFACE MODEL FOR INCOMPRESSIBLE TWO-PHASE FLOW: STABILIZED FINITE ELEMENT METHOD FOR LARGE DENSITY RATIOS, GRID RESOLUTION STUDY, AND 3D PATTERNED SUBSTRATE WETTING PROBLEM 36

3.1 Abstract 36

3.2 Introduction 37

3.3 Governing equations 41

3.4 Numerical Schemes 46

3.5 Validations and examples 52

3.5.1 Convergence tests 52

3.5.2 Capillary wave 53

3.5.3 Rayleigh-Taylor instability 60

3.5.4 Droplet impact on liquid surface 64

3.5.5 3D droplet impact on solid surface 68

3.5.6 3D droplet impact on patterned substrate 69

3.6 Scalability test on the numerical framework 78

3.7 Conclusion 81

CHAPTER 4. FAULT TOLERANT ADAPTIVE SPARSE GRID COLLOCATION OVER HETEROGENEOUS COMPUTING ARCHITECTURES 82

4.1 Abstract 82

4.2 Introduction 83

4.3 Problem definition 86

4.3.1 Governing stochastic equations 86

4.4 Adaptive sparse grid collocation method 87

4.5 Basic requirements of designing fault-tolerant systems 89

4.5.1 Definition of campaign 89

4.5.2 A brief introduction to the axiomatic approach 90

4.5.3 Recovery protocol 91

4.5.4 Basic rules of a fault-tolerant system 92

4.6 Design a fault-tolerant implementation framework for sparse grid collocation method . . . 92
4.6.1 Conventional "static" linking strategy . . . 92
4.6.2 Desired properties of the fault-tolerant framework . . . 94
4.6.3 Fault-tolerant framework for ASGC . . . 95
4.7 Fault-tolerant framework implementation methods . . . 96
4.7.1 Architecture of the "dynamic" linking ASGC framework . . . 96
4.7.2 Communication between manager process and sub-processes . . . 98
4.7.3 Functions of the monitoring unit . . . 99
4.7.4 Data storage optimization . . . 102
4.8 Numerical examples . . . 105
4.8.1 Stochastic elliptic problem . . . 105
4.8.2 High discontinuity interpolation . . . 111
4.9 Conclusions . . . 113
CHAPTER 5. GENERAL CONCLUSION . . . 116
5.1 General discussions . . . 116
5.2 Potential future work . . . 118
BIBLIOGRAPHY . . . 120

LIST OF TABLES

Table 2.1 Coefficients of the polynomial fittings of the thermalphysical properties of water with temperature. 13

Table 3.1 Relative L_∞ and relative L_2 errors for the interface discretization tests. 56

Table 3.2 Relative L∞ and relative L2 errors for the convergence tests of Cahn number Cn. 58

LIST OF FIGURES

Figure 2.1 Schematic diagram of the flow past a heated cylinder. 11

Figure 2.2 Thermalphysical properties of saturated water at different temperatures. 14

Figure 2.3 Flowchart of the iteration solver for the semi-coupled Navier-Stokes and heat convection/diffusion equation system. 17

Figure 2.4 3D mesh used for simulating the flow past a heated cylinder. 20

Figure 2.5 Spatial convergence results for the simulation of flow past a heated cylinder. 22

Figure 2.6 Temporal convergence results for the simulation of flow past a heated cylinder. 23

Figure 2.7 Comparisons between the numerical results and the experimental measurements for the flow past a heated cylinder simulation. 24

Figure 2.8 Comparison of the vortex shedding patterns under different temperatures between experimental observations. 25

Figure 2.9 Ensemble-averaged centerline velocity plots at different Richardson numbers for aspect ratios of 6.3, 5.0, 4.0 and ∞. 28

Figure 2.10 Wake closure length plotted for all Richardson numbers at aspect ratio of 4.0, 5.0, 6.3 and ∞. 29

Figure 2.11 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = 5.0$. 30

Figure 2.12 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = 4.0$. 31

Figure 2.13 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = \infty$. 32

Figure 2.14 Central surface vortex strength levels plotted for all Richardson numbers at $ar = 4.0$, 5.0, 6.3 and ∞. 34

Figure 3.1 Flow chart of the iteration solver for the semi-coupled Cahn-Hilliard Navier-Stokes equation system. 48

Figure 3.2 Temporal and spatial convergence tests. 53

Figure 3.3 Evolution of interface oscillation magnitude with various number of elements per interface. 56

Figure 3.4 Effect of Cahn number Cn on the evolution of interface oscillation magnitude. 57

Figure 3.5 Evolution of interface oscillation with different density and viscosity ratios. 59

Figure 3.6 Convergence rates of Cn with density ratios in different ranges. 61

Figure 3.7 Rayleigh-Taylor instability. Snapshots of the interface at different time for the case $At = 0.50$. 63

Figure 3.8 The evolution of the peak of the rising fluid and the bottom of the falling fluid in the simulation of Rayleigh-Taylor instability. 64

Figure 3.9 Snapshots of the interface at different time for the case $At = 0.82$ in the simulation of Rayleigh-Taylor instability. 65

Figure 3.10 Snapshots of the interface at different time for the case $At = 0.98$ in the simulation of Rayleigh-Taylor instability. 66

Figure 3.11 Snapshots of the simulation of a 2D Droplet impacting on a liquid surface. 68

Figure 3.12 Snapshots of the simulation of a 3D droplet impact on a solid surface. 70

Figure 3.13 Evolution of droplet wetting spot and droplet height of the 3D droplet impacting on a solid surface. 70

Figure 3.14 Snapshots of a 3D droplet impacting on a groove patterned solid surface. 72

Figure 3.15 Formation of the wetting spots above and inside the grooves during the process of a 3D droplet impacting on a groove patterned solid surface. 74

Figure 3.16 Snapshots of a 3D droplet impacting on a checker patterned solid surface. 75

Figure 3.17 Formation of the wetting spots above and on the checkered surface during the process of a 3D droplet impacting on a checker patterned solid surface. 76

Figure 3.18 Evolution of the wetting diameter and height of a 3D droplet impacting on patterned surfaces. 78

Figure 3.19 Scalability tests of the numerical solver for the diffuse interface model simulation. 80

Figure 4.1 Architecture of the "dynamic" linking framework for implementing the ASGC method. 98

Figure 4.2 Diagram showing the communication between the manager process and the monitoring units 100

Figure 4.3 Flowcahrt of the monitoring unit working procedure. 103

Figure 4.4 Mean and variance of temperature field for stochastic elliptic problem with correlation length $L = 1$. Stochastic dimension $N = 5$, discretization with 300×300 elements. 107

Figure 4.5 Mean and variance of temperature field for stochastic elliptic problem with correlation length $L = 1/8$. Stochastic dimension $N = 5$, discretization with 300×300 elements. 107

Figure 4.6 Convergence histories for the stochastic elliptic problems with large physical scales. Stochastic dimension $N = 5$, 300×300 elements. Left:$L = 1$; right: $L = 1/8$. 108

Figure 4.7 Mean and variance of temperature for stochastic elliptic problem with high stochastic dimensions, $N = 100$. 109

Figure 4.8 Mean and variance of temperature for stochastic elliptic problem with high stochastic dimensions, $N = 200$. 110

Figure 4.9 Convergence histories for stochastic elliptic problems with high stochastic dimensions. Left:$N = 100$; right: $N = 200$. 110

Figure 4.10 Increase of average run time with the probability of the occurrence of artificial processor exceptions. 112

Figure 4.11 A five-colored map in a square domain. Each block is assigned with a color of unique gray scale. 113

Figure 4.12 Growth of the ASGC collocation points for detecting the discontinuity in the five-colored map at different interpolation levels. 114

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Baskar Ganapathysubramaian for his guidance, patience and support throughout this research and the writing of this thesis. He led me into the exciting world of computational mechanics and broadened my vision with valuable knowledge and experiences which will be the treasure for my whole life. I would also like to thank my committee members, Dr. Karin Dorman, Dr. Theodore Heindel, Dr. Sriram Sundararajan, and Dr. Yan Jue, for their efforts and contributions to this work. I would additionally like to thank all my colleagues in the lab, for their help and fruitful discussions through these many years.

CHAPTER 1. GENERAL INTRODUCTION

1.1 Introduction

Simulation of complex fluid systems has been attracting more and more attention. Though theoretical analysis and scientific experiments are both powerful tools for unveiling secrets hidden behind the phenomena or providing guidance to industrial design and manufacture, computational study still shows its great advantages when the theoretical analysis is stuck seeking an appropriate mathematical expression or the experimental investigation is restricted by the lack of accuracy for a small scale observation. Complex fluid systems are commonly described by coupled physical equations. For example, the flow past a heated object is governed by both the fluid dynamic equations and the heat convection/diffusion equation. To be able to simulate the motion of an air bubble in water, or the deformation of an oil droplet hitting a surface, we need to solve both the fluid equation and the equation governing the motion of the the interface between liquid and air. Thus to accurately simulate the behavior of the complex fluid systems, it is crucial to have accurate/efficient equation solvers, and to think about the strategy to handle the coupling of these equations.

Various numerical methods of solving partial differential equations have been developed, for instance, finite difference method (FDM) (Morton and Mayers (2005); Rübenkönig (2006)), finite volume method (FVM) (Versteeg and Malalasekera (2007); Anderson et al. (1995)), and finite element method (FEM) (Hughes (2012); Brooks and Hughes (1982); Oden (1973)). In this study, we adopt the finite element method for solving the physical equations for its ease of implementation, convenience of parallelization, and flexibility for dealing with various physical problems. Brooks and Hughes (1982) introduced the streamline upwind/Petrov-Galerkin (SUPG) terms to stabilize the velocity solution for convection dominated incompressible flows,

where the spurious oscillations of the numerical solution is eliminated from the traditional Petrov-Galerkin method. Tezduyar et al. (1992) improved this technique by adding the pressure stabilization Petrov-Galerkin (PSPG) terms where velocity and pressure can be interpolated on equal order elements.

Complex fluid systems involving coupled equations usually present more physical properties and behaviors than the ones governed by a single equation system. Taking the problem of flow past a single cylinder as example, the vortex shedding behind the cylinder forms a staggered pattern known as the "von Karman vortex street", which is a result of the conflict between the inertia and viscosity of the fluid. When a heat source is added to the cylinder, the vortex shedding process presents more patterns as observed experimentally by Hu and Koochesfahani (2011). A numerical analysis provides the first step of understanding how the heat convection/diffusion influences the flow.

Though the investigation of complex/multi-physics systems leads us to a journey of exploring more wonders in nature, numerical solutions for such systems face many challenges, coming from both the physical side and the limitation of computational resources. A coupled equation system not only introduces more unknown physical variables and controlling parameters into the numeric scheme, but it also brings things like stability conditions and restrictions on grid and time step sizes for all the equation sets under consideration. For example, when simulating multiphase flows, we need to solve both the flow equations and interface dynamic equations. Wodo and Ganapathysubramanian (2011) pointed out that there exists a minimal spatial resolution of the grid through the interface to maintain the accuracy of the numerical solution to the interface motion. So when designing the numerical scheme for such a problem, we need to meet this condition, which is not required by most numerical schemes for the flow equations. However, reducing the grid size increases the consumption of computational resources, which is an important factor that we hope to control at a low level most of the time. Similar opposing requirements found when designing the numerical schemes motivate the need for smarter ways to handle the coupling of the equation sets. Thus a well designed numerical scheme, or solution algorithm, for handling the coupled equation systems becomes the key to success.

Another problematic feature of complex fluid systems is that they usually present strong nonlinearity and sensitivity to uncertainties. An outcome of this feature is that we often observe multiple phases of a certain behavior in the complex systems. For example, flow past a heated cylinder gives multiple vertex shedding patterns. Organic solar cell morphology is highly affected by the ratio of components in the solvent and spinning speed of the fabrication base (Wodo and Ganapathysubramanian (2012)). Velocity perturbation of the flow in the micro-channel can have significant response to nano-level roughness on the channel bottom (Jaeger et al. (2012)). Numerical simulations are performed on a large set of controlling parameters can help us investigate the features of these phases and study their transitions, then categorize their results based on their behaviors. One extension to this type of problem is the stochastic analysis using the adaptive sparse grid collocation (ASGC) method, where a large set of input parameters are a group of random variables. For all the above scenarios, the key requirement is how to efficiently handle these high throughput numerical tasks and how to manage the huge data set generated from these simulations. The ASGC method processes the stochastic calculation in a hierarchical pattern, where the next level of iteration is related to the previous ones. This implies that the numerical framework should be fault-tolerant to both machine failures and software level incidents occurred for the scientific solvers. This numerical framework should also be pluggable to various scientific solvers, and be flexible to a wide range of scales of physical problems.

1.2 My contributions

The purpose of this study is to provide such a framework covering both the implementation of the scientific solver for coupled equation systems and the development of a platform for handling high throughput simulation tasks. The following is a list of my contributions towards the design and implementation of such a framework.

(1) Derived the finite element formula for solving the coupled equation system, where the SUPG stabilization term is incorporated to stabilize the velocity solution in the convection dominant flows.

(2) Improved the `TalyFEM` FEM solver package by developing the domain decomposition technique, which partitions the whole grid into sub-domains distributed on multiple processors. This feature allows the solver to handle very large scale (hundreds of millions of degrees of freedoms) three dimensional problems.

(3) Treated the coupled equation sets in a semi-coupled way, where each equation set is solved alone and the coupled unknown variables are updated in an iterative procedure. This method reduces the number of degree of freedoms solved at the same time, which therefore reduces the consumption of computational resources and allows the solution of very large scale problems.

(4) Applied the coupled equation solver on simulating flow past a heated cylinder. The numerical analysis helped us verify the multiple vortex shedding patterns observed in experiment. We further investigated the conditions for the occurrence of these vortex shedding patterns for various cylinder aspect ratios.

(5) Applied the coupled equation solver to numerically study multiphase flows. We found that a minimum grid resolution is required through the interface to assure the accuracy of the solution to the interface motion. We also numerically investigated the convergence of the interface thickness for the diffuse interface model.

(6) Designed and developed a fault-tolerant framework for handling high throughput parametric driven scientific simulation tasks. This framework is designed with a friendly user interface and provides scientific solvers covering most of engineering problems. This framework is also scalable to any size of problem, where the numerical solutions are managed by a hard drive storage database.

1.3 Outline of this thesis

In this thesis, we introduce the numerical solution strategy in several chapters. Chapter 1 gives the overall motivations and guidelines for deriving such a framework. Chapter 2 to Chapter 4 comprise works modified from manuscripts to be submitted to journals, which discuss the details of derivation of the finite element solvers on coupled equation systems, validations, and their applications on various physical problems. These chapters also introduce the design

and implementation of the high throughput scientific solver batch processing tool AdaGiO and its applications on solving stochastic partial differential equations. At last, Chapter 5 summarizes the conclusions of the findings discussed in this thesis and lists potential applications and improvements could be extended from this study.

CHAPTER 2. PHASE TRANSITIONS IN VORTEX SHEDDING IN THE WAKE OF A HEATED CIRCULAR CYLINDER AT LOW REYNOLDS NUMBER

Modified from a paper to be submitted to *Physics of Fluids*

Yu Xie, Hui Hu, and Baskar Ganapathysubramanian

2.1 Abstract

The present study describes our numerical investigation on two transitions observed in the wake behind a horizontally placed heated circular cylinder in a closed channel with low Reynolds number, where the incoming flow is in the direction of gravitational force. The thermally deduced buoyancy coupled with the convection and viscous diffusion introduces further instability into the flow system. In this study, a finite element scheme is derived with inclusion of the streamline upwind Petrov-Galerkin (SUPG) stabilization terms to provide accurate numerical solution to the current hydraulic/thermal system. Our numerical method is validated by various comparisons with previous experimental studies. Our contributions include the provision of further understanding to the mechanism of the flow transition and vortex shedding pattern transition observed in previous experimental studies. Though the heat source on the cylinder has the core function to influence the wake flow structure, we demonstrated that the existence of a velocity gradient in the cylinder axis direction plays a crucial role on triggering both the flow transition and vortex shedding pattern transition. The numerical method enables us to mimic the situation where such velocity gradient is eliminated from the channel, and we found that both the flow transition and vortex shedding pattern transition vanish in such case. Our

hypothesis on the relation between the velocity gradient in the cylinder axis direction and the two transitions is verified by quantitative analysis shown in this report.

2.2 Introduction

The instability of the wake behind a bluff obstacle has been investigated by fluid mechanicians for several decades. The most significant phenomenon in wake flow is the shedding of vortices. These vortices can form the famous staggered pattern of vortex street which was first studied by Von Karman and Rubach (1912). The pattern of the vortex street can be influenced by external forces like the direction of gravity, angle of attack, and heating of cylinder. For the last century, attention was focused on the study of an isothermal cylinder [Von Karman and Rubach (1912); Green and Gerrard (1993); Roshko (1954); Williamson (1988)]. Characteristics of flow past an unheated cylinder are relatively well understood after extensive experimental, theoretical and numerical analysis [Williamson (1996)]. However, the study of flow past a heated body has not been extensively analyzed until recent years. Interest in this problem has been primarily encouraged by improving experimental and sophisticated computational techniques. Research on laminar flow past a heated circular cylinder leads to a better understanding of more complex flow, as well as to better serve industrial applications, particularly in electronic packaging, cooling and manufacturing.

For the problem of flow past an unheated circular cylinder, Von Karman and Rubach (1912) pointed out that the stability of the wake flow is only determined by the vortex street configurations. The Reynolds number Re is a crucial factor that decides the behavior of the wake flow. Green and Gerrard (1993) studied the bluff-body wake flow vortex shedding mechanism at low Reynolds number using the particle streak method. They gave a quantitative description of the vortex splitting phenomenon, and discovered that when the Reynolds number is small, shear stress dominates the vortex splitting, while a higher Reynolds number contributes more to the inertia shedding mechanism.

A representative feature of vortex shedding is a transition to three dimensional flow. Researchers have been trying to understand the mechanisms of the flow transition to three-dimensional flow. Roshko (1954) originally detected and analyzed the three-dimensional wake

flow transition. Based on hot-wire techniques, spectrum and statistical measurements, he studied experimental data from a low-speed wind tunnel with the purpose of investigating wake development behind circular cylinders. Laminar-turbulent transition occurs when the Reynolds number is in the "irregular range" (Re=150 to 300). Williamson (1988) observed that in the near wake of a circular cylinder the transition to three-dimensionality consists of two transition modes, known as "Mode A" and "Mode B". These two modes of instability have distinct wavelengths and frequencies. In another seminal paper, Williamson (1996) estimated the critical Reynolds number for the occurrence of the three-dimensional flow transition by using laser-induced fluorescence and Particle-Image-Velocimetry (PIV) techniques.

This vortex shedding process can be disturbed by external sources, like adding external force on the cylinder, heating or cooling the cylinder, and varying the angle of attack. Karniadakis and Triantafyllou (1989) employed the spectral-element method to study the response of wake flow to an external force to the cylinder. They pointed out that a synchronized vortex shedding occurs under vibration of the cylinder with large amplitudes. Slaouti and Gerrard (1981) recognized that the end effect of the cylinder causes the three-dimensional vortex tubes to slant with respect to the cylinder axis. Williamson (1989) later found methods to suppress the oblique shedding of vortex tubes by modifying the cylinder ends with diameter thickening or end plates.

Heating the cylinder is another aspect that influences the vortex shedding characteristics. The study of flow past heated objects has not received enough attention, although this problem has significant importance for both research and application [Kieft et al. (2003)]. Compared with forced convection in the case of the unheated cylinder, mixed convection caused by heating the cylinder involves viscous, inertial and buoyant forces which increase the difficulty of analysis (both experimental and numerical). The effect of heat input on the cylinder has previously been studied to determine the quantity of heat transfer between the heated cylinder and fluid with the purpose of improving hot-wire performance [Hatton et al. (1970)]. For modern manufacturing engineering, the study of the influence of heat input on the wake behind a circular cylinder provides information to improve electronic device cooling techniques and designs for heat exchange tubes [Varma et al. (2007)]. Both experimental and numerical studies reflect

that heating the cylinder can significantly change vortex shedding characteristics. Michaux-Leblond and Belorgey (1997) investigated ascendant flow passing a heated circular cylinder experimentally with laser Doppler velocimetry. They found that above a critical input heat, vortex shedding is gradually suppressed until the wake becomes a thermal plume. Lecordier et al. (2000) obtained similar results in their numerical simulation. They concluded that the sudden disappearance of vortex shedding is primarily due to the influence of input heat on viscosity. Chang and Sa (1990) showed more details about the near wake behind a heated/cooled circular cylinder with numerical methods. They observed the degeneration of the flow into a "steady twin-vortex" pattern. Studies have revealed that the vortex structure behind a heated circular cylinder can also be affected by changing the local features of the surrounding fluid or the obstacle. Noto and Matsumoto (1991) investigated the influence of attack angle on the stability of wake flow behind a heated circular cylinder. Kieft et al. (2003) studied the wake flow behind a heated horizontal cylinder both experimentally and numerically. They pointed out the difference of strength between the two rows of vortices is attributed to the downward motion of the shed vortex structures. They concluded that the thermally induced baroclinic vorticity production is the primary contribution for this phenomenon.

The aspect ratio of cylinder also plays an important role in determining the vortex shedding pattern and wake instability. Mittal (2001) performed a computational study of comparing vortex shedding modes of flow behind an unheated cylinder for various cylinder aspect ratios by using Direct Numerical Simulation (DNS). To the best knowledge of the authors, there appears to be no such analysis of vortex shedding modes for heated cylinder. We utilize numerical techniques to understand the instability of wake behind a heated cylinder with different cylinder aspect ratios. This study reveals that the behavior of wake flow near the heated cylinder under different cylinder aspect ratios are more complex than expected. In complementary experimental research [Hu and Koochesfahani (2005, 2011)] for flow past a heated cylinder, a mode shift of average velocity distribution along the axis on the wake central surface was found. This phenomenon is also detected in the present study.

This paper is organized as follows. In Section 2.3, problem description, governing equations and boundary conditions will be discussed. The stabilized finite element formula for the sim-

ulation is derived in Section 2.4, where the details of the simulations, such as solver package and machine information is provided as well. Section 2.5 discusses the spatial and temporal convergence behaviors of the present numerical method, and validates the solver by comparing the numerical results with experimental measurements. The 3D flow transition and the vortex shedding pattern transition caused by the heated cylinder will be investigated in Section 2.6 and 2.7, respectively, where the influence of the cylinder aspect ratio will be incorporated for both studies.

2.3 Problem description and governing equations

In this study, we simulate the forced flow past a heated cylinder horizontally placed between two lateral walls of a channel, as illustrated by the schematic in Figure 2.1. Denote the diameter of the cylinder as D, and length as L, where $D = 4.76\,\text{mm}$. The aspect ratio of the cylinder is then noted as $ar = L/D$. The width of the channel is $10.5D$, and length is $42D$. The cylinder locates at the center between the two side walls, and has a $10D$ distance to the inlet. Incoming flow with uniform velocity comes into the channel from the inlet with velocity $U_\infty = 0.0255\,\text{m/s}$, and has the same direction with the gravitational force. The incoming fluid has temperature T_0 of the surrounding environment. An equal or higher constant temperature T_1 ($T_1 \geq T_0$) is applied on the surface of the cylinder. Here we assume that the cylinder surface is made of highly thermal conductive material thus the surrounding water in touch with the cylinder has the same temperature of the cylinder. When $ar = 6.3$, $T_0 = 24\,^\circ\text{C}$ (297K), and T_1 varies from $24\,^\circ\text{C}$ (297K, unheated case) to $85\,^\circ\text{C}$ (358K), the geometrical and physical settings of the computational model are the same as the experimental set-up performed by Hu and Koochesfahani (2011).

Behavior of the fluid is governed by the transient incompressible Navier-Stokes equations with an external bouncy force caused by the density change of fluid due to heterogeneous temperature distribution:

$$\rho\left[\frac{\partial \boldsymbol{u}}{\partial t} + (\nabla\cdot\boldsymbol{u})\,\boldsymbol{u}\right] = \nabla\cdot\boldsymbol{\sigma} - \Delta\rho\boldsymbol{g}, \tag{2.1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2.2}$$

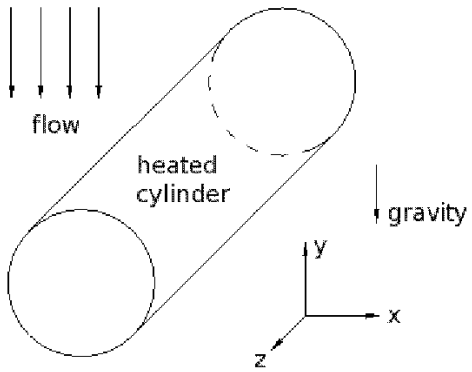


Figure 2.1 Schematic diagram of the problem. The heated cylinder is horizontally placed in a close channel between two lateral walls.

where ρ is the density of fluid, $\Delta\rho = \rho(T) - \rho(T_0)$ is the change of density due to the difference between local temperature T and reference temperature T_0, $\boldsymbol{u}$ is the velocity vector, t is time, $\boldsymbol{g}$ is the gravitational acceleration, and $\boldsymbol{\sigma}$ is the the stress tensor given as

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}), \tag{2.3}$$

with

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\left[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right], \tag{2.4}$$

and p, μ, and $\boldsymbol{I}$ being the pressure, the dynamic viscosity, and the identity tensor.

Evolution of temperature T is governed by the heat convection/diffusion equation:

$$\frac{\partial T}{\partial t} + (\nabla \cdot \boldsymbol{u})\, T - \nabla \cdot (\alpha \nabla T) = 0, \tag{2.5}$$

where $\alpha = k/(\rho C_p)$ is the thermal diffusivity of water, with k and C_p being the thermal conductivity and specific heat capacity, respectively.

Boundary conditions of the governing equations are now discussed. At the inlet, a uniform velocity field is applied. The temperature is also fixed at T_0. The uniform velocity inlet condition is aimed to mimic the inlet condition of fluid entering the testing channel where a honeycomb mesh structure is imposed at the upstream to regularize the approaching flow. Denote the streamwise, transverse and perpendicular velocity components as u, v, and w,

respectively. Then we have, at the inlet, the boundary conditions of velocity and temperature are:

$$u = U_\infty, \quad v = w = 0, \quad T = T_0. \tag{2.6}$$

On the lateral walls and the cylinder surface, the "no-slip" boundary condition is employed for all velocity components. On the surface of the cylinder, the temperature is fixed at T_1. The lateral walls are assumed to be adiabatic where temperature remains constant T_0. Thus on the cylinder and lateral walls of the channel, we have:

$$u = v = w = 0, \quad T = \begin{cases} T_1 & \text{on the cylinder,} \\ T_0 & \text{on lateral walls.} \end{cases} \tag{2.7}$$

Since that the problem is transient, and vortices form behind the cylinder then pass through the channel to the outlet, the Neumann boundary conditions of velocity and temperature are applied on the outflow boundary:

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{n}} = \mathbf{0}, \quad \frac{\partial T}{\partial \boldsymbol{n}} = \mathbf{0}, \tag{2.8}$$

where $\boldsymbol{n}$ is the normal vector on the outlet.

Notice that the problem is symmetric to the central plane of the channel perpendicular to the cylinder. So the fluid flow is only simulated in the top half of the physical domain, which requires only half of the computational resources without loss of accuracy. For this purpose, a mirror boundary condition of velocity needs to be applied on the symmetry plane, which is equivalent to restricting the w component of velocity as 0. The temperature distribution is required to be normal to the symmetry plane as well. Thus the mirror boundary condition of velocity and temperature on the central plane are given by:

$$w = 0, \quad \frac{\partial T}{\partial \boldsymbol{n}_c} = \mathbf{0}, \tag{2.9}$$

where $\boldsymbol{n}_c$ is the normal vector on the central surface of the channel.

The thermal properties of water, density (ρ), dynamic viscosity (μ), thermal conductivity (k), and specific heat capacity (C_p) all vary with temperature T. In a small range of change of temperature, it is possible to assume a linear change of density with temperature while

Table 2.1 Coefficients of the polynomial fittings of the thermalphysical properties of water with temperature.

Thermalphysical property	a_0	a_1	a_2	a_3	a_4
Density (kg/m^3)	-1.67E+03	3.14E+01	-1.38E-01	2.69E-04	-2.01E-07
Specific heat ($kJ/kg \cdot K$)	3.15E+01	-3.24E-01	1.45E-03	-2.88E-06	2.15E-09
Thermal conductivity ($W/m \cdot K$)	8.148E-02	8.28E-04	6.50E-06	-8.34E-09	-9.39E-12
Viscosity ($N \cdot m/s^2$)	2.58E-01	-2.88E-03	1.22E-05	-2.30E-08	1.64E-11

keeping other three properties constants. This simplification is known as the Boussinesq approximation [Boussinesq (1897)], which then derives that the buoyancy force increases linearly with temperature. The Boussinesq equation has been applied for analyzing the wake structure behind a heated cylinder in cross-flow [Kieft et al. (2003)]. However, the largest temperature difference considered in the present study is as high as 61 °C, which makes the Boussinesq approximation invalid and thermally induced changes of the other three thermal properties other than density of water need to be incorporated into the simulation. We found that a 4th order polynomial ($\sum_{i=0}^{4} a_i T^i$) serves as sufficiently accurate interpolation for all four thermal properties of water with in the range of temperature from 285K to 370K, where temperature T is in the unit of Kelvin (K). The coefficients a_i's for all four thermal properties are listed in Table 2.1, and the plots of the interpolated curves versus the measurement data [Incropera and DeWitt (2002)] are given in Figure 2.2.

2.4 Numerical Methods

In this section we discuss the numerical method employed to simulate the present problem. Given numerical solutions $\boldsymbol{u}^n$, p^n, and T^n for velocity, pressure and temperature at time step t^n, we need to determine the solutions $\boldsymbol{u}^{n+1}$, p^{n+1}, and T^{n+1} for time step t^{n+1}. We adopt the finite element method for solving the numerical solutions for both the hydraulic Equations (2.1), (2.2), and the energy equation Equation (2.5). The streamline upwind Petrov-Galerkin (SUPG) method [Brooks and Hughes (1982)] is incorporated to avoid spurious oscillations of the numerical velocity solution field. However, Brooks and Hughes (1982) treated the pressure term with the penalty method, where pressure is expressed by the divergence of velocity thus only the velocity is actually being solved. This method requires the penalty term for pressure to be stored

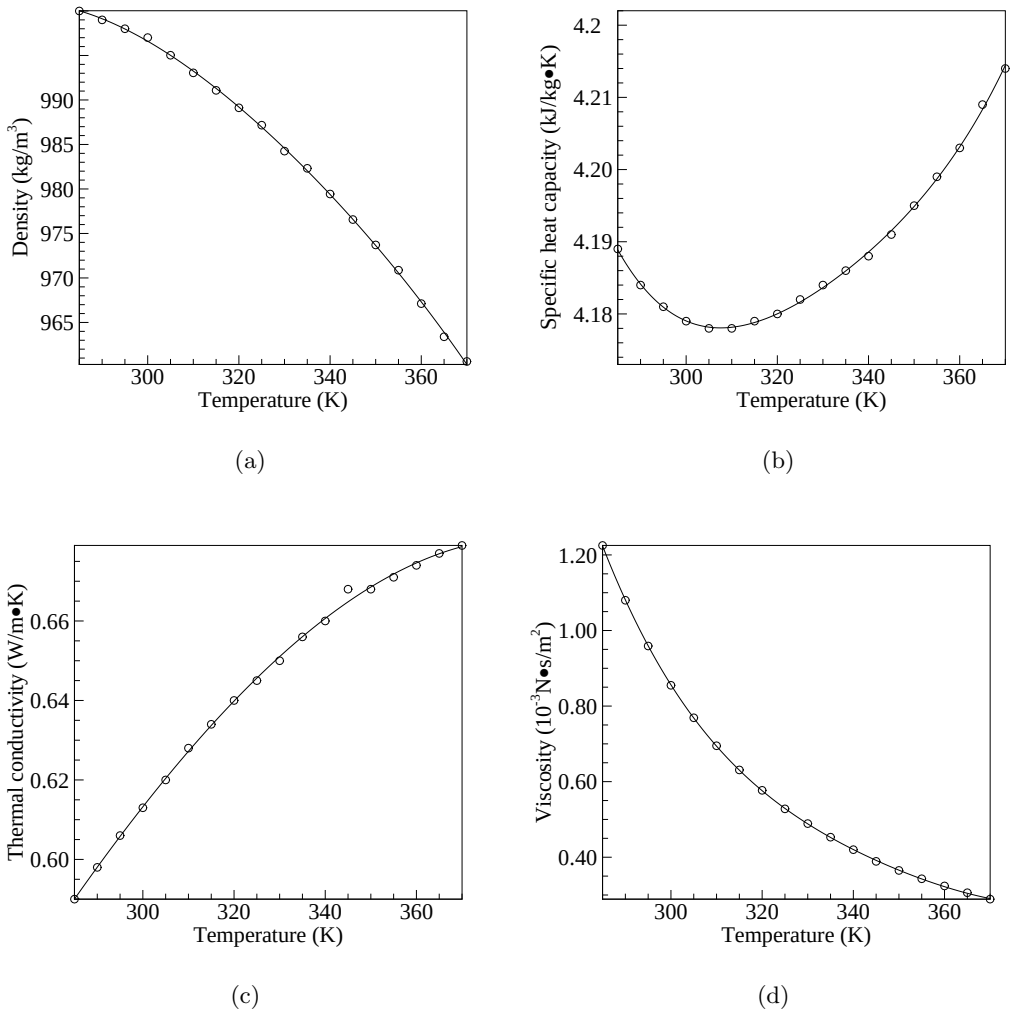


Figure 2.2 Thermalphysical properties of saturated water at different temperatures. '-': polynomial fitting; '∘': measurements [Incropera and DeWitt (2002)].

on a lower order element than that of velocity, which increases the difficulty in coding. Tezduyar et al. (1992) introduced the pressure stabilized Petrov-Galerkin (PSPG) method which allows the use of equal-order-interpolation velocity-pressure elements. This method has been successfully applied for solving fluid problems with various geometries [Jaeger et al. (2012); Amini et al. (2013)]. The shortcoming of such strategy is that this velocity-pressure integrated method is computationally inefficient in terms of computer memory and computing time, which tends to overcome its advantages such as accuracy and ease of handling for solving 3D large scale problems. A remedy for such issue is known as a family of velocity-pressure segregated methods, where the pressure is solved separately from the momentum equation, and velocity is then corrected by the updated pressure solution. In this study, we employ the four-step fractional method [Choi et al. (1997)] for solving the incompressible Navier-Stokes equations.

The four-step fractional method decouples the pressure from the momentum equation, and solves the pressure from a Poisson type equation. With this method, Equations (2.1) and (2.2) are solved by the following equations sequentially at each time step:

$$\rho(T^{n+1})\left(\frac{\hat{\boldsymbol{u}}-\boldsymbol{u}^n}{\Delta t}+\frac{1}{2}\left(\hat{\boldsymbol{u}}\cdot\nabla\hat{\boldsymbol{u}}+\boldsymbol{u}^n\cdot\nabla\boldsymbol{u}^n\right)\right)+\nabla p^n=\frac{\mu(T^{n+1})}{2}\nabla\cdot\left[\left(\nabla\boldsymbol{u}^n+(\nabla\boldsymbol{u}^n)^T\right)\right.$$
$$\left.+\left(\nabla\boldsymbol{u}^{n+1}+(\nabla\boldsymbol{u}^{n+1})^T\right)\right]-\left[\rho(T^{n+1})-\rho(T_0)\right]\boldsymbol{g}, \tag{2.10}$$

$$\rho(T^{n+1})\frac{\boldsymbol{u}^*-\hat{\boldsymbol{u}}}{\Delta t}=\nabla p^n, \tag{2.11}$$

$$\nabla\cdot\nabla p^{n+1}=\frac{\rho(T^{n+1})}{\Delta t}\nabla\cdot\boldsymbol{u}^*, \tag{2.12}$$

$$\rho(T^{n+1})\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^*}{\Delta t}=-\nabla p^{n+1}. \tag{2.13}$$

In the above equations, $\hat{\boldsymbol{u}}$ and $\boldsymbol{u}^*$ are the intermediate velocities which are solved based on pressure p^n from the previous time step. Pressure p^{n+1} is then updated by Equation (2.12) with the intermediate velocities. Notice that the intermediate velocities do not satisfy the incompressible condition, i.e. $\nabla\cdot\boldsymbol{u}^*\neq 0$. So the last step corrects the velocity with the latest

pressure field p^{n+1} to guarantee the incompressible condition. Here we also applied the Crank-Nicolson scheme on the convective term and diffusive term in the momentum equation to obtain second order accuracy in time discretization.

The temporal discretization of the energy Equation (2.5) is written as following,

$$\frac{T^{n+1}-T^n}{\Delta t}+\frac{1}{2}\left(\nabla\cdot\boldsymbol{u}^n\right)T^n+\frac{1}{2}\left(\nabla\cdot\boldsymbol{u}^{n+1}\right)T^{n+1}=\frac{\alpha(T^{n+1})}{2}\nabla\cdot\left[\nabla T^n+\nabla T^{n+1}\right], \tag{2.14}$$

where similarly, the Crank-Nicolson scheme is adopted on the convective and diffusive terms.

At each time step, Equations (2.10)-(2.14) are solved sequentially. However, notice that in Equation (2.10), temperature T^{n+1} is still unknown until being updated by Equation (2.14). To cope with this problem, we first start solving Equation (2.10) by using the temperature T^n from the previous time step. Then the intermediate velocity is obtained from Equations (2.10) and (2.11). With this intermediate velocity, pressure p^{n+1} is obtained from Equation (2.12). After this, velocity $\boldsymbol{u}^{n+1}$ is corrected from the pressure solution p^{n+1}. Finally, Equation (2.14) gives the temperature field T^{n+1} at time step t^{n+1}. The latest temperature solution T^{n+1} might be different with the guessed solution used for starting Equation (2.10). So it is necessary to repeat the above process for solving Equations (2.10)-(2.14) by starting with the latest solution to temperature, until the solution converges. The details of this iterative procedure is given by the flowchart shown in Figure 2.3. In the flowchart we see that the iteration stops when the relative change of temperature reaches below ε, where ε is a threshold controlling the convergence. Notice at least two iterations are required to perform this comparison. This method is also known as the "block-iterative" approach for solving coupled equations [Tezduyar and Sathe (2007)]. There is no general theory on the convergence condition for such method [Cervera et al. (1996)]. Considering the iteration is required at each time step, the value of ε should be within the order of Δx^2 to avoid introducing more numerical error from this scheme, regarding the second order spatial convergence rate of the finite element formula, e.g. take $\varepsilon = 1\times10^{-3}$ when Δx has characteristic value of 1×10^{-1}. We have observed that the convergence is typically reached within two or three iterations, which demonstrates that this strategy is computationally efficient. Also, without loss of accuracy, we evaluate the term $\alpha(T^{n+1})$ in Equation (2.14) by

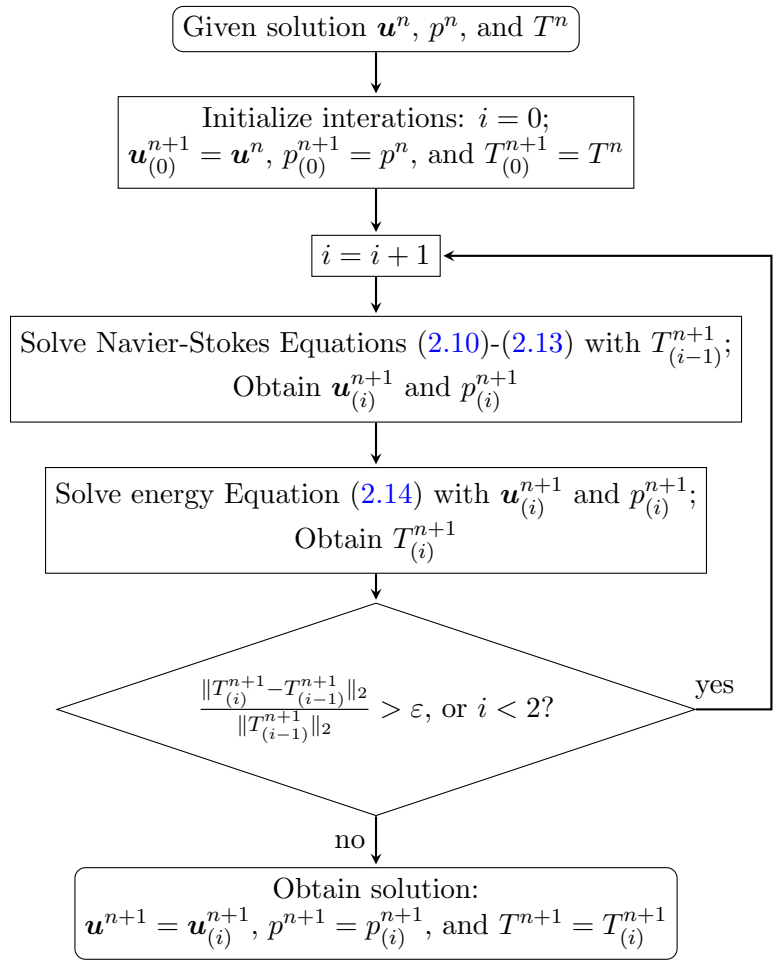


Figure 2.3 Flowchart of the iteration solver for the semi-coupled Navier-Stokes and heat convection/diffusion equation system.

using the latest temperature solution instead of treating it as an unknown parameter, to avoid the unnecessary effort of introducing a nonlinear term in the energy equation.

The weak form of equations are obtained from multiplying the governing equations with a weighting function $\boldsymbol{w}$ and integrating over the physical domain Ω, where $\boldsymbol{w} \in \mathcal{H}^1$, $\mathcal{H}^1$ is the Soblev space of vector functions on Ω. In this study, we also incorporate the SUPG terms into the weak form equations to stabilize the solution. Then the finite element method tends to give the approximate solutions to velocity and pressure in a finite dimensional subspace $\mathcal{H}_h$ of space $\mathcal{H}^1$. Then the numerical procedure can be described as follows: given approximate solutions, $\boldsymbol{u}_h^n$, p_h^n, and T_h^n at time step n, for any test functions $\boldsymbol{w}_h \in \mathcal{H}_h$, $q_h \in \mathcal{H}_h$, and $s_h \in \mathcal{H}_h$, find the

approximate solutions $\boldsymbol{u}_h^{n+1}$, p_h^{n+1}, and T_h^{n+1} at time step t^{n+1} satisfying the following weak form of equations:

$$\begin{aligned}
&\int_\Omega \boldsymbol{w}_h \cdot \left\{ \rho(T_h^{n+1}) \left(\frac{\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h^n}{\Delta t} + \frac{1}{2}\boldsymbol{u}_h^n \cdot \nabla \boldsymbol{u}_h^n + \frac{1}{2}\hat{\boldsymbol{u}}_h \cdot \nabla \hat{\boldsymbol{u}}_h \right) + \nabla p_h^n + \left[\rho(T_h^{n+1}) - \rho(T_0)\right] \boldsymbol{g} \right\} \mathrm{d}\Omega \\
&+ \int_\Omega \frac{\mu(T_h^{n+1})}{2} \nabla \boldsymbol{w}_h : \left[\left(\nabla \boldsymbol{u}_h^n + (\nabla \boldsymbol{u}_h^n)^T \right) + \left(\nabla \hat{\boldsymbol{u}}_h + (\nabla \hat{\boldsymbol{u}}_h)^T \right) \right] \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \boldsymbol{\delta}(\boldsymbol{w}_h) \cdot \left\{ \rho(T_h^{n+1}) \left(\frac{\hat{\boldsymbol{u}}_h - \boldsymbol{u}_h^n}{\Delta t} + \frac{1}{2}\boldsymbol{u}_h^n \cdot \nabla \boldsymbol{u}_h^n + \frac{1}{2}\hat{\boldsymbol{u}}_h \cdot \nabla \hat{\boldsymbol{u}}_h \right) + \nabla p_h^n \right. \\
&\left. + \left[\rho(T_h^{n+1}) - \rho(T_0)\right] \boldsymbol{g} \right\} \mathrm{d}\Omega - \int_{\Gamma_2} \boldsymbol{w_h} \cdot \boldsymbol{h} \mathrm{d}\Omega = 0,
\end{aligned} \tag{2.15}$$

$$\int_\Omega \boldsymbol{w_h} \cdot \left[\rho(T_h^{n+1}) \frac{\boldsymbol{u}_h^* - \hat{\boldsymbol{u}}_h}{\Delta t} - \nabla p_h^n \right] \mathrm{d}\Omega = 0, \tag{2.16}$$

$$\frac{\Delta t}{\rho(T_h^{n+1})} \int_\Omega \nabla q_h \cdot \nabla p_h^{n+1} \mathrm{d}\Omega - \int_\Omega \nabla q_h \cdot \boldsymbol{u}_h^* \mathrm{d}\Omega + \int_\Gamma q_h \hat{\boldsymbol{u}}_h \cdot \boldsymbol{n} \mathrm{d}\Gamma = 0, \tag{2.17}$$

$$\int_\Omega \boldsymbol{w_h} \cdot \left[\rho(T_h^{n+1}) \frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^*}{\Delta t} + \nabla p_h^{n+1} \right] \mathrm{d}\Omega = 0, \tag{2.18}$$

and the stabilized finite element formula of the energy equation:

$$\begin{aligned}
&\int_\Omega s_h \cdot \left(\frac{T_h^{n+1} - T_h^n}{\Delta t} + \frac{1}{2}(\nabla \cdot \boldsymbol{u}_h^n) T i_h^n + \frac{1}{2}(\nabla \cdot \boldsymbol{u}_h^{n+1}) T_h^{n+1} \right) \mathrm{d}\Omega \\
&+ \int_\Omega \frac{\alpha(T_h^{n+1})}{2} \nabla s_h \cdot \left(\nabla T_h^n + \nabla T_h^{n+1} \right) \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \delta(s_h) \cdot \left(\frac{T_h^{n+1} - T_h^n}{\Delta t} + \frac{1}{2}(\nabla \cdot \boldsymbol{u}_h^n) T i_h^n + \frac{1}{2}(\nabla \cdot \boldsymbol{u}_h^{n+1}) T_h^{n+1} \right) \mathrm{d}\Omega = 0,
\end{aligned} \tag{2.19}$$

where $(\cdot)_h$ denotes the approximate solution in the finite dimensional subspace $\mathcal{H}_h$, Ω^e's, e=1, ..., n_{el}, are the elements with n_{el} being the total number of spatial discretization. Vector $\boldsymbol{\delta}^h(\cdot)$ is the SUPG stabilization term [Brooks and Hughes (1982)], which is applied on both Equation (2.15) of momentum and Equation (2.19) of energy. Equation (2.17) is the finite element formula for the pressure Equation (2.12), which contains the integration of the approximate intermediate velocity vector $\hat{\boldsymbol{u}}_h$ on the Neumann boundary Γ (i.e. the outlet of the channel) of domain Ω. Since that Equation (2.15) does not ensure that $\hat{\boldsymbol{u}}_h$ is divergence free, it is necessary

to add restriction to $\hat{\boldsymbol{u}}_h$ to satisfy the incompressibility condition. Here at each solution iteration, we calculate the mass flow rate of the fluid on the outlet Γ, and compare this quantity with the inlet mass flow rate, then calibrate the intermediate velocity vectors on the outlet to make the two flow rates equal. We have found that this method is very easy to operate and successfully guarantees that the final velocity solution is divergence free.

The 3D mesh for the simulation is generated by stretching an unstructured 2D grid along the direction of the cylinder, where the 2D grid is created using the CUBIT mesh generation toolkit [Shepherd et al. (2000)]. Details of the mesh are shown in Figure 2.4. The 2D grid is divided into three sub-domains as shown in Figure 2.4(a), where different element sizes are assigned. In sub-domain (1), within the half diameter distance surrounding the cylinder, we assign the smallest element size, since this section contains the boundary layer and largest temperature gradient. Denote the element size on the outer circular boundary of sub-domain (1) as Δx. Extend sub-domain (1) by $1D$ distance and backwards by $10D$ distance, we get sub-domain (2), where elements have average size $2\Delta x$, aimed at accurately capturing the behavior of the shedding vortices and evolution of temperature field. In the outer region, sub-domain (3), element size is further doubled to help reduce the total computational cost, since this section has relatively less effect on the near wake behavior. The element size is reduced further along the axial direction in sub-domain (1), and is clustered towards the cylinder (Figure 2.4(b)), to ensure the accuracy of solution inside and around the boundary layer. Comparison of the element sizes in these three sub-domains is clearly shown by Figure 2.4(c). Then the 3D mesh can be easily formed by stretching the 2D grid along the direction of the cylinder (only the top half of channel is required due to symmetry in geometry), as seen in Figure 2.4(d).

Notice that Equation (2.15) is a non-linear equation. Non-linear equation can be solved by the exact Newton-Raphson method. For this study we use the SNES C++ toolkit provided by the open source package PETSc [Balay et al. (2013b,a, 1997)], which contains convenient and powerful tools to solve nonlinear equations. At each iteration, the linear equation is solved by the BCGS solver and basic ASM pre-conditioner which are available in PETSc. The simulations in this paper are performed on CyEnce at Iowa State University and on TACC Stampede.

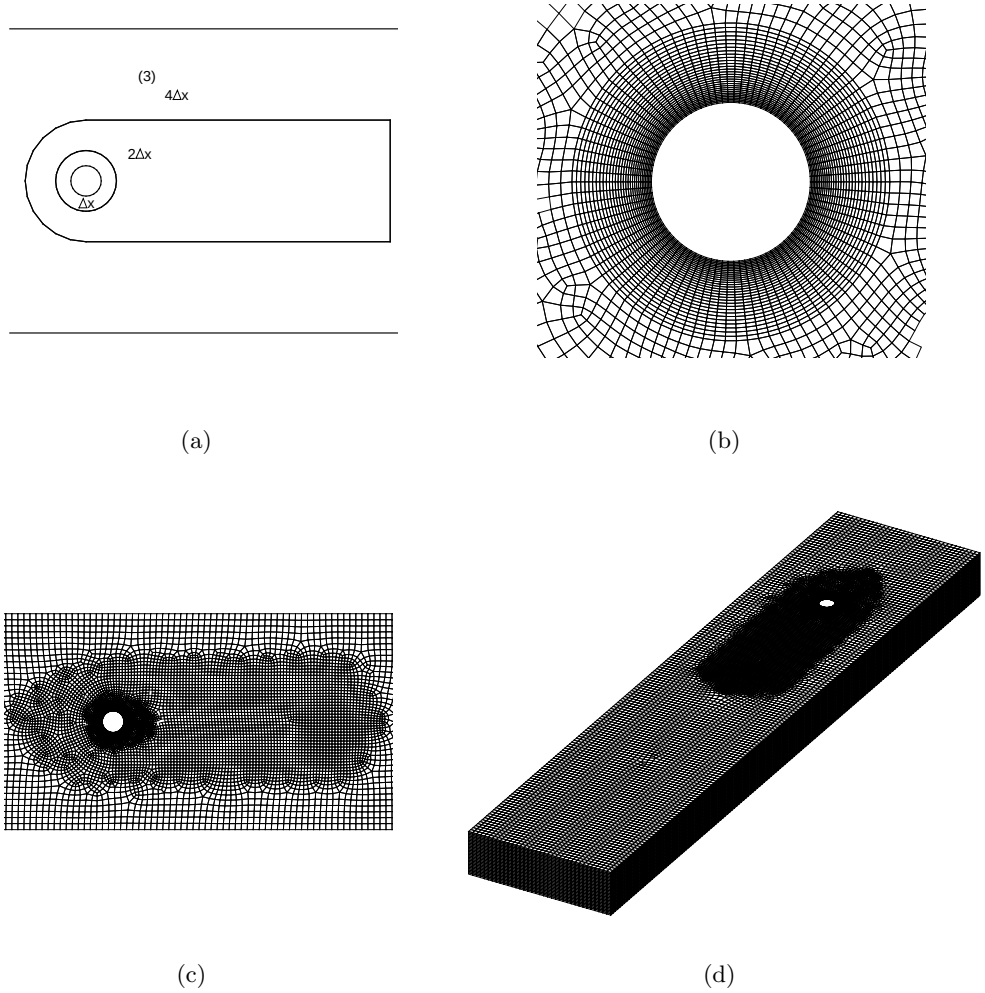


Figure 2.4 3D mesh used for simulations and related details. (a) Sections of the 2D slice of the geometry with different spatial resolutions; (b) detail of the spatial discretization around the cylinder; (c) 2D spatial discretizations in the three sections on the 2D slice of geometry; (3) the 3D mesh stretched from the 2D mesh.

2.5 Convergence tests and validation

In this section we report the convergence tests on the numerical method, as well as the validation of our methodology by comparing with previous experimental results. Both the spatial and temporal convergence tests are obtained from 2D simulations.

To perform the spatial convergence tests, we assign four values to the mesh size Δx, $\Delta x/D = 2\pi/45\,(\approx 0.14)$, $2\pi/90\,(\approx 0.07)$, $2\pi/180\,(\approx 0.035)$, and $2\pi/360\,(\approx 0.017)$, as giving the coarse mesh, the medium mesh, the fine mesh, and a reference mesh with highest resolution for calculating the relative errors of spatial convergence. The above choices of Δx's are obtained by dividing sub-domain (1) circumferentially into equal parts by every 8,4, $2, and 1$. The time step increment is kept at $\Delta t = 0.0125$ dimensionless time scaled by D/U_∞.

Spatial convergence test results are given by Figure 2.5. Figure 2.5(a) plots the ensemble-averaged streamwise velocity profiles along the wake centerline segment in the range of $[0,\ 10D]$ for all spatial resolutions. Each velocity profile is average for all time steps from dimensionless time 100 to 200, along the centerline pointed out in Figure 2.1. The ensemble-averaged streamwise velocity profiles along the wake centerline reflects the flow structure in the wake behind the cylinder, where the negative part of the profile implies a recirculation region in the wake. We see in Figure 2.5(a), as the spatial resolution increases, the profile clearly converges. To quantitatively investigate the convergence rate of the numerical method on this problem, we plot the relative error of spatial resolution as a function of Δx in Figure 2.5(b). The relative error of each spatial resolution Δx reflects the relative difference of the velocity profile under Δx from the reference case with highest spatial resolution in Figure 2.5(a), and is defined as:

$$\mathrm{err}_{u_C}^{\Delta x} = \frac{\|u_C^{\Delta x} - u_C^{\mathrm{ref}}\|_2}{\|u_C^{\mathrm{ref}}\|_2} = \frac{\left[\int_0^{10D} \left(u_C^{\Delta x}(x) - u_C^{\mathrm{ref}}(x)\right)^2 \mathrm{d}x\right]^{1/2}}{\left[\int_0^{10D} \left(u_C^{\mathrm{ref}}(x)\right)^2 \mathrm{d}x\right]^{1/2}}, \tag{2.20}$$

where $\|\cdot\|_2$ indicates the L_2 norm, $u_C^{\Delta x}(\cdot)$ denotes the velocity profile for spatial resolution Δx, and $u_C^{\mathrm{ref}}(\cdot)$ refers to the reference case. Figure 2.5(b) clearly shows a second order spatial convergence rate of the current numerical method.

The temporal convergence behavior is studied by applying the simulation on the fine mesh ($\Delta x/D = 2\pi/180$) by varying the dimensionless time step increment Δt. Here we still adopt

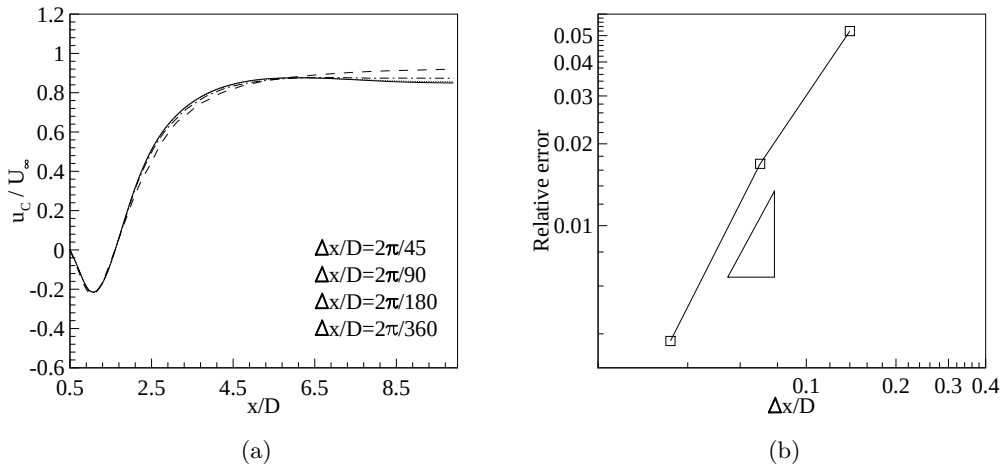


Figure 2.5 Spatial convergence results. (a) ensemble-averaged centerline velocity plotted for all spatial resolutions; (b) convergence errors plotted in log-log format.

the ensemble-averaged streamwise velocity profiles along the wake centerline to perform the investigation on the temporal convergence. As for the study on the spatial convergence, the velocity profile is considered on the line segment between 0 to $10D$. However, here the velocity profile is averaged by all the time steps between dimensionless time 0 and 200 to incorporating all the accumulated errors from temporal discretization. In this study, we consider four dimension less time step increment Δt, $\Delta t = 0.1$, 0.05, 0.025, and 0.0125 as the reference case for calculating the relative error.

Figure 2.6 gives the results of the temporal convergence tests. Figure 2.6(a) plots the ensemble-averaged streamwise velocity profiles for all the four temporal resolutions. As clearly shown by the figure, we see the curves approach a limit as Δt decreases. Figure 2.6(b) shows the relative error evolution with temporal resolution, where the relative error of the Δt is defined in the same way as that given by Equation (2.20) by replacing Δx with Δt. From the figure we see an obvious second order convergence rate with Δt, which demonstrates the second order temporal discretization scheme deployed in Equation (2.10).

The validation of the 3D solver is also performed by comparing the numerical results with the experimental measurements [Hu and Koochesfahani (2011)]. The aspect ratio ar is kept as 6.3, which gives the same geometrical configuration of the test section as used in the ex-

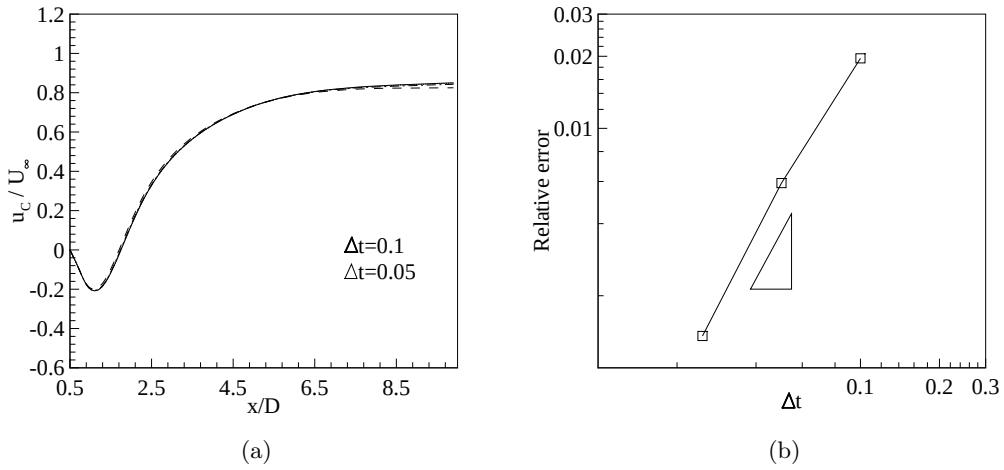


Figure 2.6 Temporal convergence results. (a) ensemble-averaged centerline velocity plotted for all temporal resolutions; (b) convergence errors plotted in log-log format.

periment. Three temperatures on the wall of the cylinder are considered for the validation, $T_1 = 24\,^{\circ}\mathrm{C}$ (297 K), 66 °C (339 K), and 85 °C (358 K), where the channel wall temperature T_0 is always fixed at 24 °C (297 K) thus the first test case corresponds to the flow past an unheated cylinder. The corresponding Richardson numbers Ri's for these three cases are 0, 0.72, and 1.04. Comparison between the simulations and experimental data is given by Figure 2.7. The comparison of the ensemble-averaged streamwise velocity profiles is shown in Figure 2.7(a). The figure clearly shows that flows pasted heated cylinder has quite different behaviors compared with those of the flow pasted unheated cylinder. The most significant difference is that the heated cylinder extends the recirculation region farther downstream, which is reflected by the velocity profile with a wider negative part for the heated cylinder. This phenomenon has been accurately captured by the simulations. The wake closure length l_C of the recirculation region is a characteristic quantity for describing the near wake structure behind the cylinder, which can be read from the centerline velocity profile as the distance from the cylinder center to the point where the curve changes its sign. For the unheated cylinder, both the experiment and simulation give $l_C/D \approx 3$. However, for the heated cases (both $Ri = 0.72$ and 1.04), the wake closure length l_C/D jumps to about 9, which is also captured by the simulations but with some slight shifts. This jump of l_C with cylinder wall temperature is not observed for 2D simulations.

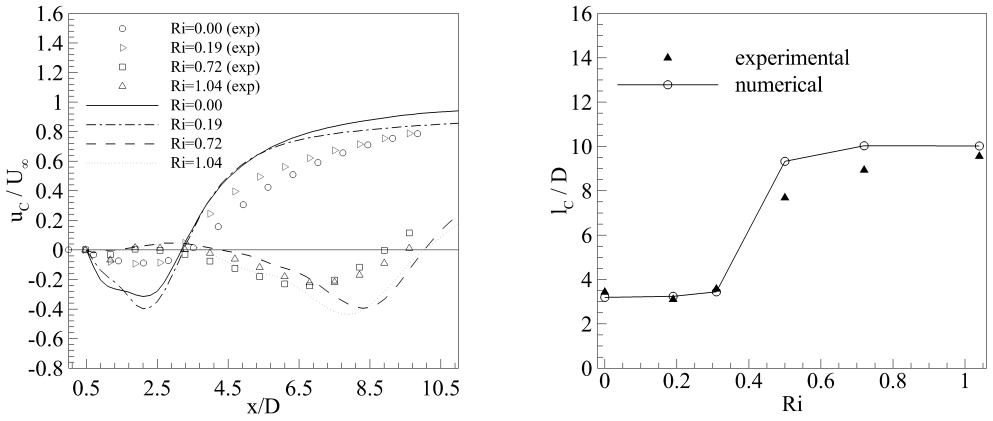
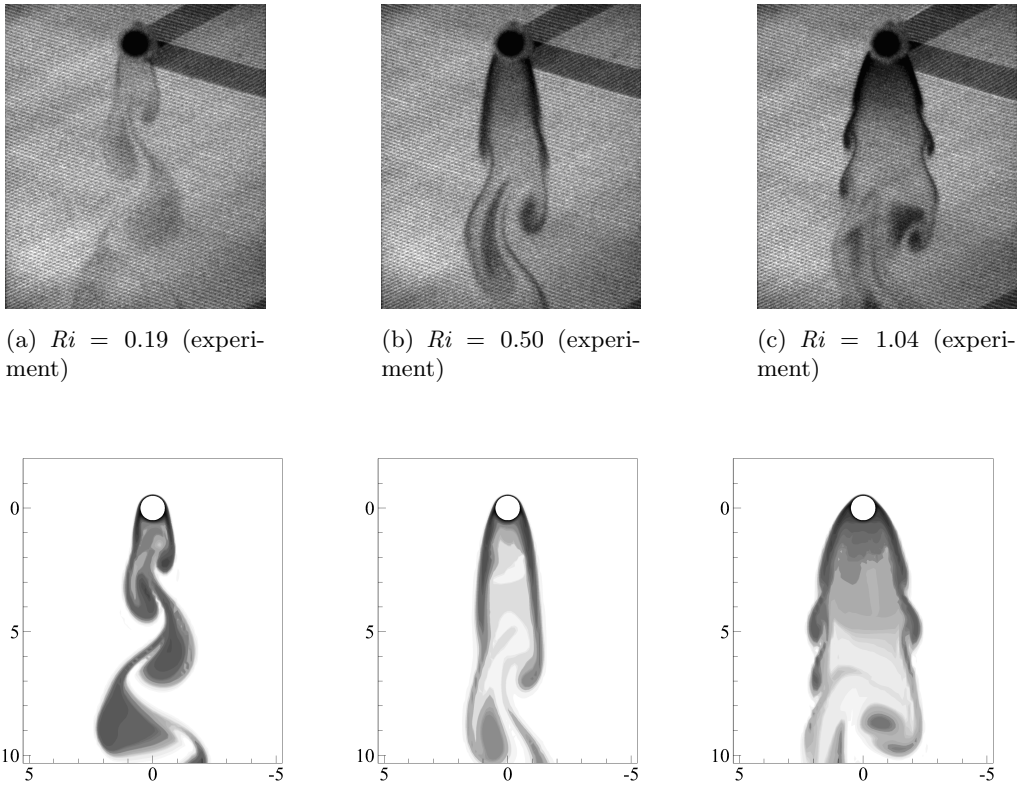


Figure 2.7 Comparisons between the numerical results and the experimental measurements. (a) comparison between our numerical results with experimental data on the ensemble-averaged centerline velocity on the symmetric plane at different Richardson numbers; (b) The comparison of wake closure length at different Richardson numbers.

It is also noticeable that the numerical results give stronger streamwise recirculation velocity compared with the experiment. To better understand how the wake structure responses to the cylinder wall temperature, the relation between l_C and Ri is plotted in Figure 2.7(b). The plot reveals that the evolution of the wake closure length with increasing cylinder wall temperature is very well captured by the numerical simulation when comparing with the experimental data, where both the simulation and experiment show that l_C stays around $3D$ for relatively smaller Ri's and jump to about $9D$ when Ri reaches above 0.5.

Another major effect caused by the 3D heated cylinder is the change of the vortex shedding pattern in the near wake behind the cylinder. This effect has been observed in the experiment and very well described [Hu and Koochesfahani (2011)]. When relatively lower temperature ($Ri \leqslant 0.3$) is applied on the cylinder, the vortex shedding pattern is more like a "von-Karman" vortex street. As the temperature increases but not too high ($Ri \sim 0.5$), a "dead flow" zone appears behind the cylinder where the flow is almost quiescent inside. However, for very high temperature on the cylinder ($Ri \sim 1.0$), the vortex shedding pattern adapts a "Kelvin-Helmholtz" like structure, with a more widely opened zone behind the cylinder. All these



(a) $Ri = 0.19$ (experiment) (b) $Ri = 0.50$ (experiment) (c) $Ri = 1.04$ (experiment)

(d) $Ri = 0.19$ (simulation) (e) $Ri = 0.50$ (simulation) (f) $Ri = 1.04$ (simulation)

Figure 2.8 Comparison of the vortex shedding patterns under different temperatures between experimental observations [Hu and Koochesfahani (2011)] (top) and numerical simulations (bottom).

characteristic vortex shedding patters have been captured by our simulations, which give very good agreement to experimental snapshots, as shown by Figure 2.8. Later part in this paper, we will provide more details on the investigation of the flow transition and the vortex shedding pattern change, especially when considering the effect of cylinder aspect ratio ar.

2.6 3D flow transition

One significant feature of the flow behind a heated cylinder is the existence of a flow transition. This phenomenon has been observed in previous studies. Flow transition always happens with some discontinuous shift in the measurement of some parameters as the controlling parameter smoothly changes. Williamson (1988) first found there is a discontinuity in the curve

of Strouhal-Reynoulds relationship (in his work on flow past an unheated cylinder). When the cylinder is imposed as a heat source, as mentioned before, Hu and Koochesfahani (2011) figured out that the discontinuity appearing in the distribution of ensemble-averaged centerline mean velocities clear reflects the existence of three dimensional flow transition as cylinder temperature changes. Our numerical method has also accurately captured this behavior (Figure 2.7). In Figure 2.9(a), we reconsider the same situation as shown in the validation, but plot the mean centerline velocities at denser sampling points of the Richardson number Ri. From the plot, we find that the transition is discontinuous, which is saying, as cylinder temperature increases, the recirculation zone of the mean velocity curve does not gradually move from very near wake ($\sim 3D$) to far wake ($\sim 9D$), instead is distinguished by two groups. The flow transition is triggered by the heat source on the cylinder, which has very different behavior compared with the unheated case.

The flow is performed in a closed channel where the cylinder aspect ratio is relatively small. The "no-slip" flow condition applied on the channel walls gives a significant uneven, parabolic shaped velocity profile on the channel cross-section, which generates a velocity gradient along the cylinder axis direction. In this part, we investigate if this velocity gradient is another prerequisite condition along with the cylinder temperature for the occurrence of the flow transition. If yes, we hope to understand how the cylinder aspect ratio ar affects the wake behind the cylinder. For this purpose, we performed the simulations for cylinder aspect ratios $ar = 5.0$ and $ar = 4.0$, and repeated the analysis as we did for the $ar = 6.3$ case. The simulation was also performed on the case with "no-velocity-gradient" condition, where "slip" condition is applied on the upper and central surface of the channel ("no-slip" condition is still valid for the other two lateral walls, and here the geometry ar is 6.3 for the mesh), to mimic the flow past an infinitely long cylinder, defined as $ar = \infty$. The centerline velocity curves for these cases are listed in Figure 2.9.

From Figure 2.9, we find that the flow transition exists for all the finite aspect ratios (i.e. $ar = 4.0$, 5.0, and 6.3). Before the flow transition is triggered (for low Ri's), the cylinder aspect ratio has very limited effect on the flow wake structure for all the three cases with finite ar's. This implies that the velocity gradient along the cylinder axis direction does not

influence the flow wake structure before the flow transition is triggered. However, the cylinder aspect ratio does influence the wake structure when the flow transition is triggered. For $ar = 6.3$ (Figure 2.9(a)), the mean velocity curves for the transited flows give a wake closure length about $9D$. As ar drops to 5.0 (Figure 2.9(b)), the velocity gradient along the cylinder axis direction increases, and the recirculation region of the transited flow is slightly pushed downstream. When ar decreases to 4.0 (Figure 2.9(c)), the recirculation zone for the transited flow grows even longer with the wake closure length about $14D$. This reflects a trend, that the cylinder aspect ratio ar (as well as the induced velocity gradient along the cylinder axis direction) has less effect on the wake structure for the transited flow as ar increases. It also implies that as $ar \to \infty$, the flow transition will eventually disappear, where the transited flows have similar wake structure as the untransited flows do. This hypothesis is verified by the Figure 2.9(d), where we see that all the mean velocity curves gather in one group with a wake closure length of about $2D$. Thus we conclude that the flow transition is triggered by the increasing heat input into the flow system, but needs the existence of a velocity gradient along the cylinder axis direction to show significant difference with the regular wake structure.

This trend of flow transition with different cylinder aspect ratios is also illustrated by the wake closure length plot with Richardson number at all ar's in Figure 2.10. Shown clearly in the plot, there exists a jump of the wake closure length l_C for relatively small ar ($ar = 4.0$), where on the other hand the velocity gradient along the cylinder axis direction is very large. As ar decreases to 5.0, we find from the plot that the jump of the wake closure curve still exists, but is less significant than that of $ar = 4.0$, with entirely smaller l_C's for large Ri's. The situation of $ar = 6.3$ is similar to that of $ar = 5.0$, but with a even lower position of the curve shown in the plot. We also see that as ar increases, the flow transition tends to disappear in an increasing rate. When the velocity gradient is totally removed by increasing the aspect ratio to infinity, the l_C is eventually reaches its limit as a flat position, which indicates the vanishing of flow transition. This analysis further supports our hypothesis that velocity gradient long the cylinder axis direction is a crucial influencing factor for the occurrence of flow transition.

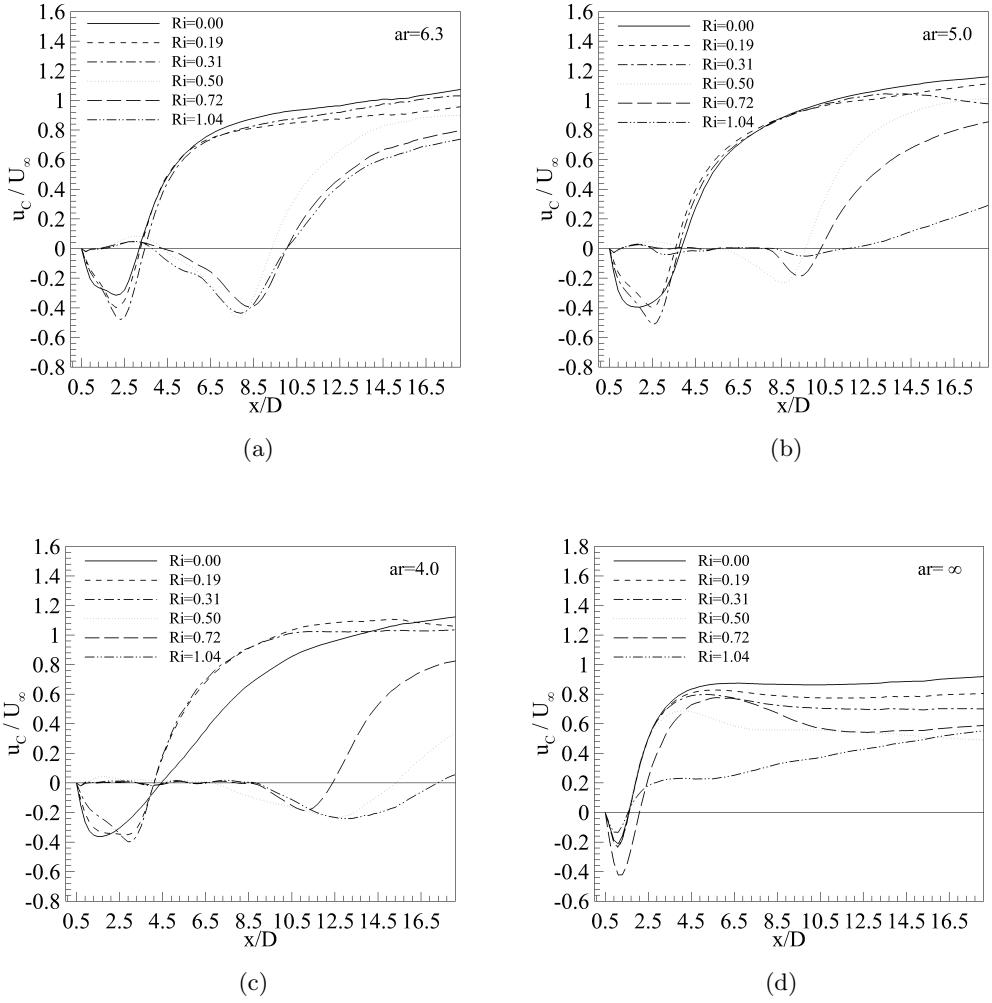


Figure 2.9 Ensemble-averaged centerline velocity plots at different Richardson numbers for aspect ratios of 6.3, 5.0, 4.0 and ∞.

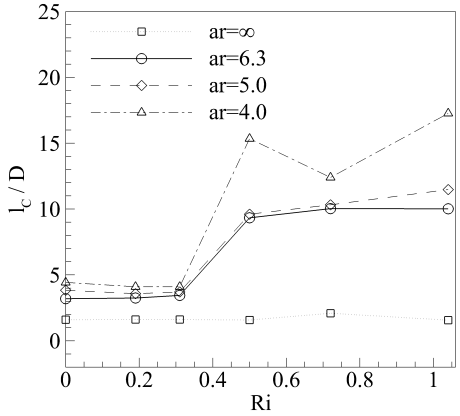
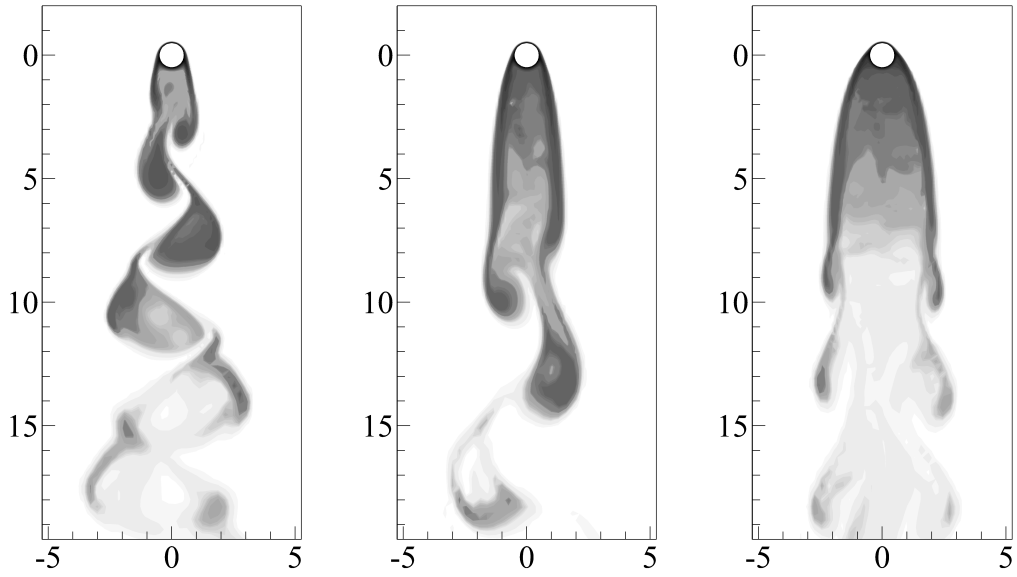


Figure 2.10 Wake closure length plotted for all Richardson numbers at aspect ratio of 4.0, 5.0, 6.3 and ∞.

2.7 Vortex shedding pattern transition

Another major feature found for the 3D flow past a heated cylinder is that there exists a transition in the vortex shedding patterns, which has been pointed out in Section 2.5. Briefly recall the facts, as $ar = 6.3$, that a smaller Richardson number tends to give a staggered vortex shedding pattern, while very large Richardson number forms a simultaneous vortex shedding patter, where in between, a extended wake behind the cylinder with almost quiescent flow inside formed in a very short range of Richardson number. As introduced before, Hu and Koochesfahani (2011) category these modes as "vor-Karman" typed, "dead-flow zone" typed, and "Kelvin-Helmholtz" typed patterns. Since we have verified that the velocity gradient crucially affects the flow transition in the wake behind the flow past a heated cylinder, it will be interesting to demonstrate if the vortex shedding pattern transition also requires the existence of such a velocity gradient. To answer this question, we investigate the vortex shedding patterns for $ar = 5.0$, $ar = 4.0$, as well as $ar = \infty$.

Figure 2.11 shows the snapshots of the temperature fields on the central plane at $Ri = 0.19$, $Ri = 0.50$, and $Ri = 1.04$ for the case of $ar = 5.0$. The figure clearly illustrates the three typical vortex shedding patterns as those exist in the case of $ar = 6.0$. Here we see, for a very small Ri (e.g. $Ri = 0.19$, representing lower temperature difference between the cylinder and incoming

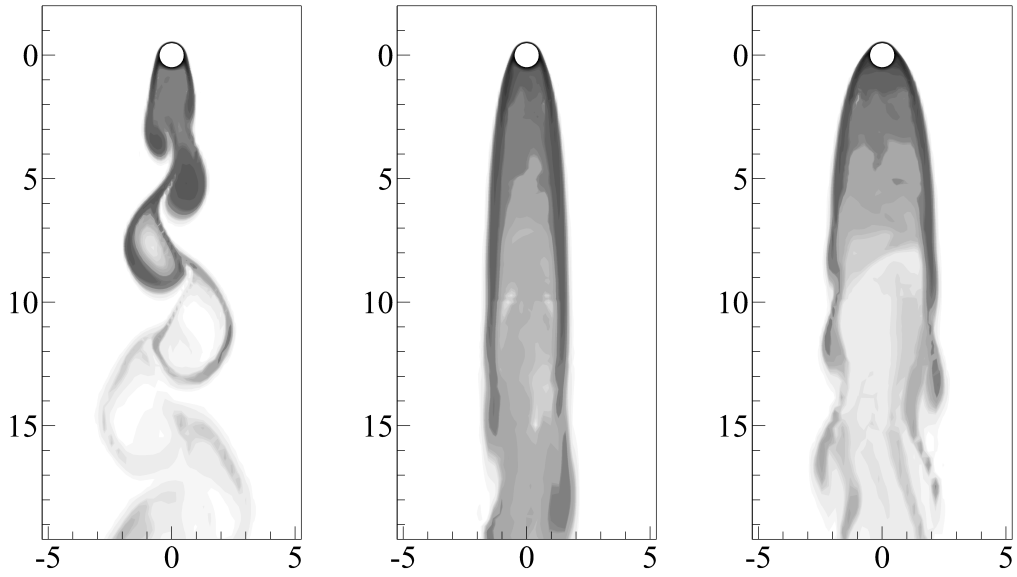


(a) $Ri = 0.19$ (b) $Ri = 0.50$ (c) $Ri = 1.04$

Figure 2.11 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = 5.0$.

flow), the wake presents a clear staggered shedding pattern as that of the case with unheated cylinder. For a medium Ri (e.g. $Ri = 0.50$), the staggered pattern in the near wake vanishes and is replaced by a "dead flow" zone similar to that of $ar = 6.3$. However, compared to the case of $ar = 6.3$, the "dead flow" zone appears much longer in the current scenario, which has a steeper velocity gradient along the cylinder axis direction. For a large Ri (e.g. $Ri = 1.04$), the simultaneous vortex shedding pattern turns to be significant, where we see symmetric small side vortices pairs along the widely opened near wake behind the cylinder.

Figure 2.12 plots the snapshots of the temperature field on the central plane at the same sampled Ri's (0.19, 0.50 and 1.04) for $ar = 4.0$, which gives a even steeper velocity gradient along the cylinder axis direction. In this scenario, the vortex shedding pattern of a very small Ri shows no difference with the staggered ones behind the cylinder with no heat and very small Ri's in other scenarios. Notice that both the width and length of the recirculation zone appear no difference compared with those of very small Ri's in other scenarios. However, for a medium Ri, e.g. 0.50 in the figure, though the width of the wake does not differ from those of $ar = 0.50$

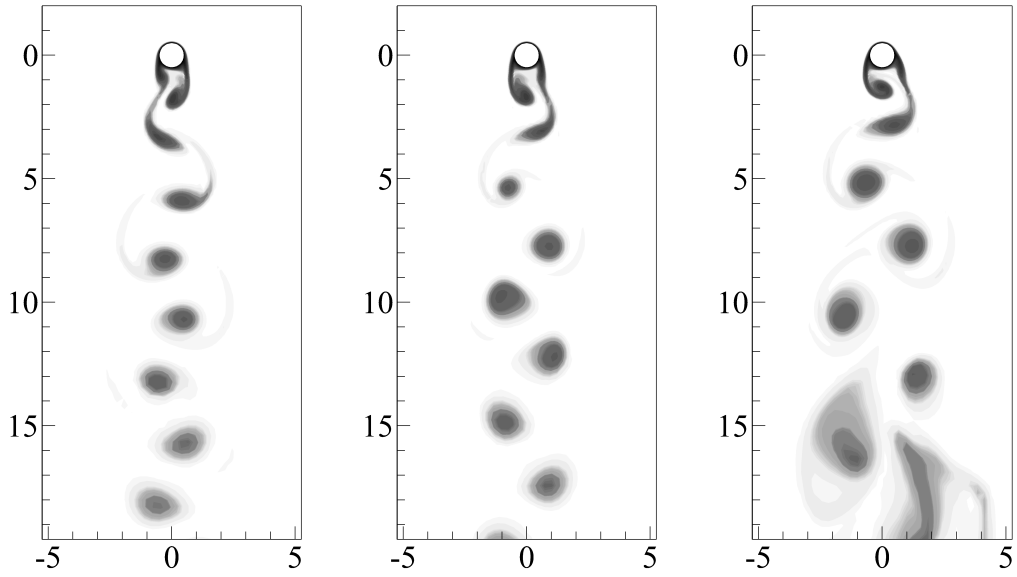


(a) $Ri = 0.19$ (b) $Ri = 0.50$ (c) $Ri = 1.04$

Figure 2.12 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = 4.0$.

and $ar = 6.3$, the recirculation zone is extremely extended streamwise. This phenomenon is similarly found for the very large Ri, e.g. $Ri = 1.04$, where the simultaneous shedding pattern appears with almost the same width as other scenarios but with an extended recirculation zone.

To investigate the effect of velocity gradient on the vortex shedding pattern transition, we performed the analysis on the case of $ar = \infty$. From the previous analysis on $ar = 4.0$, 5.0, and 6.3, we have observed that as ar increases (where the velocity gradient along the cylinder axis direction decreases), the recirculation regions in both the "dead-flow zone" typed pattern and the "Kelvin-Helmholtz" typed pattern become shorter. This indicates a trend that as the velocity gradient along the cylinder axis direction vanishes, the "dead-flow zone" and the simultaneous pattern will eventually be eliminated. This hypothesis is demonstrated by Figure 2.13. In the plot we clearly see that as Ri increases (from 0.19 to 1.04), the wake behind the cylinder gains an increasing width, but neither the "dead-flow" zone nor the vortex pattern changes. In all the three illustrations, we observe very typical staggered vortex shedding pattern



(a) $Ri = 0.19$ (b) $Ri = 0.50$ (c) $Ri = 1.04$

Figure 2.13 Vortex shedding patterns illustrated by the snapshots of the thermal fields for Richardson number of 0.19, 0.50 and 1.04 at $ar = \infty$.

only, as noted as the "von-Karman" vortex street. This finding indicates that the existence of the velocity gradient along the cylinder axis direction is a prerequisite for the occurrence of the vortex shedding pattern transition, where the larger the gradient is, the bigger the "dead-flow" zone.

We have visually analyzed the vortex shedding pattern transition, where the wake behind the cylinder opens up as cylinder temperature increases, and the vortex shedding pattern changes from staggered mode to simultaneous mode. These observed vortex shedding patterns have obvious different vortex structures presented on the central plane of the channel, which implies that it is possible to quantitatively detect the vortex shedding patterns by measuring the vorticity level on the central plane. Here we define the vorticity level, $|V_C|$, by the integration of the magnitude of vorticity in the cylinder axis direction (perpendicular to the central plain) over the whole central plane Ω_C:

$$|V_C| = \int_{\Omega_C} |\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}| \, \mathrm{d}\Omega. \tag{2.21}$$

The central plane vorticity levels at all simulated Ri for each ar are given in Figure 2.14. For very small ar, e.g. $ar = 4.0$, we clearly see that V_C forms into two segments with different slopes. The segment for lower Ri's represents the state of the staggered vortex shedding pattern; the segment for higher Ri's represents the state of the simultaneous vortex shedding pattern. The discontinuous transiting region in between represents the state of the "dead-flow" zone typed shedding pattern. We see that the transiting state exists only in a narrow range of Ri, which indicates a quick transition between the staggered pattern and the simultaneous pattern. The situations for $ar = 5.0$ and $ar = 6.3$ are similar to the case of $ar = 4.0$. But we can observe the trend that the two segments in each vorticity level plot gradually gradually converge to the same slope as ar increases, but still stay in two disconnecting lines when ar is still finite. However, the two segments of vorticity level curves fall into a single linear curve in the case of $ar = \infty$, which indicates the vanishing of the vortex shedding pattern transition. This analysis of central plane vorticity level gives strong agreement with our visual observations discussed before.

2.8 Conclusions

In this paper we have investigated features of the wake in flow past a heated cylinder. The traditional Boussinesq approach has been replaced with an exact fluid-thermal coupled equation system to obtain more accurate simulation results for a wide range of temperature difference between the cylinder surface and the incoming flow. We derived the finite element scheme for the numerical solution of the coupling system. To couple with the difficulty of strong convective flow in the simulation, the SUPG term is included in the scheme to obtain numerical stability as well as accuracy. The coupled numerical equation system is solve in a semi-coupled way, where the hydraulic equations and the thermal equation are solved sequentially by updating their solutions in an iterative pattern. A detailed convergence study for both the spatial and temporal convergence behaviors was discussed in this paper. Various comparisons between the numerical solutions and previous experimental results validate the accuracy of our numerical methods.

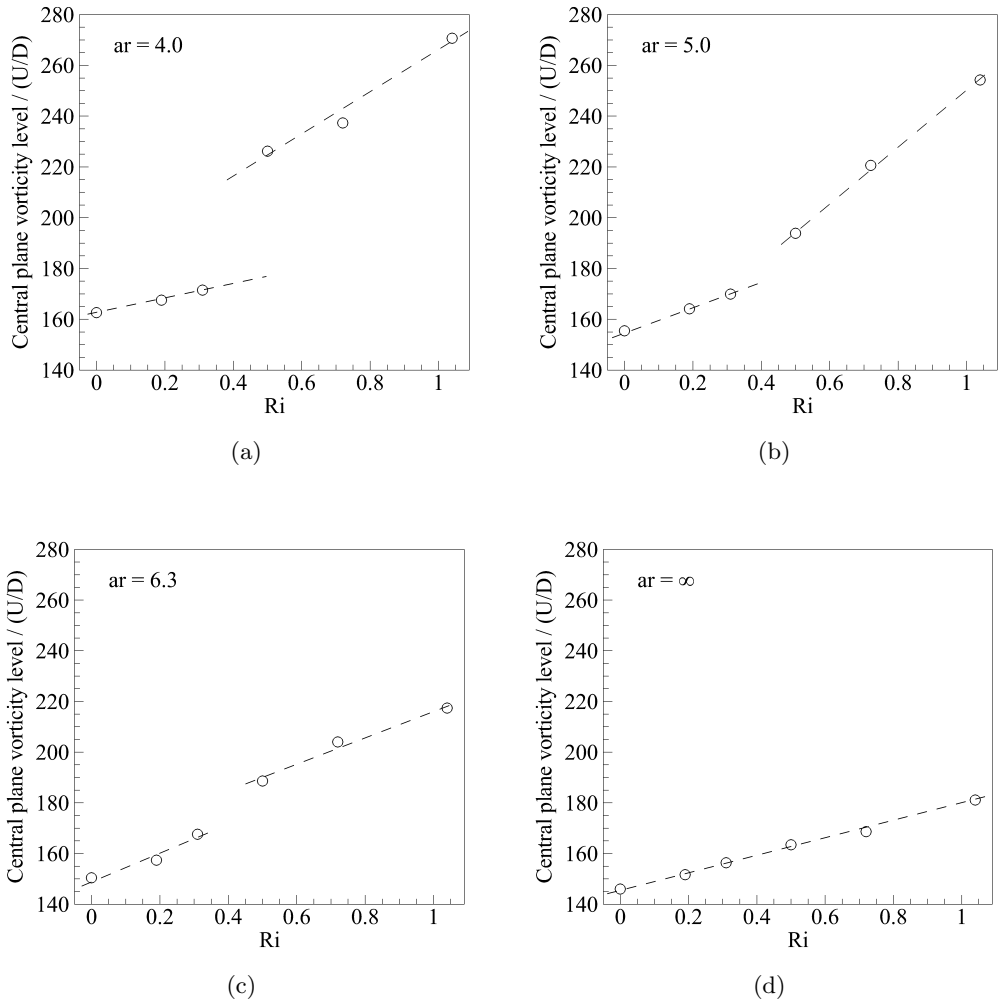


Figure 2.14 Central surface vortex strength levels plotted for all Richardson numbers at $ar = 4.0$, 5.0, 6.3 and ∞.

We focused on investigating the transitions of the flow in the wake behind the heated cylinder. One is the existence of a flow transition in the wake. When the Richardson number reaches beyond a critical value, the ensemble-averaged centerline velocity curve significantly shifts to a position downstream. We observed from our numerical results that this flow transition is a discontinuous change of the wake flow, where the transition occurs in a very short range of Richardson number. Then we discussed the effect of the velocity gradient along the cylinder axis direction on the flow transition by varying the cylinder aspect ratio. We confirmed that the flow transition exists in all range of finite cylinder aspect ratios. However, the flow transition appears less obvious as the velocity gradient along the cylinder axis direction vanishes. The flow transition eventually disappears as such velocity gradient is eliminated. Another is the transition of the vortex shedding patterns with change of Richardson number. We numerically confirmed that two stable vortex shedding pattern exist in the wake behind the heated cylinder, the staggered shedding pattern and the simultaneous shedding pattern. In between a "dead-flow" zone typed pattern was also captured, but was found existing only in a very short range of Richardson number. All these observations of the vortex shedding patterns from our numerical study formed good agreement with previous experimental results. We then confirmed that the vortex shedding pattern exists where there is a nonzero velocity gradient along the cylinder axis direction. The "dead-flow" zone behind the cylinder becomes shorter, and the simultaneous shedding pattern turns less obvious as such velocity gradient tends to zero. Eventually the vortex shedding pattern converges to a unit staggered pattern as this velocity gradient is totally removed. The discussions and analysis on the relation between the velocity gradient along the cylinder axis direction with the flow transition and then vortex shedding pattern transition broaden our understanding of the mechanisms of the convection/diffusion contraflow behind the heated cylinder with low Reynolds number.

CHAPTER 3. A DIFFUSE INTERFACE MODEL FOR INCOMPRESSIBLE TWO-PHASE FLOW: STABILIZED FINITE ELEMENT METHOD FOR LARGE DENSITY RATIOS, GRID RESOLUTION STUDY, AND 3D PATTERNED SUBSTRATE WETTING PROBLEM

Modified from a paper to be submitted to

Computer Methods in Applied Mechanics and Engineering

Yu Xie, Olga Wodo, and Baskar Ganapathysubramanian

3.1 Abstract

Flows involving interface between fluid components have been widely investigated both experimentally and numerically due to their important theoretical and engineering features. Diffuse interface model has shown its success in simulating multiphase flows. In this paper, a stabilized finite element method is formulated based on the diffuse interface model to simulate incompressible and immiscible two-phase flows. The phase field function, velocity, and pressure are governed by a coupled Cahn-Hilliard Navier-Stokes equation system. A conservative form of the convective term in the Cahn-Hilliard equation is used which guarantees the conservation of mass of both fluid components. A continuous formula is used to compute the surface tension of the interface, which puts lower requirements on the resolution of the spatial discretization of the interface. A four-step fractional scheme is employed to decouple the pressure from the Navier-Stokes equation, which provides an efficient and second order accurate time discretization on velocity. The streamline-upwind Petrov-Galerkin (SUPG) stabilization term is added to avoid spurious oscillations. Special treatments on the approximations of density and viscosity terms

towards dealing with large density ratios are discussed. We also perform exhaustive tests to determine the minimal spatial discretization resolution in this study. Finally we apply this framework to both 2D and 3D simulations to validate the capability of the solver. Specifically, we explored the effects of surface patterns on the droplet spreading process. Formations of wetting spots and air entrapment are studied on both grooved and checker-patterned surfaces. It is found that the droplet is able to stay on top of the grooves or the cubic pillars in the spreading stage, but merges into the grooves or space among the cubic pillars during the receding stage. We also found that the grooved surface tends to reduce the size of the wetting spot but almost has no effect on the height of the droplet when receding reaches steady state, while the checkered pattern maintains the size of the wetting spot but reduces the droplet height, compared with the non-patterned solid surface case.

3.2 Introduction

Simulation of multiphase flows involving an interface has long been an interesting and challenging topic. Numerical study of such problems not only requires answers to the basic questions like what are the appropriate governing equations and whether physical properties like mass and energy conservation laws are preserved, but also faces the challenges of increasing demand of accuracy and efficiency for simulating large scale problems under complex circumstances such as strong convective effects and large density difference between fluid components. An accurate and efficient method of computing the position and simulating the motion of the interface between fluid components is always a focus of discussion, particularly in cases when surface tension drives fluid flow. There are mainly two families of methods for simulating multiphase fluid systems. The two methods have different philosophies of treating the interface to compute the surface tension.

One is the family of techniques involving tracking/capturing the exact location and shape of the interface between the two components. This family includes marker (or front-tracking) methods [Tryggvason et al. (2001)], which interpolate the interface with a set of connected marker points; volume of fluid (VOF) methods [Gueyffier et al. (1999)], which represent the fluid portion in a natural way with a color function; level set methods [Osher and Sethian

(1988)], which captures the interface as the zero level curve of a continuous function defined by the algebraic distance between the current position and the interface; and many other variants based on this idea. This family of methods with interface reconstruction is known for its accuracy of resolving the interface and the surface tension calculation.

Though the interface reconstruction methods are well known for their accuracy in capturing the interface, they also have drawbacks and difficulties in tracking complex topological changes and extra computational resources spent on reconstructing the interface. An alternative way is trying to avoid recording the exact position of interface, which leads to the family of diffuse interface model. The idea of the diffuse interface model can be found in several studies reviewed by Anderson et al. (1998). Different from the point of view of sharp interface models which consider the two immiscible components being connected with an interface with zero thickness, the diffuse interface model represents the interface as a smooth transition with non-zero thickness, specifically a phase field function on the whole domain. The surface tension is directly computed from the phase field function without reconstructing the interface first.

Numerical algorithms for implementing the diffuse interface model have been the focus of several recent reports. Hohenberg and Halperin (1977) derived an abstract model for simulating interfacial flows of incompressible fluids with matched densities by coupling the hydraulic equation with Cahn-Hilliard diffusion, which is well known as "model H." Lowengrub and Truskinovsky (1998) pointed out that binary fluids with incomprehensible components may actually be compressible, and they derived a quasi-incompressible formula for the flows of binary mixtures with a density contrast. Yue et al. (2006) gave a 2D fully adaptive finite element model for simulating interfacial dynamics in incompressible viscoelastic fluids. A 3D numerical scheme was later developed by Zhou et al. (2010) based on the 2D model. Ceniceros et al. (2010) decoupled the discrete model H system with a semi-implicit time discretization, and solved the linear system with a multigrid method combining a mesh refinement algorithm. Ding et al. (2007) derived a finite volume scheme for simulating binary mixture flows with large density ratios. Researchers are also curious about the accuracy of the diffuse interface model. Caginalp and Chen (1998) mathematically verified the existence of the sharp interface limit for the phase field model. Numerical investigations of the convergence of the phase field

model to sharp interface limit can be found in Lowengrub and Truskinovsky (1998) and Yue et al. (2006). Yue et al. (2010) and Yue and Feng (2011) later extended the study of the sharp interface limit on the moving contact line problem. There are still interesting open questions on the diffuse interface model. On the theory for this model, researchers questioned whether the model preserves the mass conservation of fluid components regardless of the density ratio, how the surface tension can be accurately but also conveniently evaluated from the phase field, and whether the thin interface limit exists for this model with unmatched densities of fluid components. On the aspect of computation, researchers question whether the discretized numerical scheme preserves the mass conservation ensured by the theoretical model, whether the diffuse interface model has a requirement on the spatial resolution of elements through the interface, how the interfacial thickness affects the accuracy of the model with unmatched density ratios, and if the numerical scheme scales with the size of the problem. Our work attempts to understand and answer these questions.

The most frequently utilized model to describe phase field evolution is the Cahn-Hilliard model, which requires a fourth order derivative of the phase field function. The Cahn-Hilliard equation theoretically guarantees the mass conservation law of fluid components.[1] Based on this model, the solution procedure of the multiphase system requires solving a coupled equation system of Navier-Stokes equation and Cahn-Hilliard equation. The finite element method is employed in this paper to obtain the numerical solution to such an equation system. Compared with the finite volume method, the finite element method does not require reconstruction of cell quantities and flux terms from neighbor cells, making it more easily parallelized. To cope with the stability issue of the traditional Petrov-Galerkin scheme for applying the finite element method on fluid dynamics, Brooks and Hughes (1982) introduced the streamline upwind/Petrov-Galerkin (SUPG) term in the traditional weak form of Navier-Stokes equations. This method significantly reduced the spurious oscillations in the numerical solution of velocity, thus providing more accurate approximation. Their method treated the pressure with a penalty method, which was later improved [Tezduyar et al. (1992)] with a more robust method by incor-

[1] Feng et al. (2007) also solved the phase field function based on an Allen-Cahn model containing the phase field function and its second order derivative, which however does not promise the mass conservation of fluid components [Feng (2006)].

porating the pressure-stabilizing/Petrov-Galerkin (PSPG) term into the continuity equation. A finite element scheme combined with these two terms has been applied to simulate microscale flows with complex geometry [Jaeger et al. (2012)] and flows under a wide range of Reynolds numbers and situations [Amini et al. (2013)]. However, when flow with a large density ratio was considered, it was observed that the fully implicit finite element scheme with the PSPG method led the linear solver to encounter more with divergence issues. Thus for the current study, a four-step fractional method [Chorin (1968); Choi et al. (1997)] is utilized to decouple the pressure equation from the momentum equation, which performs efficiently in solving large problems without losing accuracy of the velocity field solution. Details of this scheme with the finite element method stabilized by the SUPG term will be introduced in section 3.4. Ceniceros et al. (2010) employed a similar projection method based pressure correction scheme to decouple the pressure from the momentum equation for density matched flows ("model H"), and numerically demonstrated the spatial convergence of this scheme. However, how well the scheme performs in parallelization, especially for the flows with unmatched densities, is not answered by previous literature. In this study, we numerically investigate the parallel capability of the four-step fractional method when coupled with the Cahn-Hilliard equation for flows with unmatched densities.

This paper is aimed at providing an efficient and robust framework for applying the stabilized finite element method to simulate incompressible and immiscible two-phase flows. Our contributions include: (1) gave a semi-coupled mass-conserved Cahn-Hilliard Navier-Stokes equation system with an accurate continuous approximation of the surface tension for governing two-phase flows, and verified its capability of simulating flows with a large density ratio; (2) discretized the governing equation using the four-step fractional scheme with SUPG stabilized finite element method, where the pressure is decoupled from the momentum equation, ensuring the capability of solving large scale problems; (3) provided a guidance on the minimal grid resolution of the diffuse interface for the combined Cahn-Hilliard and Navier-Stokes system to ensure physical correctness of the phase field model; (4) studied the convergence behavior of interface thickness, or alternatively, the Cahn number Cn at different density ratios of fluid components; (5) performed detailed scalability tests of our numerical framework for simulating

3D problems on different machines; (6) verified the accuracy of the current framework with various benchmark numerical tests, and utilized this framework on investigating the wetting process of 3D droplet impact on solid substrate with different kinds of surface patterns.

Structure of this work is organized as follows. In section 3.3, we will introduce the governing equations, evaluation of surface tension, and a strategy for dealing with large density ratios. Then the numerical schemes are discussed in section 3.4. Validation of our numerical schemes as compared with previous experimental and numerical results will be completed in section 3.5, where readers will also find various other applications with potential engineering value. Finally, a series of detailed scalability tests are performed and will be reported in section 3.6.

3.3 Governing equations

The diffusive interface model represents the incompressible binary mixture by a phase field function ϕ, as a measure of the volume fraction of the immiscible fluid components, where $\phi = \phi(\boldsymbol{x})$ is a smooth function of the spatial coordinates $\boldsymbol{x}$, $\boldsymbol{x} \in \Omega$, and $\Omega \subset \mathbb{R}^{n_{sd}}$ is the n_{sd} dimensional ($n_{sd} = 2, 3$) physical domain. The value of ϕ varies between -1 and 1, with each of these end points representing one fluid component. Thus the physical properties of the fluid components, i.e. density ρ and viscosity η, are expressed as function of ϕ

$$\rho(\phi) = \rho_- \frac{1-\phi}{2} + \rho_+ \frac{1+\phi}{2}, \quad \eta(\phi) = \eta_- \frac{1-\phi}{2} + \eta_+ \frac{1+\phi}{2}, \tag{3.1}$$

where $(\cdot)_-$ and $(\cdot)_+$ represent the parameters of the negative ϕ ($\phi = -1$) and the positive ϕ ($\phi = 1$) components, respectively.

Motion of the two fluids is governed by the transient incompressible Navier-Stokes equation with an additional surface tension $\boldsymbol{f}_s$ determined by the phase field function ϕ

$$\rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + (\nabla \cdot \boldsymbol{u}) \boldsymbol{u} \right] = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g} + \boldsymbol{f}_s, \tag{3.2}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{3.3}$$

where $\boldsymbol{g}$ is the gravitational acceleration, $\boldsymbol{\sigma}$ is the the stress tensor given as

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\eta\boldsymbol{\epsilon}(\boldsymbol{u}), \tag{3.4}$$

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\left[\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right], \tag{3.5}$$

and $\boldsymbol{I}$ being the identity tensor. There have been various discussions on the form of the continuity Equation (3.3) for the fluids with a density contrast. Lowengrub and Truskinovsky (1998) mentioned in their approach that the mass-averaged velocity field is not solenoidal, i.e. does not satisfy Equation (3.3). So they introduced a quasi-incompressible model as a modification of model H. However, numerical instability was observed for topology change of the interface as mentioned by Ding et al. (2007). Ding et al. (2007) also further discussed the verification of Equation (3.3) and suggested that Equation (3.3) serves as an accurate approximation for binary flows with a density contrast, where such equation gives simulation results with good agreement to the ones obtained from level-set model based methods.

The boundary conditions for Equations (3.3) are given as

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma_1, \tag{3.6}$$

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{h}(\boldsymbol{x}), \quad \boldsymbol{x} \in \Gamma_2, \tag{3.7}$$

where Γ_1 and Γ_2 are the essential boundary (or Dirichlet boundary) and natural boundary (or Neumann boundary), respectively, and $\Gamma_1 \cup \Gamma_2 = \partial\Omega$, $\Gamma_1 \cap \Gamma_2 = \emptyset$. Function $\boldsymbol{g}(\boldsymbol{x})$ are prescribed velocity vectors on Γ_1, $\boldsymbol{h}(\boldsymbol{x})$ is the traction vectors prescribed on Γ_2, with $\boldsymbol{n}$ the outward normal vectors.

The diffuse interface model assumes a non-zero thickness of the interface as a smooth transition between the two fluid components. Evolution of the interface is described by the Cahn-Hilliard equation Cahn and Hilliard (1958), which is derived by minimizing the free energy of the system given as the GinzburgLandau form

$$H(\phi) = \int_\Omega \left[f(\phi) + \epsilon^2 |\nabla \phi|^2\right] \mathrm{d}\Omega, \tag{3.8}$$

where ϵ is a constant defined as the interfacial parameter. The expression in the integral represents the summation of bulk energy density $f(\phi)$ and surface energy $\epsilon^2|\nabla\phi|^2$. Usually $f(\phi)$ is chosen as a double-well function

$$f(\phi) = \frac{1}{4}(1-\phi^2)^2. \tag{3.9}$$

The double-well function has two minima for $\phi \in [-1, 1]$, corresponding to the two equilibrium states of the binary fluid components. Then the Cahn-Hilliad equation is represented as

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \boldsymbol{u}) = \nabla \cdot [M(\phi) \nabla \mu(\phi)], \tag{3.10}$$

$$\mu(\phi) = \frac{\delta H(\phi)}{\delta \phi} = \phi^3 - \phi - \epsilon^2 \Delta \phi, \tag{3.11}$$

where $\boldsymbol{u}$ is the flow velocity, $M(\phi)$ is the mobility, and $\mu(\phi)$ is the chemical potential. Here we set M as a constant over the whole domain.[2]

The boundary conditions for Cahn-Hilliard Equations (3.11) are written as follows

$$\boldsymbol{n} \cdot \nabla \mu = 0 \quad \boldsymbol{x} \in \partial \Omega, \tag{3.12}$$

$$\boldsymbol{n} \cdot \nabla \phi = \frac{1}{\sqrt{2}\epsilon} \cos(\theta_S)(1 - \phi^2) \quad \boldsymbol{x} \in \partial \Omega, \tag{3.13}$$

where θ_S is the static contact angle. The boundary condition for μ acts as a no-flux restriction for the fluid components in the control volume. Equation (3.13) determines the direction of the interface contacting the boundary surface given static contact angle θ_S [Li et al. (2009)].

The Navier-Stokes and Cahn-Hilliard equations are coupled by introducing the convective term $\nabla \cdot (\phi \boldsymbol{u})$ into the Cahn-Hilliard equation and by including the surface tension $\boldsymbol{f}_s = \boldsymbol{f}_s(\phi)$ into the Navier-Stokes equation. Sometimes the convective term $\nabla \cdot (\phi \boldsymbol{u})$ is also written as $\phi \nabla \cdot \boldsymbol{u}$. However, it has been discussed by Minjeaud (2012) that this form cannot guarantee the conservation of mass of the fluid components. Thus we keep the form used in Equation (3.11), which preserves the mass conservation of fluid components within domain Ω.

The surface tension $\boldsymbol{f}_s$ in the momentum Equation (3.3) is a numerical approximation of the surface tension using the continuum surface force (CSF) formula [Brackbill et al. (1992)], which converts the surface tension into a continuous body force as a function of ϕ. There are multiple expressions for $\boldsymbol{f}_s$. Lowengrub and Truskinovsky (1998) represented $\boldsymbol{f}_s$ as

$$\boldsymbol{f}_s^1 = \sigma \epsilon \alpha \nabla \cdot (|\nabla \phi|^2 I - \nabla \phi \otimes \nabla \phi), \tag{3.14}$$

[2] Wodo and Ganapathysubramanian (2011) also suggest a degenerated form of M, $M = D(1 - \phi^2)$, where D is the diffusivity, when assuming that the diffusion process mainly occurs in a small neighbourhood of the thin interface. However, we observed that the linear solver is more flexibly controlled for convergence without loss of accuracy when constant mobility is used. Thus in the current paper, we use constant mobility without mentioning it separately.

where σ is the surface tension coefficient. The parameter α matches the body force of the diffusive interface model with the surface tension of a sharp interface, which will be evaluated later in Equation (3.18). The operator $\otimes$ is the tensor product operator, i.e. $(\nabla\phi \otimes \nabla\phi)_{ij} = \frac{\partial\phi}{\partial x_i}\frac{\partial\phi}{\partial x_j}$. Boyer (2002) introduced the form of $\boldsymbol{f}_s$ as the product of the chemical potential M and the gradient of ϕ

$$\boldsymbol{f}_s^2 = \frac{\sigma\alpha}{\epsilon}\mu\nabla\phi. \tag{3.15}$$

Another similar expression can be found in the work of Jacqmin (1999) with an exchange of the derivative:

$$\boldsymbol{f}_s^3 = -\frac{\sigma\alpha}{\epsilon}\phi\nabla\mu. \tag{3.16}$$

Kim (2005) gave another continuous formula for $\boldsymbol{f}_s$ as an approximation of the force term used in the level set method [Chang et al. (1996)]:

$$\boldsymbol{f}_s^4 = -\sigma\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\epsilon\alpha|\nabla\phi|^2\frac{\nabla\phi}{|\nabla\phi|}, \tag{3.17}$$

where $\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)$ represents the interface curvature κ, $\epsilon\alpha|\nabla\phi|^2$ approximates the Dirac delta function $\delta(\phi)$, and $\frac{\nabla\phi}{|\nabla\phi|}$ is the normalized gradient of ϕ. Force $\boldsymbol{f}_s^1$ usually requires a modification of pressure to an effective value as shown in the paper of Dong and Shen (2012). Kim also recommended the usage of $\boldsymbol{f}_s^4$ by comparing its performance against the other three expressions with various numerical experiments. Additionally, the curvature κ can be easily solved from a Poisson type equation, without introducing higher order derivatives of ϕ, which is easily implemented by the finite element method with linear basis function. Thus in the present paper, we use the surface tension form $\boldsymbol{f}_s^4$.

The parameter α relates the diffusive model with the sharp interface model, and can be derived from the equilibrium state solution. Parameter α should satisfy the equation

$$\epsilon\alpha\int_{-\infty}^{\infty}(\phi_x^{eq})^2\mathrm{d}x = 1, \tag{3.18}$$

where ϕ_x^{eq} is the equilibrium state solution [Kim (2005)]. In the one dimension case, the equilibrium state has an analytical solution, $\phi_x^{eq}(x) = \tanh\left(\frac{x}{\sqrt{2\epsilon}}\right)$ [Bray (1994)], when ϕ of the two stable uniform states are $\phi(x) = \pm 1$, and chemical potential is given by Equation (3.11).

Substituting the equilibrium state solution into Equation (3.18), we get $\alpha = \frac{2\sqrt{2}}{3}$. Then we get the Cahn-Hilliard Navier-Stokes governing equation system from the above analysis as following

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) + \nabla p - \nabla\cdot\left[\eta\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right)\right] = \rho\boldsymbol{g} - \sigma\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)\epsilon\alpha|\nabla\phi|^2\frac{\nabla\phi}{|\nabla\phi|}, \tag{3.19}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{3.20}$$

$$\frac{\partial\phi}{\partial t} + \nabla\cdot(\phi\boldsymbol{u}) = \nabla\cdot\left[M\nabla\mu(\phi)\right], \tag{3.21}$$

$$\mu(\phi) = \phi^3 - \phi - \epsilon^2\Delta\phi. \tag{3.22}$$

The governing Equations (3.21)-(3.20) can be written in a dimensionless form by scaling variables with characteristic velocity magnitude U, length L, density ρ_+:

$$\boldsymbol{u}^* = \frac{\boldsymbol{u}}{U}, \quad \boldsymbol{x}^* = \boldsymbol{x}/L, \quad t^* = \frac{t}{L/U}, \quad p^* = \frac{p}{\rho_+ U^2}. \tag{3.23}$$

Then the dimensionless governing equations are given by dropping the superscript "$*$"

$$\tilde{\rho}\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) + \nabla p - \frac{1}{Re}\nabla\cdot\left[\tilde{\eta}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right)\right] = \frac{\tilde{\rho}\boldsymbol{g}_0}{Fr} - \frac{2\sqrt{2}}{3}\frac{Cn}{We}\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right)|\nabla\phi|^2\frac{\nabla\phi}{|\nabla\phi|}, \tag{3.24}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{3.25}$$

$$\frac{\partial\phi}{\partial t} + \nabla\cdot(\phi\boldsymbol{u}) = \frac{1}{Pe}\nabla\cdot\left[M(\phi)\nabla\mu(\phi)\right], \tag{3.26}$$

$$\mu(\phi) = \phi^3 - \phi - Cn^2\Delta\phi, \tag{3.27}$$

where $\tilde{\rho} = \rho_-/\rho_+$ is the density ratio, $\tilde{\eta} = \eta_-/\eta_+$ is the viscosity ratio, and $\boldsymbol{g}_0$ indicates the gravitational direction unit vector. In the above equation, parameter Re is the Reynolds number, defined as $Re = \rho_+ UL/(\mu_+)$, representing ratio between inertia force and viscous force. Relative gravity is represented by the Froude number Fr, defined as $Fr = U^2/(gL)$, where g is the magnitude of gravitational acceleration. The Weber number, We, takes the form $We = \rho_+ U^2 L/\sigma$, representing the ratio between inertial force and surface tension. The

Cahn number Cn is given by $Cn = \epsilon/L$, representing the relative interface thickness and affects the accuracy of the solution to the diffusive interface model. Parameter Pe is the Péclet number, $Pe = LU/(M\mu)$, where M and μ are characteristic mobility and chemical potential. Commonly, the Péclet number is taken as a number proportional to the inverse of the Cahn number [Ceniceros et al. (2010); Ding et al. (2007)], i.e. $Pe = O(1/Cn)$. In this paper, we take $Pe = 1/Cn$.

3.4 Numerical Schemes

In this section, we discuss the numerical algorithms for solving the governing equations. Given the solutions at time step t^n as $\boldsymbol{u}^n$, p^n, ϕ^n, and μ^n, which represent the velocity, pressure, phase field function and chemical potential respectively, we solve for the solution at time step t^{n+1} successively with the time discretization schemes introduced below.

The phase field function and chemical potential at time step t^{n+1} are obtained by solving the Cahn-Hilliard equation with a Crank-Nicolson scheme

$$\frac{\phi^{n+1}-\phi^n}{\Delta t} + \frac{1}{2}\nabla\cdot(\phi^n\boldsymbol{u}^n) + \frac{1}{2}\nabla\cdot\left(\phi^{n+1}\boldsymbol{u}^{n+1}\right) = \frac{1}{2}\frac{1}{Pe}\nabla\cdot\nabla\mu^n + \frac{1}{2}\frac{1}{Pe}\nabla\cdot\nabla\mu^{n+1}, \tag{3.28}$$

$$\mu^{n+1} = {\phi^{n+1}}^3 - \phi^{n+1} - Cn^2\nabla\cdot\nabla\phi^{n+1}. \tag{3.29}$$

The curvature κ^{n+1} is then calculated based on the phase field function ϕ^{n+1} by the following equation

$$\kappa^{n+1} = \nabla\cdot\left(\frac{\nabla\phi^{n+1}}{|\nabla\phi^{n+1}|}\right). \tag{3.30}$$

Velocity and pressure are solved by a four-step fractional step method [Choi et al. (1997)], where pressure is decoupled from the momentum equation and solved by a Poisson type equation. The four-step fractional method incorporated with surface tension and varying density/viscosity is given as following equations

$$\begin{aligned}&\tilde{\rho}^{n+1}\left(\frac{\hat{\boldsymbol{u}}-\boldsymbol{u}^n}{\Delta t} + \frac{1}{2}\left(\hat{\boldsymbol{u}}\cdot\nabla\hat{\boldsymbol{u}} + \boldsymbol{u}^n\cdot\nabla\boldsymbol{u}^n\right)\right) + \nabla p^n - \frac{1}{2}\frac{1}{Re}\nabla\cdot\left[\tilde{\eta}\left(\nabla\boldsymbol{u}^n + (\nabla\boldsymbol{u}^n)^T\right)\right]\\ &-\frac{1}{2}\frac{1}{Re}\nabla\cdot\left[\tilde{\eta}\left(\nabla\boldsymbol{u}^{n+1} + (\nabla\boldsymbol{u}^{n+1})^T\right)\right] = \frac{\tilde{\rho}^{n+1}}{Fr}\boldsymbol{g}_0 - \frac{2\sqrt{2}}{3}\frac{Cn}{We}\kappa^{n+1}|\nabla\phi^{n+1}|\nabla\phi^{n+1},\end{aligned} \tag{3.31}$$

$$\tilde{\rho}^{n+1}\frac{\boldsymbol{u}^* - \hat{\boldsymbol{u}}}{\Delta t} = \nabla p^n, \tag{3.32}$$

$$\nabla \cdot \nabla p^{n+1} = \frac{\tilde{\rho}^{n+1}}{\Delta t}\nabla \cdot \boldsymbol{u}^*, \tag{3.33}$$

$$\tilde{\rho}^{n+1}\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\nabla p^{n+1}. \tag{3.34}$$

In the above equations, $\hat{\boldsymbol{u}}$ and $\boldsymbol{u}^*$ are the intermediate velocities which are solved based on pressure p^n from the previous time step. Pressure p^{n+1} is then updated by Equation (3.33) with the intermediate velocities. Notice that the intermediate velocities do not satisfy the incompressible condition, i.e. $\nabla \cdot \boldsymbol{u}^* \neq 0$. So the last step corrects the velocity with the latest pressure field p^{n+1} to guarantee the incompressible condition. Here we also applied the Crank-Nicolson scheme on the convective and diffusive terms in the momentum equation to obtain second order accuracy in the time discretization.

Equations (3.28)-(3.34) form a coupled system of ϕ^{n+1}, μ^{n+1}, $\boldsymbol{u}^{n+1}$ and p^{n+1}. Instead of directly solving the whole system in one step, we perform an iterative algorithm to solve a semi-decoupled equation system, shown by the flowchart in Figure 3.1. The Cahn-Hilliard Equations (3.28) and (3.29) are solved first at each time step. Notice that in Equation (3.28), velocity $\boldsymbol{u}^{n+1}$ is still unknown at the current stage. So we assign the solution at previous time step, $\boldsymbol{u}^n$, as its initial value. After obtaining the solution of ϕ, curvature κ is evaluated from Equation (3.30) with the latest phase field function. Velocity and pressure are then updated with the latest phase field function and curvature by solving the Equations (3.31)-(3.34). After this one cycle of iteration, we compare the current phase field function ϕ^{n+1} with the one at the previous iteration, if the L2-norm of the relative error is above a threshold ε, then we continue the iteration using the latest solution, until the stop condition is satisfied. Notice at least two iterations are required to perform this comparison. This method is also known as the "block-iterative" approach for solving coupled equations Tezduyar and Sathe (2007). No general theory is available on the convergence condition for such method Cervera et al. (1996). Considering the iteration is required at each time step, the value of ε should be within the

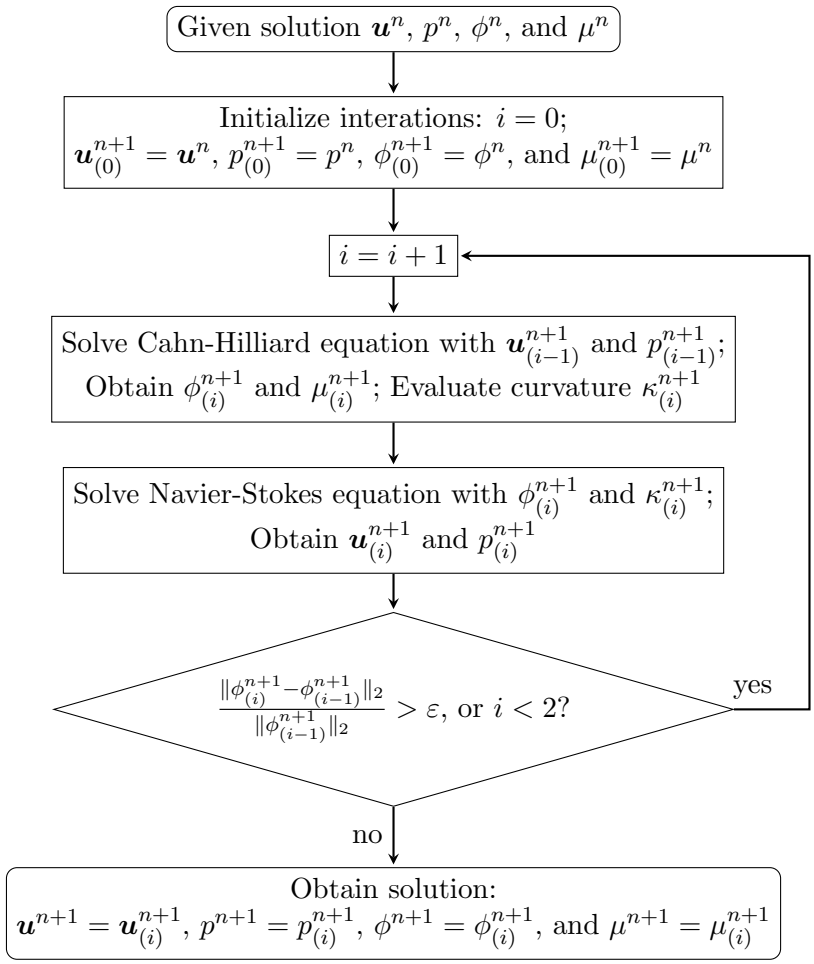


Figure 3.1 Flow chart of the iteration solver for the semi-coupled Cahn-Hilliard Navier-Stokes equation system.

order of Δx^2 to avoid introducing more numerical error from this scheme, regarding the second order spatial convergence rate of the finite element formula, e.g. take $\varepsilon = 1 \times 10^{-3}$ when mesh size Δx has characteristic value of 1×10^{-1}. Usually the convergence is acquired within two or three iterations, which is highly efficient.

Denote the spatial discretization of the physical domain Ω as Ω^e, $e = 1, 2, \cdots, n_{el}$, where n_{el} is the total number of elements in Ω. Associated with the spatial discretization, the following finite element interpolation function spaces are defined for ϕ_h as the finite dimensional approximations of the phase field function ϕ, μ_h as the finite dimensional approximations of chemical potential μ, $\boldsymbol{u}_h$ as the finite dimensional approximations of velocity $\boldsymbol{u}$, and p_h as the finite dimensional approximations of pressure p:

$$\mathcal{S}_h^{\phi} = \mathcal{V}_h^{\phi} = \{\phi_h | \phi_h \in \mathcal{H}^1(\Omega)\}, \tag{3.35}$$

$$\mathcal{S}_h^{\mu} = \mathcal{V}_h^{\mu} = \{\mu_h | \mu_h \in \mathcal{H}^1(\Omega)\}, \tag{3.36}$$

$$\mathcal{S}_h^{\boldsymbol{u}} = \{\boldsymbol{u}_h | \boldsymbol{u}_h \in \mathcal{H}^1(\Omega), \boldsymbol{u}_h = \boldsymbol{g}_h(\boldsymbol{x}), \boldsymbol{x} \in \Gamma_1\}, \tag{3.37}$$

$$\mathcal{V}_{h,0}^{\boldsymbol{u}} = \{\boldsymbol{w}_h | \boldsymbol{w}_h \in \mathcal{H}^1(\Omega), \boldsymbol{w}_h = \boldsymbol{0} \text{ on } \Gamma_1\}, \tag{3.38}$$

$$\mathcal{S}_h^{p} = \mathcal{V}_h^{p} = \{q_h | q_h \in \mathcal{H}^1(\Omega)\}, \tag{3.39}$$

where $\mathcal{H}^1(\Omega)$ represents the Sobolev space of vector functions defined on Ω^e. Thus the stabilized Galerkin formulation of the strong form of the equation system (3.21)-(3.20) can be written. Given $\phi_h^n \in \mathcal{S}_h^{\phi}$, $\mu_h^n \in \mathcal{S}_h^{\mu}$, $\boldsymbol{u}_h^n \in \mathcal{S}_h^{\boldsymbol{u}}$, and $p_h^n \in \mathcal{S}_h^{p}$, find $\phi_h^{n+1} \in \mathcal{S}_h^{\phi}$, $\mu_h^{n+1} \in \mathcal{S}_h^{\mu}$, $\boldsymbol{u}_h^{n+1} \in \mathcal{S}_h^{\boldsymbol{u}}$, and $p_h^{n+1} \in \mathcal{S}_h^{p}$, such that, $\forall \psi_h \in \mathcal{V}_h^{\phi}$, $\forall \boldsymbol{w}_h \in \mathcal{V}_{h,0}^{\boldsymbol{u}}$, and $\forall q_h \in \mathcal{V}_h^{p}$, the following weak form of equations are satisfied:

$$\begin{aligned}
&\int_\Omega \psi_h \cdot \left(\frac{\phi_h^{n+1} - \phi_h^n}{\Delta t} + \frac{1}{2}\nabla \cdot (\phi_h^n \boldsymbol{u}_h^n) + \frac{1}{2}\nabla \cdot (\phi_h^{n+1} \boldsymbol{u}_h^{n+1}) \right) \mathrm{d}\Omega \\
&+ \int_\Omega \left(\frac{1}{2}\frac{1}{Pe}\nabla\psi_h \cdot \nabla\mu_h^n + \frac{1}{2}\frac{1}{Pe}\nabla\psi_h \cdot \nabla\mu_h^{n+1} \right) \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{el}} \int_\Omega \delta(\psi_h) \cdot \left(\frac{\phi_h^{n+1} - \phi_h^n}{\Delta t} + \frac{1}{2}\nabla \cdot (\phi_h^n \boldsymbol{u}_h^n) + \frac{1}{2}\nabla \cdot (\phi_h^{n+1} \boldsymbol{u}_h^{n+1}) \right) \mathrm{d}\Omega \\
&= 0,
\end{aligned} \tag{3.40}$$

$$\begin{aligned}
&\int_\Omega \psi_h \cdot \left((\phi_h^{n+1})^3 - \phi_h^{n+1} - \mu_h^{n+1} \right) \mathrm{d}\Omega + \int_\Omega Cn^2 \nabla\psi_h \cdot \nabla\phi^{n+1} \mathrm{d}\Omega \\
&+ \sum_{e=1}^{n_{el}} \delta(\psi) \cdot \left((\phi^{n+1})^3 - \phi^{n+1} - \mu^{n+1} \right) \mathrm{d}\Omega = 0,
\end{aligned} \tag{3.41}$$

with the curvature equation in the weak form of formula

$$\int_\Omega w_h \kappa_h^{n+1} \mathrm{d}\Omega + \int_\Omega (\nabla w_h) \cdot \left(\frac{\nabla\phi_h^{n+1}}{|\nabla\phi_h^{n+1}|} \right) \mathrm{d}\Omega = 0, \tag{3.42}$$

and the four-step fractional method for the Navier-Stokes equation in the weak form of formula as follows

$$\int_\Omega \boldsymbol{w_h}\cdot\left\{\tilde{\rho}_h^{n+1}\left(\frac{\hat{\boldsymbol{u}}_h-\boldsymbol{u}_h^n}{\Delta t}+\frac{1}{2}\boldsymbol{u}_h^n\cdot\nabla\boldsymbol{u}_h^n+\frac{1}{2}\hat{\boldsymbol{u}}_h\cdot\nabla\hat{\boldsymbol{u}}_h\right)+\nabla p_h^n-\frac{\tilde{\rho}_h^{n+1}}{Fr}\boldsymbol{g}_0\right.$$
$$\left.+\frac{2\sqrt{2}}{3}\frac{Cn}{We}\kappa_h^{n+1}|\nabla\phi_h^{n+1}|\nabla\phi_h^{n+1}\right\}\mathrm{d}\Omega$$
$$+\int_\Omega\frac{1}{Re}\nabla\boldsymbol{w}:\left[\frac{1}{2}\tilde{\eta}\left(\nabla\boldsymbol{u}_h^n+(\nabla\boldsymbol{u}_h^n)^T\right)+\frac{1}{2}\tilde{\eta}\left(\nabla\hat{\boldsymbol{u}}_h+(\nabla\hat{\boldsymbol{u}}_h)^T\right)\right]\mathrm{d}\Omega$$
$$+\sum_{e=1}^{n_{el}}\int_{\Omega^e}\boldsymbol{\delta}(\boldsymbol{w})\cdot\left\{\tilde{\rho}_h^{n+1}\left(\frac{\hat{\boldsymbol{u}}_h-\boldsymbol{u}_h^n}{\Delta t}+\frac{1}{2}\boldsymbol{u}_h^n\cdot\nabla\boldsymbol{u}_h^n+\frac{1}{2}\hat{\boldsymbol{u}}_h\cdot\nabla\hat{\boldsymbol{u}}_h\right)+\nabla p_h^n\right.$$
$$\left.-\frac{\tilde{\rho}_h^{n+1}}{Fr}\boldsymbol{g}_0+\frac{2\sqrt{2}}{3}\frac{Cn}{We}\kappa_h^{n+1}|\nabla\phi_h^{n+1}|\nabla\phi_h^{n+1}\right\}\mathrm{d}\Omega-\int_{\Gamma_2}\boldsymbol{w_h}\cdot\boldsymbol{h}\mathrm{d}\Omega=0, \tag{3.43}$$

$$\int_\Omega\boldsymbol{w_h}\cdot\left[\tilde{\rho}_h^{n+1}\frac{\boldsymbol{u}_h^*-\hat{\boldsymbol{u}}_h}{\Delta t}-\nabla p_h^n\right]\mathrm{d}\Omega=0, \tag{3.44}$$

$$\frac{\Delta t}{\rho_h^{n+1}}\int_\Omega\nabla\boldsymbol{q}_h\cdot\nabla p_h^{n+1}\mathrm{d}\Omega-\int_\Omega\nabla\boldsymbol{q}_h\cdot\boldsymbol{u}_h^*\mathrm{d}\Omega+\int_\Gamma\boldsymbol{q}_h\hat{\boldsymbol{u}}_h\cdot\boldsymbol{n}\mathrm{d}\Gamma=0, \tag{3.45}$$

$$\int_\Omega\boldsymbol{w_h}\cdot\left[\tilde{\rho}_h^{n+1}\frac{\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^*}{\Delta t}+\nabla p_h^{n+1}\right]\mathrm{d}\Omega=0, \tag{3.46}$$

where $\boldsymbol{\delta}^h(\cdot)$ is the SUPG term introduced by Brooks and Hughes (1982), aimed at stabilizing the convective flow to avoid spurious oscillations in solution field of velocity.

Remark. *Evaluation of the density and viscosity given by Equation (3.1) is valid with the assumption that ϕ is a scalar function with value in $[-1,1]$. However, the numerical scheme does not restrict ϕ strictly between -1 and 1. Notice that Equation (3.1) gives a negative ρ when a lower bound of ϕ is reached, i.e. $\phi<-(1+\tilde{\rho})/(1-\tilde{\rho})$. A negative density usually results in severe convergence issues for numerical methods. Consider two examples to see how the lower bound of ϕ changes with the density ratio $\tilde{\rho}$:*

1. *$\tilde{\rho}=0.9$: Here the density ratio is very close to 1, i.e. the fluid components have almost matched densities. This gives the lower bound of ϕ as $-(1+0.9)/(1-0.9)=-19$. This lower bound is extremely far from -1, thus is almost impossible to reach by numerical solution of ϕ. Therefore, Equation (3.1) is a safe evaluation for the case of fluid components with almost matched densities.*

2. $\tilde{\rho} = 0.001$*: In this example, the two fluid components have very large density ratios, e.g. between water and air. The lower bound of* ϕ *is evaluated as* $-(1 + 0.001)/(1 - 0.001) = -1.002$*, which is very close to -1. This value can be easily reached by small perturbation of* ϕ *coming from numerical errors, which will lead to negative value for the density using Equation (3.1). Therefore, Equation (3.1) is not an appropriate evaluation for the density when the density difference is very large between the fluid components.*

The examples show us that for very large density difference between the fluid components, small numerical errors can make the system unstable, making the lower bound of ϕ *much easier to break. Evaluation for viscosity* η *has similar implications. Therefore, to ensure the positive signs of density and viscosity, a normalization procedure is applied on* ϕ *[Dong and Shen (2012)]*

$$\hat{\phi} = \begin{cases} \phi & |\phi| \leqslant 1 \\ sign(\phi) & otherwise, \end{cases} \tag{3.47}$$

where $\hat{\phi}$ *is the normalized value of* ϕ *to be used for evaluating density* ρ *and viscosity* η

$$\rho = \rho_- \frac{1 - \hat{\phi}}{2} + \rho_+ \frac{1 + \hat{\phi}}{2}, \quad \eta = \eta_- \frac{1 - \hat{\phi}}{2} + \eta_+ \frac{1 + \hat{\phi}}{2}. \tag{3.48}$$

Notice that formula (3.47) is employed only for calculating the physical properties, ρ *and* μ*, appearing in the Navier-Stokes equation to avoid numerical instability. The phase field solution* ϕ *obtained from Cahn-Hilliard equation is actually not modified by (3.47), and is stored as it is. Therefore applying Equation (3.47) will not break the mass conservation of fluid components.*

Both Equations (3.40), (3.41) and Equation (3.43) are non-linear equations. Linearization is required to apply the numerical procedures. For both the Cahn-Hilliard equation and the Navier-Stokes equation, the nonlinear terms are treated by the Newton-Raphson method. Here we used the SNES APIs provided by the open source package PETSc [Balay et al. (2013b,a, 1997)], which provides users convenient and powerful tools to solve nonlinear equations. The simulations in this paper are performed on TACC Stampede, and on CyEnce at Iowa State University.

3.5 Validations and examples

3.5.1 Convergence tests

In this section, we perform a validation of the numerical method by using a *manufactured solution* suggested by Dong and Shen (2012)

$$\begin{cases} u &= \sin \pi x \cos \pi y \sin t, \\ v &= -\cos \pi x \sin \pi y \sin t, \\ p &= \sin \pi x \sin \pi y \cos t, \\ \phi &= \cos \pi x \cos \pi y \sin t. \end{cases} \tag{3.49}$$

Though the above expressions of the variables satisfy the continuity Equation (3.26), they do not satisfy the the momentum Equation (3.25) and Cahn-Hilliard Equation (3.27). A remedy is to add an artificial source term to the right hand side of both Equations (3.25) and (3.27) such the above functions are analytical solution to the modified ones. The dimensionless parameters in the governing Equations (3.25)-(3.27) are set to

$$Re = 100, \quad Cn = 0.1, \quad Pe = 10, \quad We = 100, \quad \text{and} \quad Fr = 0. \tag{3.50}$$

The simulation is performed in a rectangular domain, $\Omega = [0, 2] \times [-1, 1]$. Dirichlet conditions for velocity components u and v are determined from the analytical solution (3.49) and are applied on all boundaries. Initial conditions for u, v, p, and ϕ are also described by the analytical solution by setting $t = 0$.

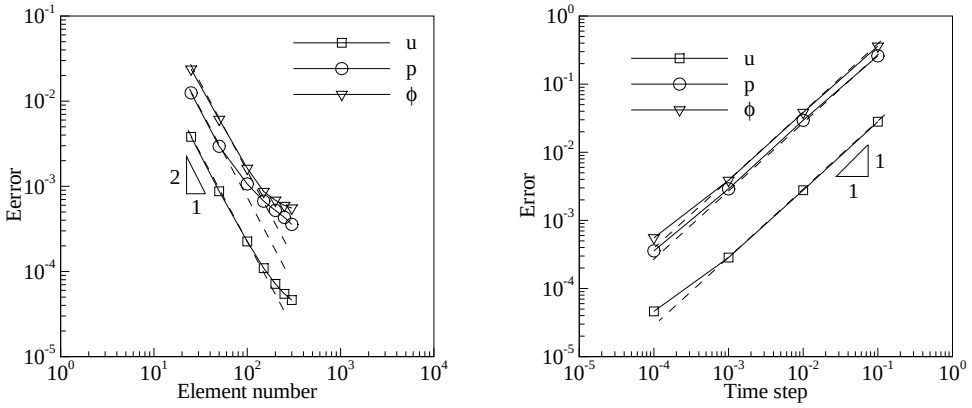
Figure 3.2(a) shows the convergence test on spatial descritization. When applying the spatial convergence test, time step Δt is fixed at 1×10^{-4}, which is small enough to satisfy the CFL condition. The error is computed as

$$\text{error}_u = \frac{\|\boldsymbol{u}_{numeric} - \boldsymbol{u}_{analytical}\|_2}{\|\boldsymbol{u}_{analytical}\|_2}, \tag{3.51}$$

$$\text{error}_p = \frac{\|p_{numeric} - p_{analytical}\|_2}{\|p_{analytical}\|_2}, \tag{3.52}$$

$$\text{error}_\phi = \frac{\|\phi_{numeric} - \phi_{analytical}\|_2}{\|\phi_{analytical}\|_2}, \tag{3.53}$$

which indicates the relative error of numeric solution to the analytical solution. The log-log plot of spatial convergence test shows the slopes of the linear trends of all three lines are around 2,



(a) (b)

Figure 3.2 Temporal and spatial convergence tests. Dashed lines reflect the estimated convergence rates. (a) Convergence of element numbers on one edge; (b) Convergence of time step Δt.

which indicates a second order spatial convergence rate. The drop of the slope for large spatial discretization is due the the saturation of temporal error from constant Δt.

Figure 3.2(b) plots the temporal convergence test results. In this test, grids for all time steps are fixed at 300×300, which give very fine spatial discretization. Similar to the spatial convergence tests, the log-log plot shows clear linear trends for all three variables. The linear parts of all the three lines have slope close to 1. Since the error is integrated at $t = 1$, the unit slope indicates a second order truncated accuracy for each time step.

3.5.2 Capillary wave

Here we use the damped oscillation of an interface between two superposed incompressible viscous fluids with the lighter fluid laying on top to test the accuracy of our numerical scheme and approximation of the surface tension. This problem is also a good test case for flows with large density contrast.

Assume the two fluids have same kinematic viscosity ν. Denote g as the magnitude of gravity, and σ as the surface tension coefficient. The two fluids are imposed with an initial perturbation of a sinusoidal function at the interface with a small amplitude H_0 and wave

number k. No initial velocity is applied at the interface. Then the evolution of the amplitude of the interface, $H(t)$, has an analytical form given by Prosperetti (1981):

$$\frac{H(t)}{H_0} = \frac{4(1-4\beta)\nu^2k^4}{8(1-4\beta)\nu^2k^4+\omega_0^2}\text{erfc}\left(\sqrt{\nu k^2 t}\right) + \sum_{i=1}^{4}\frac{z_i}{Z_i}\frac{\omega_0^2}{z_i^2-\nu k^2}e^{(z_i^2-\nu k^2)t}\text{erfc}\left(z_i\sqrt{t}\right), \tag{3.54}$$

where

$$\omega_0^2 = \frac{(\rho_+-\rho_-)gk+\sigma k^3}{\rho_++\rho_-}, \quad \beta = \frac{\rho_+\rho_-}{(\rho_++\rho_-)^2}, \tag{3.55}$$

and erfc$(\cdot)$ is the complementary error function. The variables z_i, $i = 1, \cdots, 4$, are the four complex roots of equation

$$z^4 - 4\beta\sqrt{\nu k^2 z^3} + 2(1-6\beta)\nu k^2 z^2 + 4(1-3\beta)(\nu k^2)^{3/2}z + (1-4\beta)\nu^2k^4 + \omega_0^2 = 0, \tag{3.56}$$

where

$$Z_i = \prod_{1\leqslant j\leqslant 4, j\neq i}(z_j - z_i), \quad i = 1, \cdots, 4. \tag{3.57}$$

The motion of the interface is simulated in a rectangular domain, where the top and bottom boundaries are far enough from the interface to eliminate end effects. Taking the wave length of the interface oscillation as the characteristic length, and imposing the initial shape of the interface as a sinusoidal function, $y_c = 0.01\cos(2\pi x)$, the rectangular domain is chosen as $[0, 0.5] \times [-1, 1]$ due to periodicity and symmetry of motion, which is the right half of the domain in one period. Notice that the distance from the interface to the top or bottom end is 100 times of the amplitude of oscillation. A no-slip boundary condition for velocity components is applied on the top and bottom boundaries. A no-flux condition is applied on the two side walls. Reynolds number Re, Weber number We and Froude number Fr are chosen as 100, 1 and 1, respectively.

Effect of interface discretization

Spatial discretization plays an important role in determining the accuracy of the diffusive interface model. Wodo and Ganapathysubramanian (2011) computationally validated that at least four elements across the interface (where ϕ is approximately between -0.9 to 0.9) should be maintained to ensure that the diffusive model performs correctly. However, they performed

the tests only on the Cahn-Hilliard equation without the convective term nor coupled with the Navier-Stokes equation. We perform a spatial discretization convergence test to investigate how the diffuse interface model is affected by the element size. We fix the time step increment Δt to a small value, i.e. $\Delta t = 1 \times 10^{-4}$, to satisfy the CFL condition. Density and viscosity ratios are both set to unit 1 to simplify the tests. Cahn number Cn is selected as 0.005, which suggests a value of 200 for the Péclet number Pe as the inverse of Cn. We distinguish the resolutions of spatial discretization by counting the number of elements through the interface (i.e. $\phi \in [-0.9, 0.9]$), where there are 4 cases studied here, 3, 4, 5, and 6 elements per interface.

Figure 3.3 shows the evolution of amplitude $H(t)$ related to different spatial resolutions, where the theoretical solution given by Equation (3.54) is also given for comparison. The plot clearly shows that three elements per interface is not enough for the diffuse interface model to capture the physical properties. At least four elements per interface is required to persist the damping oscillation features and to match with the theoretical frequency, and increasing numbers of elements give a more accurate solution. Table 3.1 lists the details of the errors all four cases. The absolute L_∞ (L_2) error is the L_∞ (L_2) norm of the difference between the numerical solution and the theoretical solution over the whole time range. The relative L_∞ (L_2) error is the ratio between the absolute L_∞ (L_2) error and the L_∞ (L_2) norm of the theoretical solution. The numbers clearly show that the simulations with at least four elements per interface have significantly smaller errors than the simulation with only three elements per interface. This test reveals that the resolution of spatial discretization has crucial influence on the correctness and accuracy of the governing equation system (3.25) - (3.27), where we need at least four element through the interface. This agrees very well with the conclusion obtained by Wodo and Ganapathysubramanian (2011), where the authors analyzed the mesh size influence on the accuracy of coupled phase separation and coarsening process for the Cahn-Hilliard equation.

Effect of relative interfacial thickness

The interface thickness also controls the accuracy of simulation. Ceniceros et al. (2010) proved that the solution to the diffusive model converges to the true solution as interface

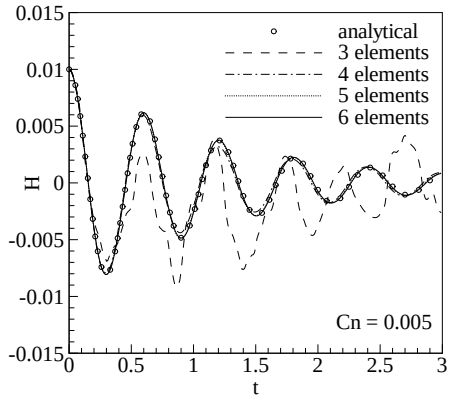


Figure 3.3 Evolution of interface oscillation magnitude with various number of elements per interface. The Cahn number Cn is fixed at 0.005 for this test.

Table 3.1 Relative L_∞ and relative L_2 errors for the interface discretization tests.

elements per interface	relative L_∞ error	relative L_2 error
3	0.677	0.983
4	0.0729	0.110
5	0.0266	0.0423
6	0.0132	0.0203

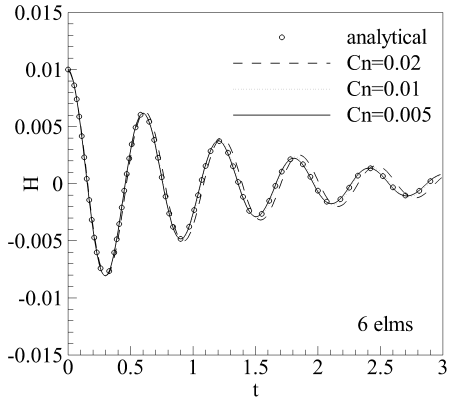


Figure 3.4 Effect of Cahn number Cn on the evolution of interface oscillation magnitude. These cases share the same element density as 6 elements per interface.

thickness ϵ tends to 0, though it is possible that a thin interface limit exists for the diffusive interface model [Yue et al. (2010)]. Therefore a convergence test on interface thickness is necessary to understand the performance of the diffusive interface model on a given problem. The interface thickness is linearly proportional to the Cahn number. Here we collected solutions from simulations with three different Cahn numbers, $Cn = 0.02$, $Cn = 0.01$, and $Cn = 0.005$, while fixing six elements per interface to reduce the element size effect. Other parameters are the same as above spatial convergence test. Comparison is given in Figure 3.4. The comparison clearly shows that the numerical results are approaching the theoretical solution as the Cahn number reduces. Details of the errors are listed in Table 3.2, where we see the error smoothly dropping as we reduce the Cahn number. We oberve no evidence of the existence of a minimum requirement on Cahn number to ensure the physical correctness, as was seen in the interface discretization test (3.3). This further verifies that the diffuse interface model converges towards the thin interface limit as the interface thickness reduces. Figure 3.4 and Table 3.2 together also numerically reveal that an interfacial thickness around 1% of the characteristic length provides very good agreement between the diffuse interface model and the sharp interface solution (where we see that the relative errors go below 0.05 when $Cn \leq 0.01$), when fluid components have matched densities.

Table 3.2 Relative L∞ and relative L2 errors for the convergence tests of Cahn number Cn.

Cn	relative L∞ error	relative L2 error
0.02	0.113	0.193
0.01	0.0231	0.0365
0.005	0.0131	0.0202

Effect of large density ratio

A highlight of the current numerical scheme is its ability to simulate flows of fluids with large density and viscosity ratios. To verify this feature, simulations of an interface with two fluids of different densities and viscosity are studied in this section. One assumption required for the solution (3.54)is that kinematic viscosity should be the same for the two fluids; we keep it at 0.01 here. Here we consider three density ratios, 10, 100, and 1000. Six elements per interface is imposed for the simulation to obtain accurate solutions. The time step increment is still kept as a small constant, $\Delta t = 1 \times 10^{-4}$. Results are shown in Figure 3.6. From the figure we see that the oscillation with larger density ratio has reduced damping effect with a longer oscillating period and slower damping rate. These features are accurately captured by the model.

The Cahn number affects the accuracy of numerical solution. However, when the Cahn-Hilliard equation is coupled with the hydraulic equation, the involvement of the density difference affects the behavior of the convergence rate with respect to the Cahn number Cn. The capillary wave simulation is performed with different Cahn numbers at various density ratios to reveal the effect of the density ratio on the convergence. We keep the grid resolution six elements through the interface, and keep $\Delta t = 1 \times 10^{-4}$, to control the numerical error from spatial and temporal discretizations. Figure 3.6(a) plots the relative L2 error of the numerical solutions with Cn in the rage of 0.005 to 0.02 at relative small density ratios. The plot shows that the convergence rate with increasing Cn (seen as the slope of a curve in the log-log plot) drops as the density ratio increases, but still has a significant absolute value for the small density ratios (i.e. $\tilde{\rho} < 40$). Figure 3.6(b) gives the convergence as a function of Cn for the fluids with relatively larger density ratios (i.e. $\tilde{\rho} \geq 40$). We see from the figure that the convergence

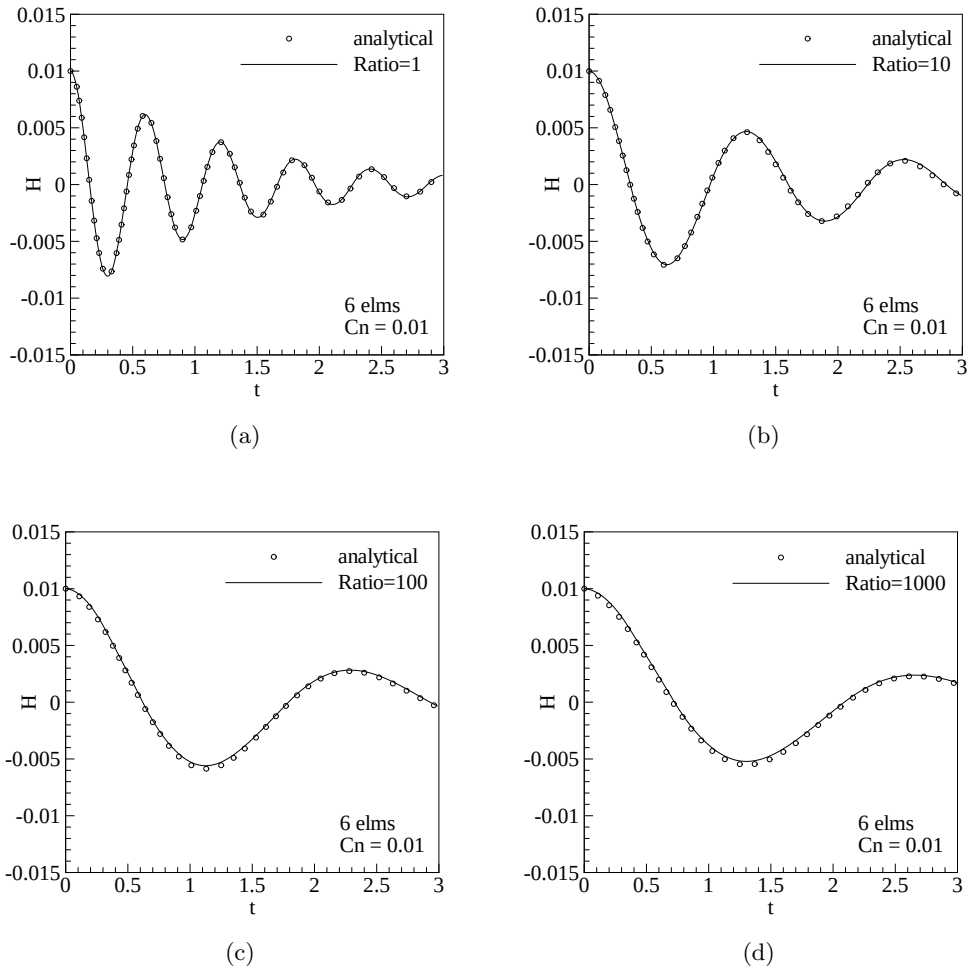


Figure 3.5 Evolution of interface oscillation with different density and viscosity ratios. (a) Same density/viscosity ratio; (b) $\rho_+ : \rho_- = \mu_+ : \mu_- = 10 : 1$; (c) $\rho_+ : \rho_- = \mu_+ : \mu_- = 100 : 1$; (d) $\rho_+ : \rho_- = \mu_+ : \mu_- = 1000 : 1$;

with respect to Cn significantly slows down compared with the cases with smaller density ratios. Combining the two plots, we see that the convergence rate as a function of Cn has an exponentially decreasing trend with increasing density ratio.

To quantitatively understand how the density ratio affects convergence as a function of Cn, the exponential convergence rates are fitted from the curves seen in figures 3.6(a) and 3.6(b) at all density ratios, and are plotted as the "dots" in Figure 3.6(c). As the density ratio increases ($\tilde{\rho} \geqslant 10$), the convergence rate changes nearly along an exponential trend. Fitting the data with an exponential curve ("power-fit"), we obtain that the order of the convergence rate reduction is about 1.41 with density ratio. This result is not surprising, since larger density ratio implies stronger inertial effects, which is governed more by the Navier-Stokes equation. Thus the numerical error from the hydraulic equation overtakes the one from the Cahn-Hilliard equation, which is mainly controlled by the interfacial thickness for the diffuse interface model. This result provides us with a heuristic suggestion for simulations using the diffuse interface model, that the Cahn number Cn may be chosen from a relatively wider range of values for the fluids with large density ratios, which will help reduce the computational resources without sacrificing the accuracy.

We next consider cases when interfacial effects become weak compared with the inertial effects.

3.5.3 Rayleigh-Taylor instability

Rayleigh-Taylor instability reflects the instability between two fluids when heavier fluid lays on top of lighter fluid. Driven by the gravity, the top heavier fluid dives down and the lighter fluid rises up. Tryggvason (1988) investigates this problem but with an assumption of zero viscosity. Guermond and Quartapelle (2000) later extended the simulation with viscous flow. Since the problem is more interesting for macro-scale flows where the effect of surface tension becomes less important than the convective and gravitational effects, the surface tension in the momentum Equation (3.25) can be neglected. A dimensionless parameter, the Atwood ratio, defines the density ratio between the two fluids as $At = (\rho_+ - \rho_-)/(\rho_+ + \rho_-)$, e.g. $At = 0.50$ when density ratio $\tilde{\rho} = \rho_-/\rho_+ = 1/3$.

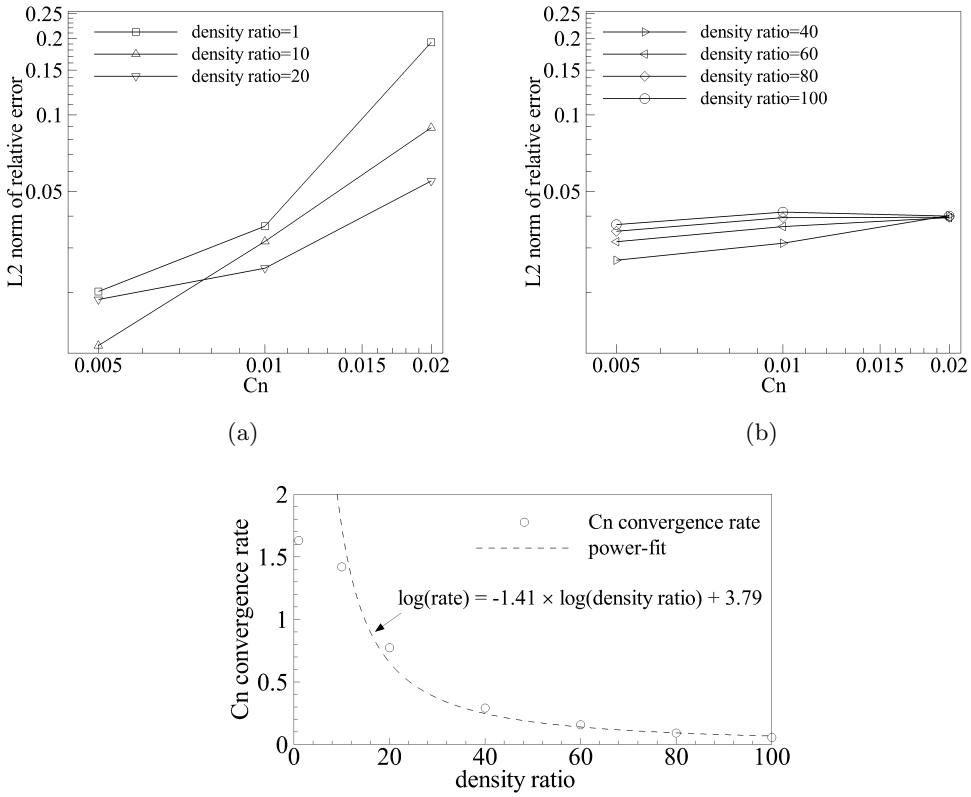
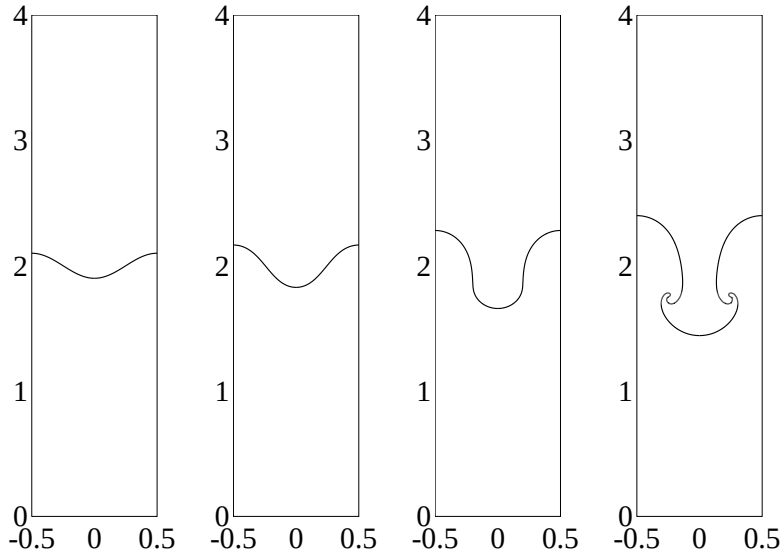


Figure 3.6 (a) Curves reflect the convergence of Cn in the range of $[0.005, 0.02]$ for small density ratios ($\tilde{\rho} < 40$); (b) Curves reflect the convergence of Cn in the range of $[0.005, 0.02]$ for relative large density ratios ($\tilde{\rho} \geq 40$); (c) Trend of the convergence rate of Cn with density ratio is fitted with a power law curve. The fitted curve matches very well with the density ratio between 10 and 100, which shows an exponentially reducing trend of the convergence rate with density ratio by an order of 1.41.

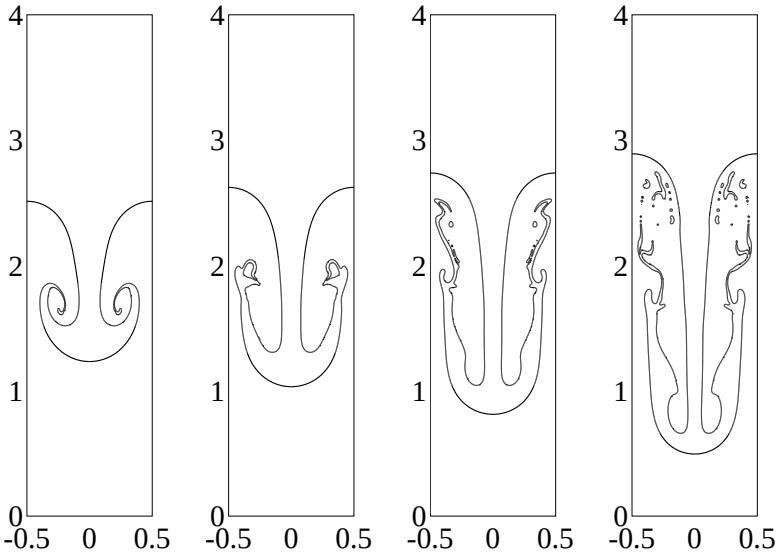
In this paper, we first illustrate the Rayleigh-Taylor instability by simulating the two fluids with $At = 0.50$. The physical domain is set as $\Omega = [-d/2, d/2] \times [0, 4d]$, where d is channel width as the characteristic length. Because of the symmetry of the flow, the simulation is applied in the right half of Ω, i.e. $[0, d/2] \times [0, 4d]$. The top and bottom boundaries are applied with no-slip boundary conditions, while the two vertical ones are set as slip boundaries. An initial perturbation is applied on the interface between the two fluids, where the initial position of the interface is represented as $y_c = 2d + 0.1d\cos(2\pi x/d)$. There is no initial velocity set for the simulation. The Reynolds number Re is fixed at 3000 to compare with the previous simulation results [Guermond and Quartapelle (2000)]. The interface thickness is $0.0025d$, which corresponds to a very small Cahn number $Cn = 0.0025$ to capture the details of the interface motion. The computational domain is discretized into 200×1600 elements, which gives four elements across the interface. Time step increment is fixed at $\Delta t = 1 \times 10^{-4}$. However, when illustrating the results, we convert the dimensionless time to the Tryggvason form [Tryggvason (1988)] defined as $t_{\text{Tryg}} = t\sqrt{At}$, where t is the conventional dimensionless time given above.

Snapshots of the interface positions are shown in Figure 3.7, at dimensionless time $t = 0.00$, 0.43, 0.86, 1.29, 1.71, 2.14, 2.57 and 3. From the plots we see that as the front of the heavier fluid sinks downward, the lighter fluid rises up on the sides. As the sinking process accelerates, the two sides of the front curve squeeze inward due to the shear flows generated from the convection between the two fluids. When the convective effect continues, small vertices grow into larger ones and essentially shed and form vortex street patterns in the wake. Figure 3.8(a) records the evolution of the vertical positions of the top of the rising fluid and the bottom of the sinking fluid. The accuracy of the current solver is quantitatively verified by matching with previous results [Tryggvason (1988); Guermond and Quartapelle (2000); Ding et al. (2007)].

We also performed the simulation for two more density ratios, $\tilde{\rho} = 0.1$, i.e. $At = 0.82$, and $\tilde{\rho} = 0.01$, i.e. $At = 0.98$. The initial conditions and other numerical settings are kept the same as the first case $At = 0.50$. The dynamic viscosities of the two fluid components are still fixed the same for these two cases. Figure 3.9 shows snapshots of the case $At = 0.82$, using the Traggvason dimensionless time. Compared with the case $At = 0.50$ (3.8(a)), the larger density

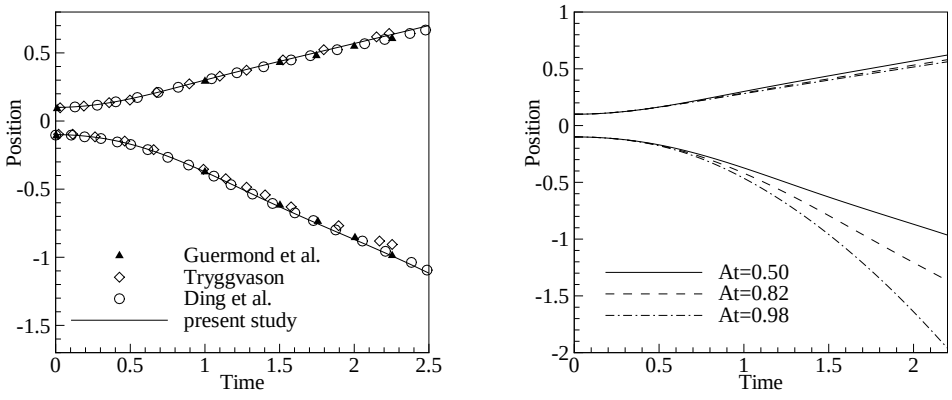


(a) $t = 0.00$ (b) $t = 0.43$ (c) $t = 0.86$ (d) $t = 1.29$



(e) $t = 1.71$ (f) $t = 2.14$ (g) $t = 2.57$ (h) $t = 3.00$

Figure 3.7 Rayleigh-Taylor instability. Snapshots of the interface at different time for the case $At = 0.50$. Tryggvason form [Tryggvason (1988)] of dimensionless time, $t_{\text{Tryg}} = t\sqrt{At}$, is used in the plots where t is the conventional dimensionless time.



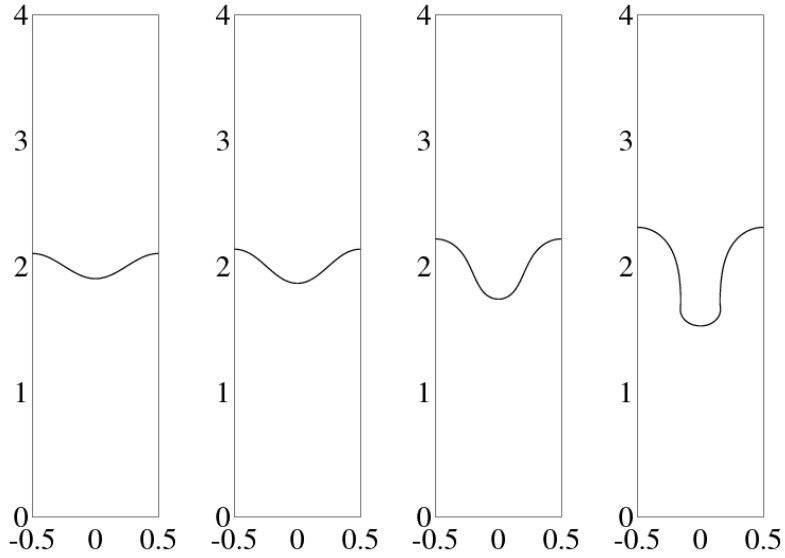
(a) (b)

Figure 3.8 (a) Rayleigh-Taylor instability. The upper part and the lower part record the evolution of the peak of the rising fluid and the bottom of the falling fluid, respectively. The present study is compared with Tryggvason (1988), Guermond and Quartapelle (2000), and Ding et al. (2007). The Tryggvason form [Tryggvason (1988)] of dimensionless time, $t_{\text{Tryg}} = t\sqrt{At}$, is used in the plots where t is the conventional dimensionless time; (b) Comparison of the evolution of the rising and falling parts for different density ratios, where $At = 0.50$, $At = 0.82$, and $At = 0.98$.

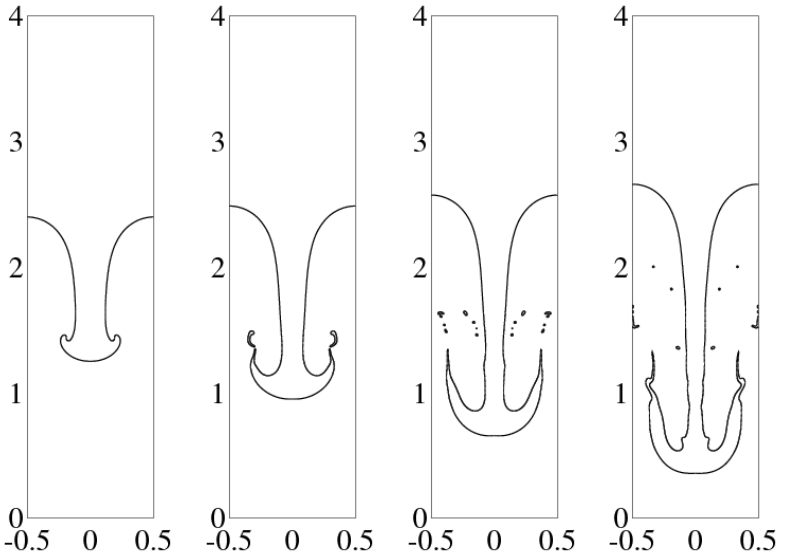
ratio induces a relatively narrower falling column and a smaller front of the heavier fluid (e.g. at $t = 1.45$). There are also a pair of vortices formed right behind the falling front (e.g. at $t = 1.81$), which do not curve inward as significantly as they do for the $At = 0.50$ case. As At increases to 0.98, snapshots plotted in Figure 3.10 show a quite different scenario. In this case, the falling fluid forms a very narrow spike all the way down until touching the bottom. The front of the falling part preserves a round shape and there are no vortices pair observed for this flow with very large density ratio. Both these two simulations give very good agreement with the results in Tryggvason (1988) for flows with large At's. The comparison of the positions of the falling and rising fronts for these three cases ($At = 0.50$, $At = 0.82$, and $At = 0.98$) are given in Figure 3.8(b).

3.5.4 Droplet impact on liquid surface

The impact of a droplet on a liquid surface has long been an interesting problem, and challenging due to the complicated mechanism of interaction between the droplet and liquid

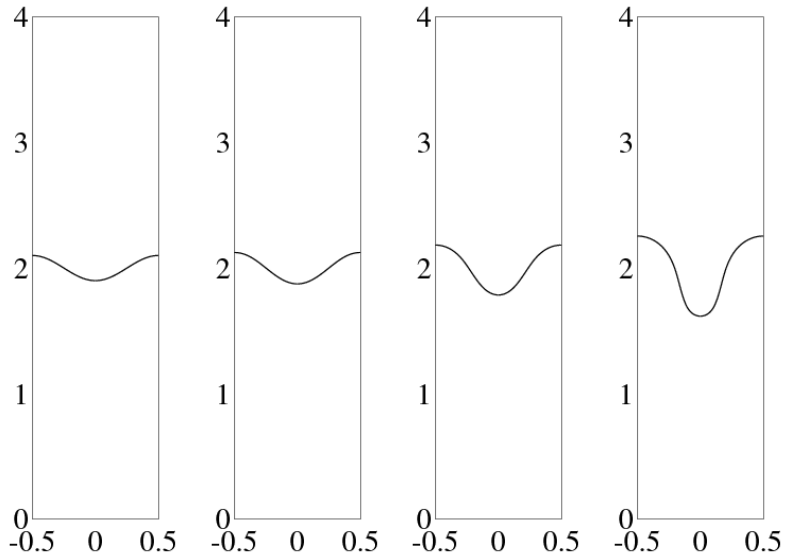


(a) $t = 0.00$ (b) $t = 0.36$ (c) $t = 0.72$ (d) $t = 1.09$

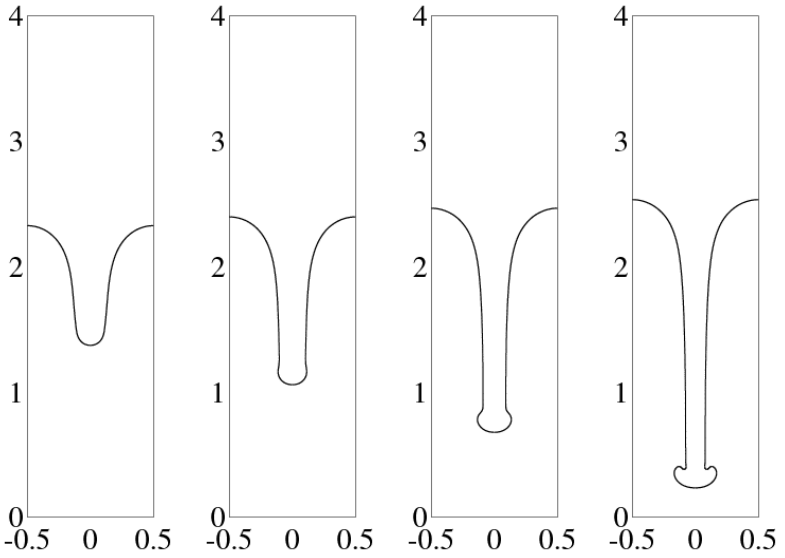


(e) $t = 1.45$ (f) $t = 1.81$ (g) $t = 2.17$ (h) $t = 2.53$

Figure 3.9 Rayleigh-Taylor instability. Snapshots of the interface at different time for the case $At = 0.82$ ($\tilde{\rho} = 0.1$). Tryggvason form [Tryggvason (1988)] of dimensionless time, $t_{\text{Tryg}} = t\sqrt{At}$, is used in the plots where t is the conventional dimensionless time.



(a) $t = 0.00$ (b) $t = 0.30$ (c) $t = 0.59$ (d) $t = 0.89$



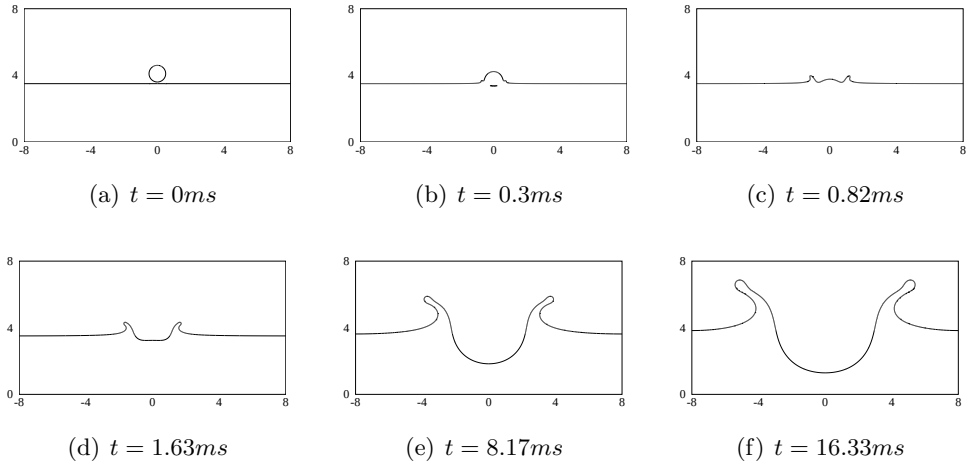
(e) $t = 1.19$ (f) $t = 1.49$ (g) $t = 1.78$ (h) $t = 2.08$

Figure 3.10 Rayleigh-Taylor instability. Snapshots of the interface at different time for the case $At = 0.98$ ($\tilde{\rho} = 0.01$). Tryggvason form [Tryggvason (1988)] of dimensionless time, $t_{\text{Tryg}} = t\sqrt{At}$, is used in the plots where t is the conventional dimensionless time.

surface. Yarin (2006) reviewed and described most of the phenomena observed during impact, like spreading, recoiling, jetting, splashing (crowning formation), and etc. Shapes and behaviors of the liquid surface resulting from the impact are strongly affected by the physical properties of the involved liquids. Deng et al. (2007) experimentally studied the importance of viscosity and surface tension during the impact process. Hasan and Prosperetti (1990) numerically investigated the influence of droplet size and impacting velocity on the air bubble entrapment process. A good approximation of surface tension is essential for obtaining accurate numerical results. However, few studies used the diffuse interface model to simulate such liquid-liquid impact processes. In this paper, we simulate the impact of a glycerine droplet on the surface of the same liquid exposed to air.

Here we illustrate the droplet impact on the liquid surface in the 2D case. Simulation is carried out in a rectangular domain, $\Omega = [-8D, 8D] \times [0, 8D]$, where D is the initial diameter of the droplet, which is selected as the characteristic length. But as before, only the right half of the domain, $[0, 8D] \times [0, 8D]$ is actually simulated, due to symmetry. The flat, free surface is imposed at the position where $y_l = 3.5D$. Experimentally, the droplet usually free falls to the liquid surface from a higher position, and the droplet is accelerated by gravitational force. We start the simulation at the moment when the droplet is right above the liquid surface assuming it maintains a spherical shape. This initial state is different from the typical droplet impact on a solid surface where the initial condition is usually taken as the moment the droplet touches the surface. The reason we choose the moment when there is still a small gap between the droplet and the free surface is that the air cushion below the droplet will be squeezed down and push the free surface inward, which will influence the formation of the jet generated from the impact. We take the initial gap as $0.1D$ for this simulation.

The impact velocity of the droplet is set as the characteristic velocity U, where U=3m/s in this paper. Droplet initial diameter D is selected as the characteristic length, and D=2.45mm. Given the thermal properties of glycerine and air, and gravitational acceleration $g = 9.8\text{m/s}^2$, the dimensionless parameters are the Reynolds number $Re = 80$, the Weber number $We = 512$, and the Froude number $Fr = 800$. The Cahn number Cn is chosen as $Cn = 0.01$, and the



(a) $t = 0ms$ (b) $t = 0.3ms$ (c) $t = 0.82ms$

(d) $t = 1.63ms$ (e) $t = 8.17ms$ (f) $t = 16.33ms$

Figure 3.11 2D Droplet impact on a liquid surface. The 2D glycerin droplet has a spherical shape with diameter of 2.45mm, and impact velocity of 3m/s. Snapshots are plotted at different times.

Péclet number Pe is then $Pe = 100$. The corresponding grid size is 800×800, which gives four elements per interface, satisfying the requirement of diffuse interface model.

Snapshots of the impact process are shown in Figure 3.11. As revealed by the plots, the small jets of liquid-liquid impact form in a very short period during the initial stage of the impact process. Then under inertia, the droplet totally merges into the lower liquid and pushes the free surface down. The two sided jets grow taller and wider then form round tips on top under the effect of surface tension. These jet shapes are typically seen for the impact of more viscous fluids, well known because of the very famous photo of the "milk crown," taken by Edgerton and Killian (1954), which has been further investigated with other shapes of crowns experimentally in Krechetnikov and Homsy (2009).

3.5.5 3D droplet impact on solid surface

Droplet impact on a solid surface is the prototype of many engineering applications, e.g. surface coating, spray drying and ink-jet printing. Spreading is mainly determined by material properties, impact velocity, substrate inclination, and surface characteristics like contact angle and roughness [Lunkad et al. (2007)]. The contact angle is commonly represented by the static contact angle (SCA). Later research [Šikalo et al. (2005); Bussmann et al. (1999)] developed

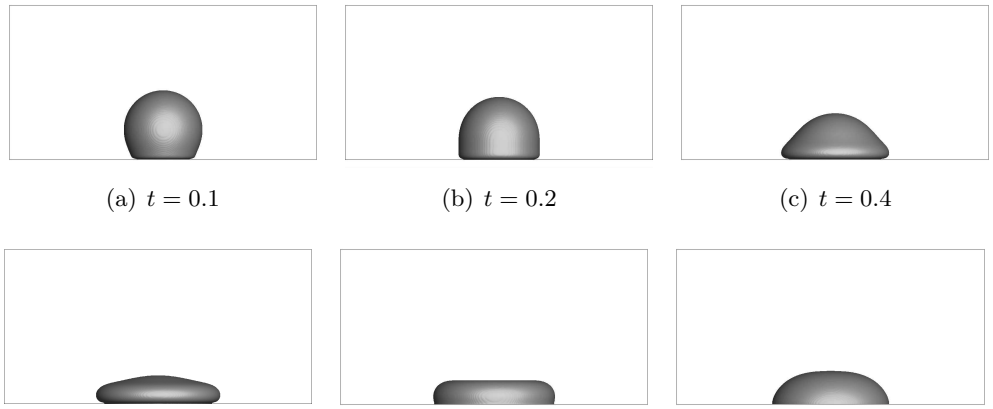
the dynamic contact angle (DCA) model and reported its accuracy in capturing the droplet spreading on solid surfaces. However, Lunkad et al. (2007) compared the SCA and DCA models and found that both models give very good agreement with experiment for non-wetting surface, i.e. SCA $\geqslant 90°$. In this paper, we focus only on non-wetting surfaces and apply the static contact angle model.

We simulate a 3D glycerine droplet impact on a solid surface in a box domain. The physical domain Ω is $\Omega = [-2D, 2D] \times [-2D \times 2D] \times [0, 2D]$, where D is the diameter of the initial droplet assuming a spheric shape. Axial symmetry allows the simulation to run in the first quarter of the domain, i.e. $[0, 2D] \times [0 \times 2D] \times [0, 2D]$. The initial position of the falling sphere is taken as the moment when the droplet touches the solid surface. The Reynolds and Weber numbers are 26.7 and 51, respectively. The Froude number is 43.8. The Cahn number is fixed as 0.01, which gives a $200 \times 200 \times 200$ discretization of the computational domain. In this example, we consider the case where the static contact angle $\theta_S = 93.5°$. The time step increment is set as a constant, $\Delta t = 1 \times 10^{-3}$.

Snapshots of the droplet spreading at different times are showcased in Figure 3.12. Seen from the result, the wetting spot diameter grows quickly in the initial stage of impact. The widest part is not at the contact line of the droplet, but at a position just above the contact surface. This phenomenon is also observed by Lunkad et al. (2007). Then the spreading process slows, and the shape of the droplet reaches a steady hemispheric form under surface tension where the system has minimum energy. A comparison of the evolution of the wetting spot diameter between the current numerical result and previous experiments is given in Figure 3.13, where the evolution of the droplet height from our simulation is also shown, though there are no available experimental data. The droplet spreading is very well resolved by our method.

3.5.6 3D droplet impact on patterned substrate

The surface topology also plays an important role on the droplet impact process. The surface wetability can be significantly affected by the surface structures thus giving different spreading and receding patterns of droplets. There are manny types of this phenomenon in nature. The most famous are the “lotus effect” and “shark skin effect.” The lotus leaf has a nano-scale



(a) $t = 0.1$ (b) $t = 0.2$ (c) $t = 0.4$

(d) $t = 0.8$ (e) $t = 1.5$ (f) $t = 3.0$

Figure 3.12 Droplet impact on a solid surface. Snapshots at different time.

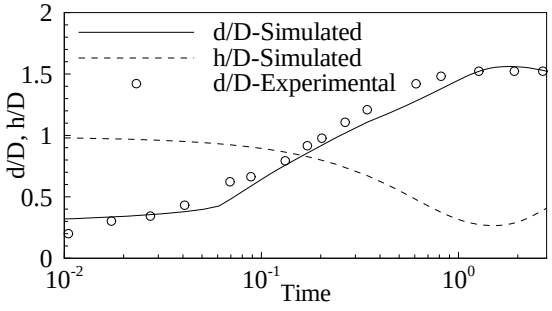
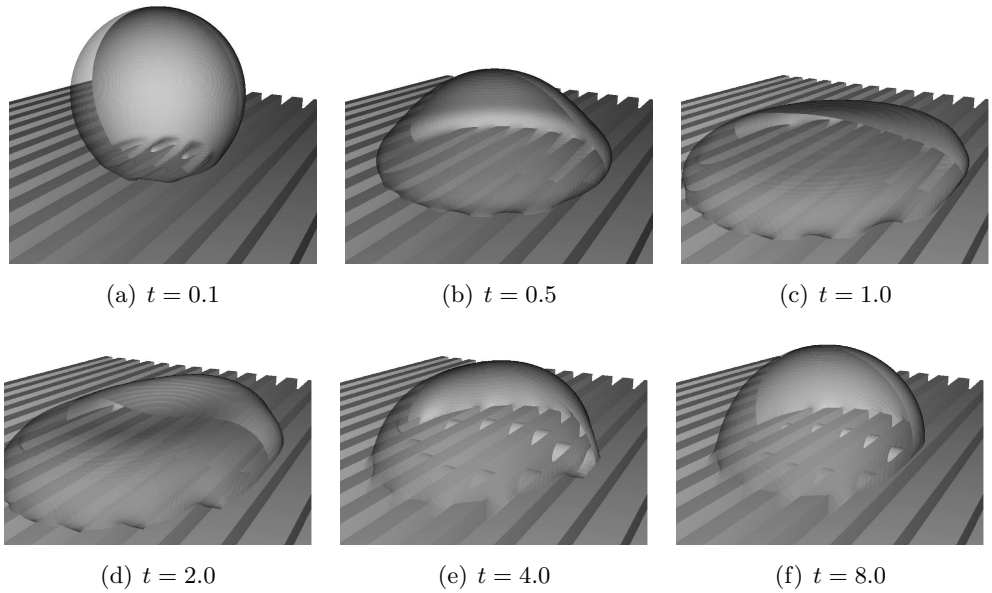


Figure 3.13 Droplet impact on a solid surface. Evolution of droplet wetting spot and droplet height by time, compared with experimental data in Šikalo et al. (2005).

structure which is extremely hydrophobic and gives the leaf a self-cleaning property. The shark skin is cover with "dermal denticles," which are teeth-like small scales, which can significantly reduce the drag. Liu and Li (2012) experimentally studied these two effects by replicating the shark skin structure with artificial materials. Structured surfaces are also significant in industrial coating and painting processes. Ahmed and Rangel (2002) numerically investigated the metal droplet impact and solidification on a wavy surface using a 2D axially symmetric model, for the droplet size around 100 μm and impact velocity of 100m/s. They observed that increasing the surface roughness improves droplet spreading and solidification. Parizi et al. (2007) simulated droplet impact on a patterned surface with VOF method and compared how surface patterns and roughness influence the final shape of droplet splat. However, the conclusions obtained from above simulations are more appropriate for very small droplet with very high impacting velocity. Kannan and Sivakumar (2008) experimentally studied droplet impact on grooved surface with droplet diameter about 3mm and impacting velocity within 10m/s. They measured how the droplet diameter evolves in parallel and perpendicular to the micro groove directions with various impacting velocities. Xu (2007) experimentally investigated the droplet splashing process on a surface with square lattice texture with droplet diameter of 3.4mm and impacting velocity of 4.3m/s. The author found that the prompt splashing forms a clear four-fold symmetry mostly in the diagonal directions of the square lattice.

Few studies have simulated droplet impact on patterned surfaces with the diffuse interface model. Here we showcase how our model can be applied to droplet impact simulation on surfaces with complex geometries. Two types of substrate patterns are illustrated here, grooved pattern and checkered pattern. The grooved pattern is formed by placing periodic grooves on the substrate with width and depth $0.1D$ and distance $0.1D$ to each other. The checkered pattern is formed by placing $0.1D \times 0.1D \times 0.1D$ micro cube pillars at a distance of $0.1D$ next to each other. The numerical configurations and dimensionless parameters for this problem are the same as the ones for the above 3D droplet impact on a smooth solid surface. As before, the simulations are carried out only in the first quadrant of the domain due to symmetry.

Figure 3.14 gives the 3D snapshots of the spreading process taken at different times, where the front interface facing the readers is made transparent so we are able to investigate the details



(a) $t = 0.1$ (b) $t = 0.5$ (c) $t = 1.0$

(d) $t = 2.0$ (e) $t = 4.0$ (f) $t = 8.0$

Figure 3.14 Droplet impact on a groove patterned solid surface. Snapshots at different dimensionless time. It is clearly seen that the lower interface in the grooves formed during the spreading process breaks into several arches with air trapped inside.

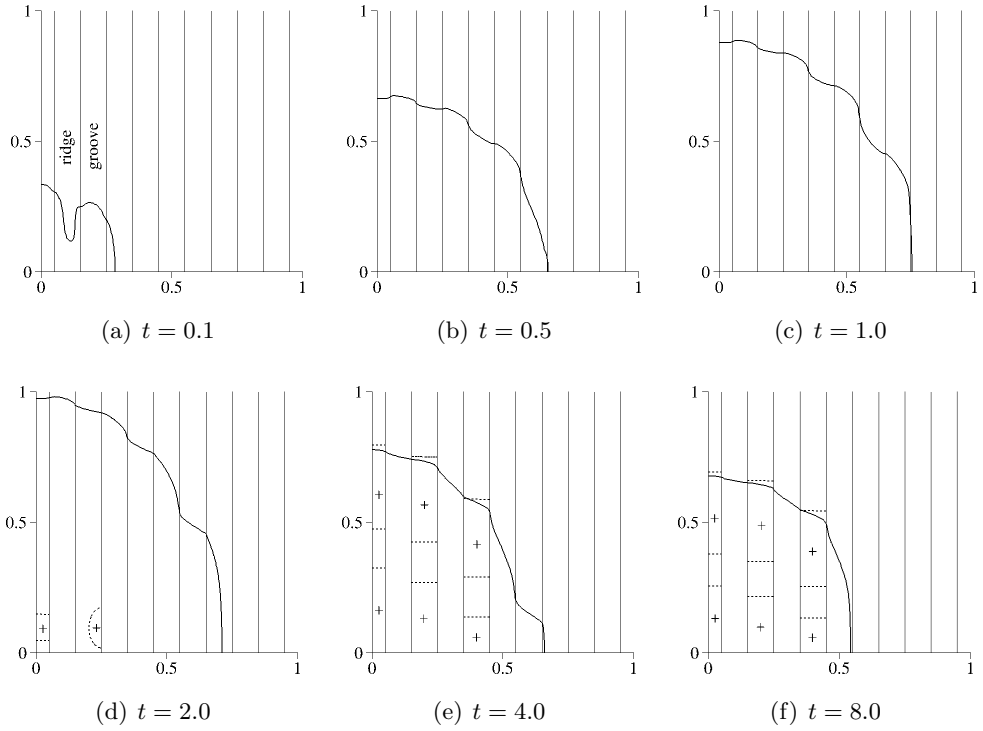
of the interface propagation occurring inside and above the patterned substrate. Similar to the case of spreading on a flat surface, the droplet deforms dramatically from spherical shape to flat shape in the beginning stage of spreading, and gradually reforms to a hemispherical shape in the receding stage. In the spreading stage, the lower interface of the droplet slightly sinks into the grooves and paves along the direction of the grooves without touching the bottom. However, in the receding stage, the parallel lower interface is broken due to instability related to surface tension and the dimension of grooves. The lower interface curves downwards and breaks into small segments with air trapped inside. We see that those air cells form in a semicircle arch shape, which minimizes the interface surface energy. It is possible to form various air entrapment patterns inside the grooves by changing the size of groove. We postpone this investigation to further study.

To better understand the droplet wetting process on the groove-patterned surface, we plot the details of the propagation of wetting spots in Figure 3.15. Each plot in Figure 3.15 forms a top-view of the patterned bottom surface in the first quadrant ($x \geqslant 0, y \geqslant 0$), where the

positions of grooves and ridges are indicated in Figure 3.15 (a). Curves in the plots record the snapshots of the borders of the droplet impacting the bottom. The solid line is the intersection between the droplet interface and the plane right on top of the grooves (with the same level of the top surface of the ridges), indicating the shape of the droplet right above the grooves. The dashed line is the intersection between the droplet interface and the bottom surfaces of the grooves, reflecting the impacting situation inside the grooves. The regions marked with the "+" signs surrounded by the dashed lines and edges of grooves point out the wetting areas on the bottom surfaces of the grooves, where the droplet touches the lower bottom. The regions without the "+" signs among the dashed lines and the edges of grooves indicate the positions where air bubbles are entrapped.

From 3.15 we clearly see how the droplet wets the groove-patterned surface. During the spreading stage (Figure 3.15 (a) - (c)), there is no dashed line appeared in the plots, which indicates that the droplet has not reached the bottom of the grooves due to its high inertia during this stage. When the spreading process slows down (figure3.15 (d)), the wetting spots begin to appear on the bottom of the grooves. Figure 3.15 (d) clearly present the formation of wetting spots in the grooves. The semicircle region marked with a "+" sign shows that the droplet creeps down to the bottom of the groove from one side, and reaches towards the other side and eventually forms a rectangular zone. The receding stage takes much longer than the spreading stage, where we see that the droplet wetting region above the grooves shrinks towards a hemispherical shape. It is interesting to notice that the grooves drag the droplet towards the direction of the grooves and gives the droplet a elliptic wetting spot. We also notice that the entrapped air stays in the grooves and the wetting spots on the bottoms of the grooves keep a rectangular shape.

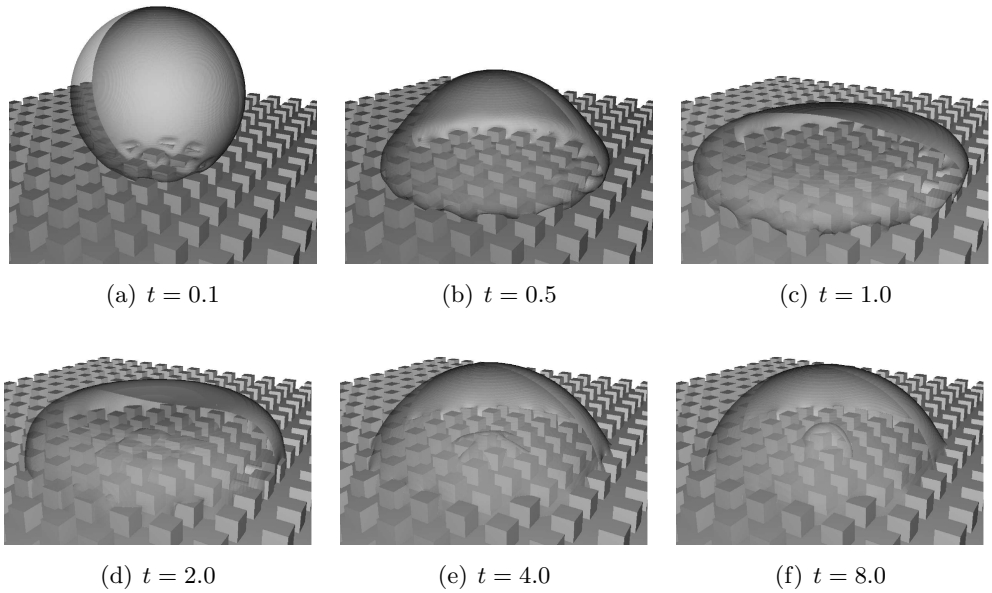
Figure 3.16 illustrates the droplet impact on a checker patterned substrate, by showing snapshots of the interface at different times. As above, the interface is made transparent on the front side. The lower interface paves among the cube pillars and forms a parallel layer above the solid bottom in the spreading process. When the droplet top interface begins to recede, the bottom layer curves downward as happens in the case of the grooved pattern. However, unlike the grooved pattern, the breaking bottom interface pieces are able to rejoin with other parts



(a) $t = 0.1$ (b) $t = 0.5$ (c) $t = 1.0$

(d) $t = 2.0$ (e) $t = 4.0$ (f) $t = 8.0$

Figure 3.15 Formation of the wetting spots above and inside the grooves during the impacting process. Positions of grooves and ridges are pointed out in (a). Solid line indicates the intersection between the droplet interface and solid surface above the grooves. The dashed line indicates the intersection between the droplet interface and the bottom surfaces of the grooves. The regions marked with the "+" signs surrounded by the dashed lines and edges of grooves indicate the wetting areas on the bottom surfaces of the grooves, while the areas in between indicate air entrapment.



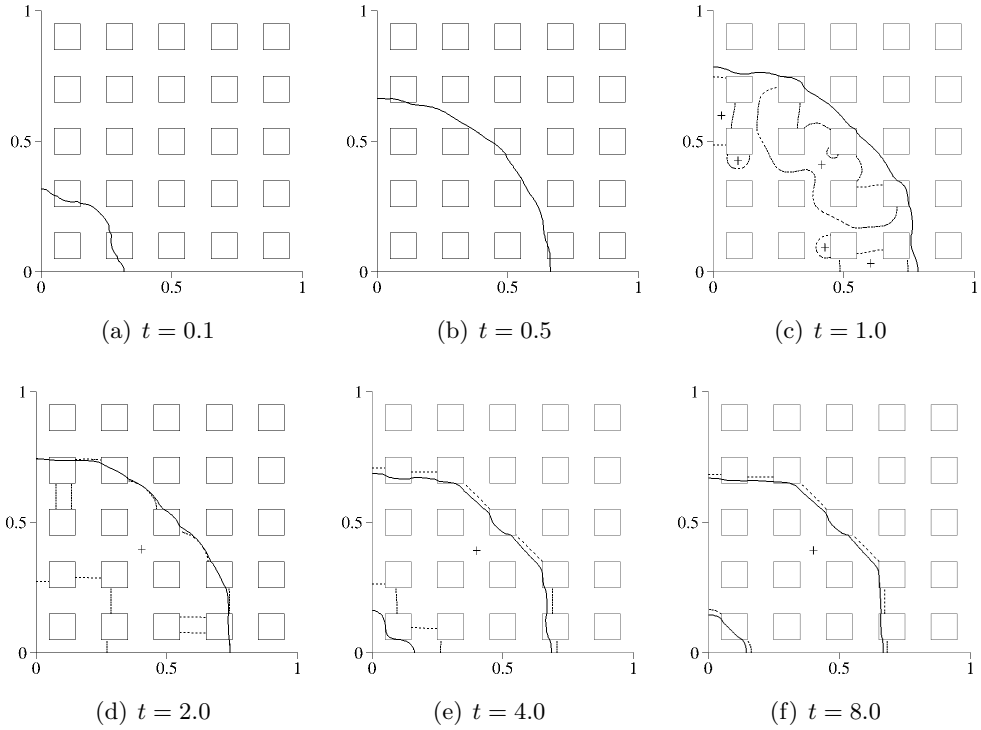
(a) $t = 0.1$ (b) $t = 0.5$ (c) $t = 1.0$

(d) $t = 2.0$ (e) $t = 4.0$ (f) $t = 8.0$

Figure 3.16 Droplet impact on a checker patterned solid surface. Snapshots at different dimensionless time. Plots illustrate the process how the bottom layer curves downwards the solid bottom and rejoins an integral hemispherical shaped inner interface with air trapped inside.

through the channels between the cube pillars and reform an integral layer. The rejoined sublayer moves towards the center and arches up in a hemispherical shape to release the surface energy. Finally we see an air entrapment form at the center of the droplet on the bottom. The upper interface performs similarly in the spreading and receding stages as the above cases.

Figure 3.17 gives the details of the wetting process of the droplet impacting on checker-patterned surface. As the grooved surface, the solid lines and dashed lines indicate the interface positions on the plane tangent to the top surfaces of the cube pillars and the bottom surface, respectively. The closed regions marked with the "+" indicate the wetting areas as well. Similar to the case of grooved pattern, the droplet does not wet the bottom surface during the fast spreading stage. The wetting spots begin to appear when spreading slows down. However, we notice here that the wetting spots show up earlier that they do on the grooved surface. Moreover, Figure 3.17 (c) reveals that the wetting spots appear away from the center, which is different from the case of the grooved surface where wetting spots appear first near the center. As seen in Figure 3.16, here we have a better view of how the wetting spots join together and



(a) $t = 0.1$ (b) $t = 0.5$ (c) $t = 1.0$

(d) $t = 2.0$ (e) $t = 4.0$ (f) $t = 8.0$

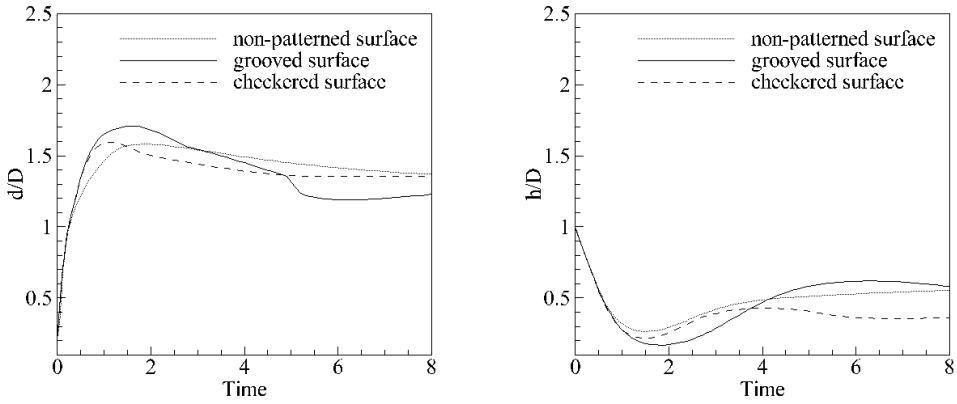
Figure 3.17 Formation of the wetting spots above and on the checkered surface during the impacting process. Solid line indicates the intersection between the droplet interface and the plane tangent to the top of the cube pillar. The dashed line indicates the intersection between the droplet interface and the bottom surfaces below the cube pillars. The closed regions marked with the "+" signs indicate the wetting areas on the bottom surface, while the areas without the "+" signs indicate air entrapment.

entrap an air bubble at the center. Eventually, the droplet and the entrapped air bubble both tend to form a hemispherical shape. But cube pillars drag the interface to form a straight shape connecting the pillar vertices diagonally, instead of a smooth surface perpendicularly contacting with the pillar side walls.

Figure 3.18 quantitatively compares the evolution of the wetting diameter and height for the droplet impact on the two patterned surfaces. Figure 3.18(a) plots the evolution of the wetting diameter. The wetting diameter is measured on the plane tangent to the top surfaces of the grooves and the cube pillars (as where the solid curves locate in Figure 3.15 and Figure 3.17). The wetting diameter for the grooved surface is defined as the square root of the multiplication of the two main diameters of the elliptic wetting spot. The solid line is for the grooved surface

and the dashed line is for the checkered surface. To compare with the case of a non-patterned flat surface, the wetting diameter evolution of droplet impact on non-patterned surface is plotted as the dotted line in the plot. From the figure we see that during the initial spreading stage, the three curves almost overlap with each other, indicating that the patterned surface has less effect on the fast spreading process. As the spreading process slows down where the wetting diameter reaches the peak value, the three types of surfaces begin to show differences in the wetting diameter sizes. We see that the checkered pattern gives a wetting diameter larger than the flat surface does. The grooved pattern has almost the same wetting size as the flat surface, but the wetting diameter reaches its peak value later than on the flat surface. During the receding stage, we see that both the grooved surface and the checkered surface give relatively smaller wetting diameters than the flat surface, which partly results from the volume of the droplet sinking below the plane into the grooves or among the cube pillars. It is interesting to notice that the wetting diameter on the checkered surface gradually approaches to the size of wetting spot on flat surface, but the one on the grooved surface does not. This indicates that the wetting spot on the grooves surface has smaller area than the one on the flat or checkered surface.

Evolution of the height of the droplet is given in Figure 3.18(b). The height of the droplet is measure as the distance from the center on the top interface to the plane tangent to the top surfaces of the grooves and the cube pillars (as where in solid curves locate in Figure 3.15 and Figure 3.17). Similar to the behavior of the wetting diameter, the height of droplet shows no difference among the three types of surfaces during the fast spreading stage. During the receding stage, the droplet height does not differ much from the one on the flat surface at beginning, but goes lower at a much later time, due to the sinking of the droplet below the upper plane. However, though the grooved surface reduces the wetting diameter, it does not change the droplet height much compared to the flat surface. From the two figures, we see that the grooved surface considered in this study has a more significant effect on the wetting spot size, but has much less effect on the droplet height. However, the checkered surface considered in this study has almost no effect on the wetting spot size, but has more effect on the droplet height.



(a) (b)

Figure 3.18 Evolution of the wetting diameter and height of a 3D droplet impacting on patterned surfaces. (a) wetting diameter; (b) droplet height.

3.6 Scalability test on the numerical framework

A scalability test is performed for our numerical framework to test its parallelization capability on 3D problems. We use the 3D droplet impact on solid surface simulation as the test problem, and solve for 10 time steps. Then the run time T_p for each iteration at each time step (during which the Cahn-Hilliard and Navier-Stokes equations are both solved for just one step) is estimated by dividing the total run time for the 10 steps with the total number of iterations, where p is the number of CPUs employed. The relative speedup S_p reflects how much faster a parallel algorithm can be by employing more processors, which is defined as the ratio between the execution time of the sequential algorithm and the execution time of the parallel algorithm with p processors, $S_p = T_1/T_p$. Because of the memory restriction of the machines where we perform the simulations, it is difficult to run such a large 3D problem on only one processor. Thus we estimate the relative speedup S_p using the minimum possible number of processors p_{min} and its related run time $T_{p_{min}}$ for the simulation on a given machine, as $S_p = p_{min}T_{p_{min}}/T_p$.

Figure 3.19(a) represents the execution time and the relative speedup obtained from the scalability tests for a small size problem. This test consists of 160^3 elements, which gives 12.5M degrees of freedom (DOF). The DOF is estimated by the maximum number of unknown

variables involved in a single linear system, e.g. the number of total nodes as 161^3 by the number of velocity components 3 for 3D simulation. The test is performed on both TACC Stampede and CyEnce at Iowa State University. From the plot, we see that our algorithm has almost linear speedup for a relatively smaller number of processors compared with the ideal linear speedup shown by the dotted line. However, the speedup slows down as the number of processors increases to larger values. This is not surprising since the total DOF is not changed but more time is spent on communication among processors when more processors are involved in the simulation. Moreover, the speedup is also restricted by the design of the cluster, where a machine with better communication strategy and advanced hardware/software can improve the speedup. We see in the plot that Stampede has better speedup than CyEnce.

Figure 3.19(b) shows the scalability test for a large size problem, where there are 252^3 elements giving 48.6M DOF. Because of the restriction on maximum number of processors a user can use for a single run on CyEnce, the large size test is performed only on TACC Stampede. From this plot, we see that the speedup significantly drops when a very large number of processors are employed for large size problems. This indicates that the communication among processors become dominant and becomes the bottleneck of run time when employing more processors.

Another way to perform the scalability test is to increase the number of processors while maintaining the DOF on each processor, known as the weak scalability test. Figure 3.19(c) shows the weak scalability test performed on Stampede. In this test, three sizes of grids are considered. The smallest size grid has 160^3 elements (12.5M DOF). The medium size grid has 200^3 elements (24.4M DOF). The largest size grid has 252^3 elements (48.6M DOF). The Cahn number Cn is adjusted for each test to ensure that there are at least four elements through the interface. From the plot we see that our framework scales when the DOF per CPU varies between 10K and 50K. This guarantees the that our framework is appropriate for very large scale 3D simulations.

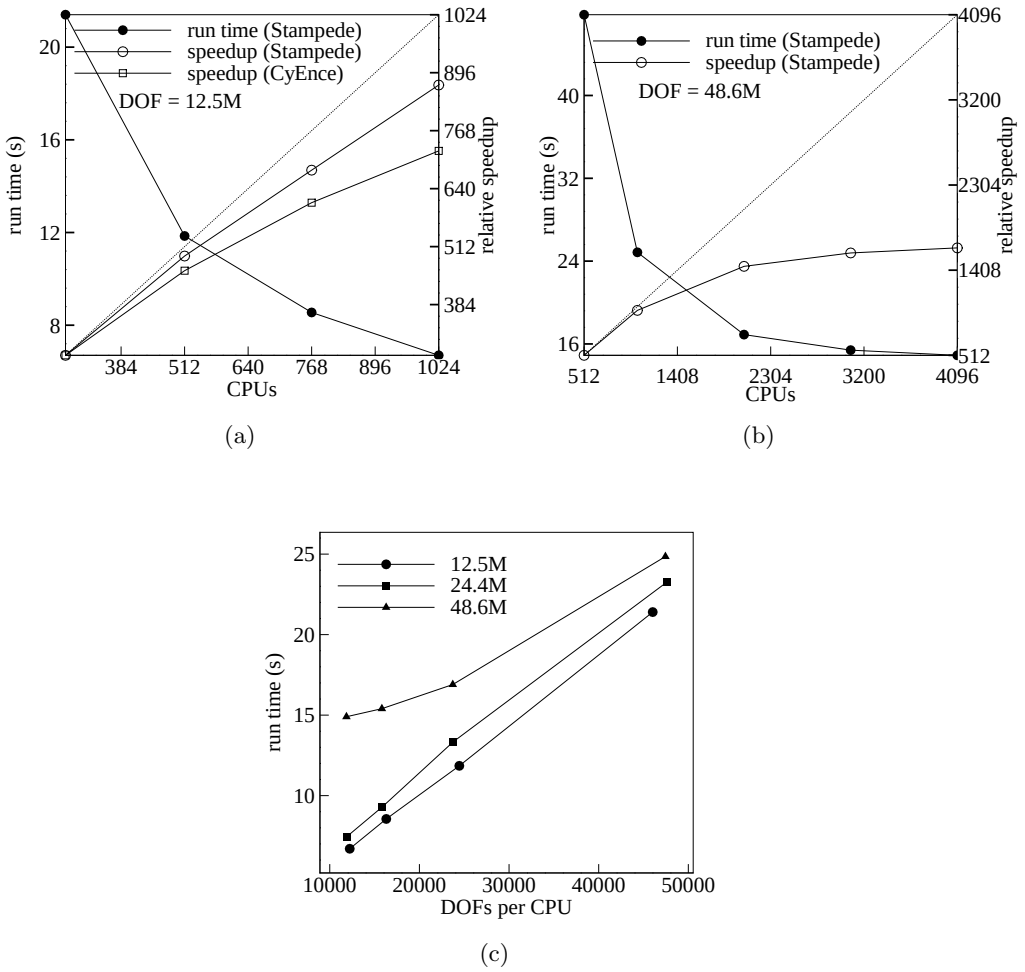


Figure 3.19 (a) Results obtained from the scalability tests for a small size problem (12.5M DOF) performed on TACC Stampede and CyEnce at Iowa State University, where T_p measures the execution time of the simulation with p processors, and S_p represents the relative speedup of run time with p processors compared with the serial execution; (b) Results obtained from the scalability test for a large size problem (48.6M DOF) performed on TACC Stampede; (c) Results for the weak scalability test on Stampede, where three sizes of meshes are considered here with DOFs of 12.5M, 24.4M, and 48.6M for each of them. The diagonal lines in (a) and (b) indicate the ideal speedup.

3.7 Conclusion

In this paper, we introduced a finite element scheme for solving two-phase flows by utilizing the diffuse interface model. To stabilize the numerical solution for velocity and pressure, the SUPG stabilization term is added to the traditional Petrov-Galerkin scheme. The four-step fractional temporal discretization is employed to decoupled the pressure and momentum equations. Coupling terms between the Navier-Stokes equation and Cahn-Hilliard equation are carefully investigated to guarantee that our numerical method preserves physical properties as conservation of mass and accurate approximation of surface tension, especially when a large density ratio is involved in the simulation. Coupling between the two equations is treated iteratively, which has been verified as an efficient and easy-to-implement method without reducing the accuracy. Within each iteration, the nonlinear Cahn-Hilliard and Navier-Stokes equations are both linearized by a Newton-Raphson scheme and solved by the Petsc SNES solver. Validations are then performed to test the accuracy of the current numerical scheme by comparing the numerical results with theoretical solutions. A useful result provided by the validations is that we verified that the minimum requirement of element density though the interface should be satisfied for the coupled system to guarantee the physical correctness of numerical solution. Various numerical examples are studied to test the performance of our method. Some of these illustrations will be further investigated in our future research based on this framework. Finally a series of scalability tests are performed on our numerical framework. From the tests we see that this framework is very scalable even for very large 3D simulations on different machines, which demonstrates the efficiency of our methods.

CHAPTER 4. FAULT TOLERANT ADAPTIVE SPARSE GRID COLLOCATION OVER HETEROGENEOUS COMPUTING ARCHITECTURES

Modified from a paper to be submitted to *Computer Methods in Applied Mechanics and Engineering*

Yu Xie, Jaroslaw Zola, and Baskar Ganapathysubramanian

4.1 Abstract

Sparse grid collocation and its adaptive variants have emerged as one of the versatile techniques to seamlessly augment legacy deterministic software to incorporate uncertainty quantification. Adaptive Sparse grid collocation (ASGC) frameworks invoke multiple calls to the deterministic solver to construct a stochastic representation of the output quantities of interest. Current implementations efficiently manage data and deploy deterministic simulation. However, for stochastic multi-scale and multi-physics problems with complicated random inputs, the number of deterministic equations to be solved is considerable. Large heterogeneous computing clusters become the best choice as the computational tools for solving such problems. However, a serious problem is that the stability of these machines cannot be guaranteed over the long time intervals involved, thus making fault tolerance a critical issue when solving high dimensional complex problems.

We develop a fault tolerant adaptive sparse grid collocation framework, which has both high immunity to hardware exceptions and high flexibility to computational scale-up. This fault-tolerant framework follows basic rules for fault-tolerant systems in terms of axiomatic approaches. This framework deploys multi-thread communicating tools to optimally parallelize the executions of the deterministic solvers. This fault-tolerant framework allows ASGC method

to deal with high stochastic dimensions and to use large scale deterministic solvers, which are impossible for previous framework because of lack of ability to operate large dataset. Performances of this fault tolerant framework are tested by showcasing illustrations in solving very high dimensional stochastic problems over thousands of processors.

4.2 Introduction

Various methods for solving stochastic differential equations, and simulating the often complex distribution of such solutions, have been developed during the past decades. There are mainly two categories of such methods: statistical and non-statistical methods. The most famous representative of the statistical approaches is the Monte-Carlo (MC) method. It is widely used for its ease in implementation. Of the non-statistical approaches, one important method is known as generalized polynomial chaos (gPC) expansion [Xiu and Karniadakis (2002)], which represents the unknown process with a set of complete orthogonal polynomials in random space. This method belongs to a family of methods known as spectral stochastic finite element methods (SSFEM) [Ghanem and Spanos (1991)]. Another widely used technique in the non-statistical family is the sparse grid collocation (SGC), with its adaptive variant as the adaptive sparse grid collocation (ASGC) method [Ganapathysubramanian and Zabaras (2007); Zabaras and Ganapathysubramanian (2008); Ganapathysubramanian and Zabaras (2008))].

The SCG method uses the Smolyak algorithm [Smolyak (1963)] and tensor products to selectively generate grid points in the sampling space. ASGC deduces the number of sampling points by defining a criterion for accepting new grid points for next interpolating level. This method is viewed as the easiest method to implement and utilize in solving stochastic equations for the following reasons. Firstly, it relies only on multiple calls to deterministic solvers, without effort to develop new stochastic routines in the code. Secondly, it allows the incorporation of multiple sources of uncertainty in a very straightforward manner. Thirdly, it is a highly scalable method because of the independence calls of the deterministic solvers, which can be completely parallelized.

Implementation of the ASGC method is also one important field of study. The conventional way of implementation is to declare the scientific solver as a subroutine of the ASGC collocation

point allocator [Ma and Zabaras (2009)], which we refer as the "one-program" design in this study. Although the ASGC method does not require users to derive additional equations for the governing equations, users still have to hardcode their scientific solvers into the conventional implementation framework and compile before performing the simulations. The conventional design supports parallel execution, for instance on a super computer, where collocation points are distributed on multiple processors. However, the scientific solver on each collocation point is restricted to using only a single processor due to the design of the conventional framework. This significantly limits the capability of such a framework for solving large scale problems which require multiple processors to execute.

When the stochastic dimension becomes very high, the number of collocation points can be considerable. Consequently, the "one-program design has to use more and more processors at the same time to preserve efficiency when the stochastic dimension increases. These many processors usually distribute on multiple computational nodes (machines which form the cluster), where each node lies a certain possibility of failure. So when a large number of nodes get involved to perform a collective task, the chance of the occurrence of machine failure increases during execution of the program. Thus it is necessary to think of a new design in implementation which is well scalable to the size and dimension of the stochastic problem.

The ASGC method solves stochastic equations in a hierarchical way, which requires the completion of all evaluation jobs at one level to start the next level. As mentioned before, the conventional "one-program" framework is distributed on multiple processors and is implemented by the message passing interface (MPI) for inter-processor communication. One consequence of the MPI based "one-program" architecture is that any failure, either software or hardware caused, on any single processor will lead to the abnormal termination of the whole solving process. This brings the necessity of designing a fault-tolerant implementation of the ASGC method, which is capable to restart only the failed sub-processes to ensure the completion of the current level.

There have been some discussions on designing a fault-tolerant system in previous literature. Schlichting and Schneider (1983) discussed the key features to design a fault-tolerant systems and provided a minimum set of criterion which a fault-tolerant system should satisfy. These

rules in their paper were written in an axiomatic form with the language invented by Hoare (1969). The key ideas for designing a fault-tolerant system include keeping some important contents on stable storage, and equipping the programs with some recovery protocols to recover failed processes. Following but not limited to these rules, we will introduce a fault-tolerant implementation of the ASGC method in this study, where we use some monitoring units to periodically access the statuses of active jobs and to restart failed ones. This design breaks the "one-program" design into independent processes and uses an inter-process method for communication between the scientific solver and a managing controller.

Another issue caused by the increasing of stochastic dimension when implementing the ASGC method is the huge data generated at each level. In the conventional "one-program" design, the implementation utilizes the random-access memory (RAM) as the storage medium. However, as the stochastic dimension and the scale of the problem increase, the data size quickly exceeds the capacities of RAM on most computers or clusters. A remedy of this issue is to store data on the hard drive. The ASGC algorithm has a hierarchical structure, where the execution of current level is based on the results from all previous levels. Therefore a very typical data operation is searching and retrieving from the data set stored for previous levels. Though the hard disk has less limit on its storage capacity, the reading and writing speed is much slower compared with RAM. However, a better structured and designed database can significantly improves the performance of data operation. So in our design, we transform the data storage from RAM to hard drive, and incorporate the hierarchical data format version 5 (HDF5, The HDF Group (2014)) database library, which organizes data in a hierarchical way and provides advanced data manipulation algorithms.

Following these requirements and logics, we designed a well scalable framework which separates the deterministic solver away from the sparse grid allocator. Thus both the allocator and the deterministic solver work independently as individual jobs on cluster, without letting one program employing many processors. We also design this framework as fault-tolerant to implement large scale and high dimension stochastic problems satisfying the fault-tolerant systems design criterion mentioned earlier. We also developed a series of implementation techniques and codes to realize such a framework, using some file system structures and operations tech-

niques in Linux. Real simulations showed that this framework is suitable for high dimension stochastic simulations.

The detailed features of this framework are introduced in this paper. We will define the stochastic problems in section 4.3, then briefly introduce the theory of adaptive sparse grid collocation in section 4.4. After this, the fault-tolerant system design criterion will be briefly reviewed in section 4.5. In section 4.6, our fault-tolerant sparse grid collocation framework will be developed step by step following the criterion towards fault-tolerant systems design. In section 4.7, implementation strategies and detailed techniques will be developed to realize the framework developed in section 4.6. Later in section 4.8, various numerical examples, especially problems with high stochastic dimensions, will be used to test the performance of the fault-tolerant framework. At last, the main contents of this paper will be summarized and potential future work will be discussed in the conclusion section, section 4.9.

4.3 Problem definition

4.3.1 Governing stochastic equations

In this section we describe some definitions and mathematical preliminaries used in stochastic modeling following the notions introduced by Ganapathysubramanian and Zabaras (2007). Define a complete probability space as $(\Omega, \mathcal{F}, \mathcal{P})$, where Ω is the set of outcomes, $\mathcal{F}$ is a σ-algebra of Ω, and $\mathcal{P} : \mathcal{F} \to [0, 1]$ defines a probability measure. Denote the spacial domain of interest as D, where $D \in \mathbb{R}^d$, is a d dimensional space, e.g. $d = 1$, 2 or 3. Denote the boundary of the physical domain as ∂D. The variable of interest is a stochastic function, denoted as $u : D \times \Omega \to \mathbb{R}$. By applying Karhunen-Loève expansion (KLE), the random field can be approximated by a finite number of random variables, ξ_i. Then the governing equation of u can be written as:

$$\mathcal{L}(u; \boldsymbol{x}, \xi_i) = f(\boldsymbol{x}, \xi_i), \quad \forall \boldsymbol{x} \in D, \xi_i \in \boldsymbol{\Gamma}, \tag{4.1}$$

with the boundary condition:

$$\mathcal{B}(u; \boldsymbol{x}, \xi_i) = g(\boldsymbol{x}, \xi_i), \quad \forall \boldsymbol{x} \in \partial D, \xi_i \in \boldsymbol{\Gamma}, \tag{4.2}$$

where N is the stochastic dimension and $\mathbf{\Gamma} = [0,1]^N$. The operator $\mathcal{L}$ is a differential operator, and $f(\cdot) : D \times \Omega \to \mathbb{R}$ is the source term, which is a known stochastic function. The boundary operator $\mathcal{B}$ and the boundary excitation term $g(\cdot)$ define the boundary condition of u on ∂D. So the purpose is seeking an approximation of u satisfying equation (4.1) and its boundary condition (4.2).

4.4 Adaptive sparse grid collocation method

The SGC method represents the unknown stochastic process as a polynomial approximation. This method interpolates the stochastic process by generating collocation points in the random space on which deterministic differential equation is solved. An intuitive but inefficient way of constructing the grid is using the full-tensor product of 1D interpolation. The shortcoming of the method is known as the "curse-of-dimensionality" [Clarkson (1994)]: utilizing k points per dimension in a N-dimensional space results in $O(k^N)$ points. This magnitude of number of points quickly explodes as dimension increases. In contrast, SGC employs a Smolyak's algorithm [Smolyak (1963)] based method to selectively construct the grid which can significantly reduced the number of points without loss of accuracy. Compared the $O(k^N)$ complexity of construction of grid, SGC use $O(klog(k)^{N-1})$ points to construct the same accurate interpolation as proved in Bungartz and Griebel (2004). Because of this advantage, SGC and its variants become more and more popular in uncertainty quantification.

The key feature of SGC is the hierarchical structure of interpolation, for which the interpolation propagates level by level. Consider a smooth function $f : [0,1]^N \to \mathbb{R}$. By using the notion of tensor product, the interpolation formula of f is:

$$(\mathcal{U}^{i^1} \otimes \cdots \otimes \mathcal{U}^{i^N})(f) = \sum_{j_1=1}^{m_1} \cdots \sum_{j_N=1}^{m_N} f(Y_{j_1}^{i^1}, \cdots, Y_{j_N}^{i^N}) \cdot (a_{j_1}^{i^1} \otimes \cdots \otimes a_{j_N}^{i^N}), \tag{4.3}$$

where $i_k \in \mathbb{N}$ is the level of interpolation on the k-th dimension, $\mathcal{U}(\cdot)^{i_k}$ denotes the interpolation of f on the k-th dimension at level i_k, $a_{j_k}^{i_k}$ are the nodal basis functions on the k-th dimension at level i_k, m_k are the number of elements on the k-th dimension. For all dimensions, at the initial level, $i_k = 0$, $\forall k = 1, 2, \ldots, N$, $\mathcal{U}^0 = 0$. If the interpolation of all dimensions are i_1, i_2,

$\ldots, i_N$, then denote the set of support nodes at this stage as $(X^{i_1} \times X^{i_2} \times \cdots \times X^{i_N})$, where $X^{i_k} = \{Y_{j_k}^{i_k} \mid Y_{j_k}^{i_k} \in [0,1], j_k = 1, 2, \ldots, m_k\}$.

By using the notion of equation (4.3), the conventional Smolyak algorithm for constructing the sparse grid interpolation $\mathcal{A}_{q,N}$ of f can be written as [Ganapathysubramanian and Zabaras (2007)]:

$$\mathcal{A}_{q,N}(f) = \sum_{q-N+1 \leqslant |\boldsymbol{i}| \leqslant q} (-1)^{q-|\boldsymbol{i}|} \cdot \binom{N-1}{q-|\boldsymbol{i}|} \cdot (\mathcal{U}^{i^1} \otimes \cdots \otimes \mathcal{U}^{i^N})(f), \tag{4.4}$$

where $q \geqslant N$, $\boldsymbol{i} = (i_1, i_2, \ldots, i_n)$ is the muli-index of levels at all dimensions with $|\boldsymbol{i}| = i_1 + i_2 + \ldots + i_n$ as the normal definition of 1-norm of $|\boldsymbol{i}|$. Define the increment of interpolation between two levels for the k-th dimension as $\Delta^{i_k} = \mathcal{U}^{i_k} - \mathcal{U}^{i_{k-1}}$. Then with the incremental notion, equation (4.4) can be written as:

$$\mathcal{A}_{q,N}(f) = \sum_{|\boldsymbol{i}| \leqslant q} (\Delta^{i_1} \otimes \cdots \Delta^{i_N})(f). \tag{4.5}$$

Recursively, the following hierarchical interpolation structure of $\mathcal{A}_{q,N}$ holds:

$$\mathcal{A}_{q,N}(f) = \mathcal{A}_{q-1,N}(f) + \Delta\mathcal{A}_{q,N}(f), \tag{4.6}$$

$$\Delta\mathcal{A}_{q,N}(f) \equiv \sum_{|\boldsymbol{i}| = q} (\Delta^{i_1} \otimes \cdots \Delta^{i_N})(f), \tag{4.7}$$

where $\mathcal{A}_{N-1,N}(f) = 0$. To compute $\mathcal{A}_{q,N}(f)$, one needs to evaluate the function values at the sparse grid points given by $\mathcal{H}_{q,N} = \bigcup_{q-N+1 \leqslant |\boldsymbol{i}| \leqslant q} (X^{i_1} \times X^{i_2} \times \cdots \times X^{i_N})$.

Equation (4.6) gives the way that ASGC improves the accuracy of interpolation by generating more sampling points. Thus the selection of the sparse grid points set X^i is made in a nested fashion, $X^i \subset X^{i+1}$. So when interpolation jumps from level $i-1$ to level i, one only needs to evaluate function values at new generated grid points set ΔX^i, $\Delta X^i \equiv X^i \setminus X^{i-1}$. Furthermore, the increment of the whole sparse grid points set from order $q-1$ to order q can be denoted as $\Delta\mathcal{H}_{q,N} = \bigcup_{|\boldsymbol{i}|=q} (\Delta X^{i_1} \times \Delta X^{i_2} \times \cdots \times \Delta X^{i_N})$.

According to the hierarchical nature of Smolyak algorithm, SGC can be improved with adaptivity which further reduced the number of sparse grid points as mathematically described in Griebel (1998). The basic idea towards ASGC method is to use the hierarchical surplus as

an error indicator and only refine the grid points whose hierarchical surplus is larger than a pre-defined threshold. From equation (4.6), we obtain

$$\Delta\mathcal{A}_{q,N}(f) = \sum_{\substack{|\boldsymbol{i}|=q,\\ \boldsymbol{j}\in B_{\boldsymbol{i}}}} (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_N}^{i_N}) \cdot \underbrace{(f - \mathcal{A}_{q-1,N}(f))(Y_{j_1}^{i_1}, \cdots, Y_{j_N}^{i_N})}_{w_{\boldsymbol{j}}^{\boldsymbol{i}}}, \tag{4.8}$$

where $B_{\boldsymbol{i}}$ is the multi-index set, $B_{\boldsymbol{i}} \equiv \{\boldsymbol{j} \in \mathbb{N}^N \mid Y_{j_k}^{i_k} \in \Delta X^{i_k}, \forall j_k = 1, \ldots, m_{i_k} - m_{i_{k-1}}, k = 1, \ldots, N\}$, and $w_{\boldsymbol{j}}^{\boldsymbol{i}}$ is the hierarchical surplus, which is the difference between the value of f and the interpolation value from the previous levels at the newly generated grid point. For continuous functions, the hierarchical surpluses tend to zero as the interpolation level tends to infinity. Furthermore, for non-smooth functions, details about the singularities are indicated by the magnitude of the hierarchical surplus. By ASGC, at each level, the surplus $w_{\boldsymbol{j}}^{\boldsymbol{i}}$ is computed at each new generated point in $\Delta\mathcal{H}_{q,N}$. If $\|w_{\boldsymbol{j}}^{\boldsymbol{i}}\| < \varepsilon$, where ε is the prescribed threshold, then this point will be accepted at current level, otherwise it will be removed from $\Delta\mathcal{H}_{q,N}$. By applying this adaptive strategy, the number of sparse grid points can be significantly deduced.

4.5 Basic requirements of designing fault-tolerant systems

4.5.1 Definition of campaign

From the definitions of ASGC in section 4.4, it can be concluded that the structure of the sparse grid propagates hierarchically level by level. At a certain level, new sample grid points are generated on which values of the function should be evaluated separately. Should the evaluation procedure at current level be completely done, necessity of creating more sample points for next level can be decided. This procedure continues until no sample points being created or the interpolation level reaches a bounded depth. We can describe this process more rigorously using the following notions. The whole modelling procedure consists of a number of *campaigns* $J = \{\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_N\}$, where N is a positive integer. Each campaign $\mathcal{J}_i$ is composed of m_i independent jobs $J_{i,j}$, $0 \leqslant j \leqslant m_i$. These m_i jobs can be executed in parallel.

Step 1. The modelling begins with a initializing campaign, $\mathcal{J}_0$, which contains only one job J_0, $\mathcal{J}_0 = \{J_0\}$;

Step 2. For the campaign $\mathcal{J}_i$, job $J_{i,j}$, $i \geqslant 0$, $j \leqslant m_i$, is a deterministic problem at a unique grid point $\boldsymbol{Y}_{i,j}$. These jobs run in parallel and the output of job $J_{i,j}$ is the solution to the deterministic solver at point $\boldsymbol{Y}_{i,j}$;

Step 3. Only when all the jobs in campaign $\mathcal{J}_i$ are complete, a sparse grid allocator [1] analyzes the results of campaign $\mathcal{J}_i$, then generates a new sparse grid points set $\Delta\mathcal{H}_{i+1} = \{\boldsymbol{Y}_{i+1,1}, \boldsymbol{Y}_{i+1,2}, \ldots, \boldsymbol{Y}_{i+1,m_{i+1}}\}$. The corresponding jobs $J_{i+1,1}, J_{i+1,2}, \ldots J_{i+1,m_{i+1}}$ form the next campaign $\mathcal{J}_{i+1}$ [2];

Step 4. Repeat the above steps until $\Delta\mathcal{H}_{i+1} = \emptyset$, or $i > N$.

4.5.2 A brief introduction to the axiomatic approach

Consequently, there comes the question how to design a framework that links the sparse grid allocator and jobs in the campaigns together. One can never prevent the occurrence of exceptions on a computational machine, but it is able to recover the failed program at the point where it was interrupted. So it is necessary to clarify the conditions to characterize the beginning and ending of a program. Most of the executions of programs can be described with the syntactic notion called a "triple", $\{P\}S\{Q\}$, as the Floyd-Hoare axiomatic approach Hoare (1969). Here S is a *programming statement*, or part of a program. The variables P and Q are assertions, which consist of programs and logical values, and usually be judged for true or false, say, by a Boolean value. P is the assertion containing the variables being valued before the initialization of S. The assertion P is called the *precondition* of S, denoted as $P = pre(S)$. When S is terminated, some results can be obtained base on the termination of S. These variables are contained in the assertion Q, which is called the *postcondition* of S, denoted as $Q = post(S)$. The notion of triple provides a rigorous and convenient way to prove the correctness of a program in a literature way.

[1] A sparse grid allocator can only generates points in the normalized random space, $\mathcal{U} = [0, 1]^d$, where d is the stochastic dimension. This is sufficient since that it is possible to map $\mathcal{U}$ to any random space using topology methods.

[2] This is called the "barrier" property of hierarchical interpolation. Campaign $\mathcal{J}_{n+1}$ cannot be started until its previous campaign $\mathcal{J}_n$ is finished.

4.5.3 Recovery protocol

In classical fault-tolerant system, an action statement is associated with a *recovery protocol*. When a failed job is restarted, the internal state and the contents of its volatile storage cannot be automatically restored to the state of the moment when failure occurs. So a program should be associated with a routine which completes the state transformation and restores storage to a safe place when failure occurs. Such a routine is called a recovery protocol [Schlichting and Schneider (1983)]. Obviously, a recovery protocol should be stored and run on a stable storage, and only uses contents stored on this stable storage. A recovery protocol should always be "available" to make correct execution without being interrupted. Notice that this assumption of existence of stable storage is reasonable for most of the computer clusters. There is usually a host node on a computer cluster as users home directories. Such a host node is protected by strict rules. No parallel jobs are allowed to run on the host node, so hardware exceptions rarely happen on the host node. The storage related to the host node is also well maintained, which can be viewed as stable.

Compared with classic fault-tolerant systems, fewer requirements of our new fault-tolerant framework bring us possibility to simplify the functions of the recovery protocol. The main function of our recovery routine is to guarantee the completion of executions of all jobs in a campaign. It is not necessary to let this recovery routine work for the whole campaign, since that only a very few number of jobs of a campaign may get failed. So an efficient choice is to associate the recovery routine with each job. Thus the requirement of this recovery is clear: monitoring the status of a single job, whenever failure is detected, immediately restart the failed job (might be restarted on the same nodes if previous failure is caused by software issues or if a hardware exception has been fixed at restarting moment). What is different with classical definition of recovery protocol is that in the current framework the recovery operations are simply substituted by restarting operations, to avoid the complex data transformations between unstable storage and stable one. For the consistency of the framework, we still use the term recoverable protocol for the simplified version used in the current framework.

4.5.4 Basic rules of a fault-tolerant system

Schlichting and Schneider (1983) also derived a series of axiom logics based criterions which can help programmers to design fault-tolerant programs and can be used to prove the correctness of the fault-tolerant actions. Instead of going into details of the theorems, here we just briefly described the important rules towards designing the fault-tolerant ASGC solver without giving demonstrations. Interested readers may refer to the original paper. A fault-tolerant action, denoted as FTA is a sequence of statements A associated with a recovery protocol R. When the action A is failed, the recovery protocol R will recover (restart) this failed job. The triple of FTA is written as $\{P\}\text{FTA}\{Q\}$, where P and Q are the precondition and postcondition, respectively. It is able to separate this triple into two triples, $\{P'\}A\{Q'\}$ and $\{P''\}R\{Q''\}$, where P' and Q' are precondition and postcondition for program A, P'' and Q'' are precondition and postcondition for the recovery protocol R. Correct execution of A can directly lead to Q, when the recovery protocol is not activated. The recovery protocol can lead to the final postcondition Q when it is activated and invoked. To ensure that the recovery protocol can continue the execution of the program, all program variables named in P'' must be in stable storage, which cannot be interrupted by any failure of unstable processors. Besides, as mentioned before, the recovery protocol itself is also running on the stable storage with stable processors. Certain types of key conditions in $\{P'\}A\{Q'\}$ and $\{P''\}R\{Q''\}$ need also lead to P'' for recovery purpose. Finally, variables stored in volatile storage may not be named in assertions appearing in programs executing on other processors. It is seen that his rule is automatically satisfied by the new framework, since that all jobs are independent and variables named in assertions are kept on stable storage.

4.6 Design a fault-tolerant implementation framework for sparse grid collocation method

4.6.1 Conventional "static" linking strategy

The conventional framework combines the function evaluating operator with the allocator. This strategy can be easily realized with most of the programming languages using Object

Oriented Programming (OOP) architecture, like C++. For example, using C++, one can define the allocator as a class, and include one function evaluation subroutine in such class as a member function, which returns function value at a given sparse grid point. This linking strategy between allocator and campaigns is named as "static" linking in this paper. At level i, when the allocator has created all sparse grid points for the current level, campaign $\mathcal{J}_i$ is claimed immediately by all these points. Then within the same program, the jobs are started on the same processors where the allocator was running. Typically with the "static" linking, each job takes only one processor. One processor may need to run more than one job, sequentially. Each job performs as an evaluation program A for a single sparse grid point. The termination of program A generates a postcondition Q' as the realization of the stochastic problem at this grid point. Notice that no variables in Q' are stored on the stable storage (they are stored in the RAM on volatile computational nodes), once exceptions fail program A, information of A will be immediately lost. Furthermore, Q' leads the program to nowhere since A is not equipped with a recovery protocol. Therefore the "static" linking framework is not fault-tolerant. When the stochastic dimension is low where only a small number of processors are employed, the static linking strategy shows its advantages as fast executing. Besides, there needs no data transmission between allocator and jobs in the campaign. However, for high stochastic dimensional simulations which involves large amount of computational resources working simultaneously and huge dataset operations, this old framework cannot guarantee the successful execution of ASGC method.

More hidden problems of the "static" linking appear as the stochastic dimension become high and scale of simulations increase. Firstly, "static" linking has low flexibility to the implementation of the function evaluation operator. Since the function evaluation operator is defined as a subroutine in the allocator class, every time users want to change this operator they have to modify the source files where the function is defined. For users who are not familiar with the source code, this modification process brings too much inconvenience. Secondly, "static" linking restricts the scalability to the function evaluation operator. Recall that using the "static" linking strategy, each processor only handles one job. Even it is possible to break this restriction to let a job run in parallel, this requires more coding work which makes the pro-

gram more complicated and less readable. The last but not the least, notice that a successful run of "static" linking framework requires that all the generated points are correctly evaluated. According to the scalability restriction of "static" linking, number of processors employed for computing increases fast as stochastic dimension increases. For a program running on a large number of processors, it is impossible to guarantee all the processors run normally. This is a fatal danger to sparse grid collocation according to the barrier property between two consecutive campaigns: campaign $\mathcal{J}_k$ may never proceed to campaign $\mathcal{J}_{k+1}$ if not all the processors for campaign $\mathcal{J}_k$ work correctly.

4.6.2 Desired properties of the fault-tolerant framework

The above problems require us to seek for a new linking strategy for the sparse grid collocation framework. There are mainly two properties that the new linking strategy to satisfy according to previous analysis. First, to cope with the problem of fast increasing number of processors and scalability of jobs, it is necessary to free the function evaluation subroutine out from the allocator as an individual program. Thus the number of processors employed can be kept at a limited and controlled level during execution. Moreover, it is possible to assign more processors to run a single job without restriction due to design issue. The new problem involved is that there needs one more process to communicate the allocator and the jobs. Second, although the number of collective processors is controlled, it is still not wise to rely on the luck of exception not occurring. The risk of processor failure always exists. The barrier property of between campaigns requires the framework to guarantee the successful executions of all jobs at each level. This implies that the new framework should be "fault-tolerant", or say, whenever processor failure occurs, the new framework should be able to protect all the running jobs on failed processors to recover to normal running status on other healthful resources.

A recovery protocol is necessary for a fault-tolerant system. The number of jobs in a campaign can be extremely large for high stochastic dimension. According to the previous analysis, these jobs will be run individually on a computer cluster. A computing cluster is a group of computational nodes connected to each other through a network. A parallel job can be assigned to one or more computational nodes by a scheduler, like PBS, or SGE scheduler. It is

very possible that a job terminates abnormally due to various reasons, such as memory overflow, or hardware exceptions. When failure occurs for a job, its related processes will be simply halted, and the volatility connected storage on its running nodes will be irretrievably lost. So the computational nodes and their connected storage can be viewed as unstable. Fortunately, the status of all running jobs can be monitored by the scheduler. The scheduler can tell whether their jobs are queuing, running, or being terminated, etc. Whenever failures of jobs have been detected by querying jobs status, the fault-tolerant framework should immediately recover the failed jobs. Thus each job should be equipped with a monitoring process to protect its execution on computational nodes.

4.6.3 Fault-tolerant framework for ASGC

We can modify the "static" linking logic to give a restartable fault-tolerant action. According to the criterion, key variables should be stored on stable storage. At level i when the evaluation procedure of job $J_{i,j}$ is done, we store the evaluated value in a result file on the stable storage. This result file will be loaded by the allocator to generate the new grid points set $\Delta\mathcal{H}_{i+1,N}$ for next level. When $\Delta\mathcal{H}_{i+1,N}$ is ready, the coordinates of all the points in $\Delta\mathcal{H}_{i+1,N}$ will also be stored in another file, which will be used to define campaign $\mathcal{J}_{i+1}$ at level $i+1$. Each job $J_{i,j}$, $0 \leqslant j \leqslant m_i$, is an individual job running on one or more computational nodes. So we need a script file to submit job $J_{i,j}$ to the scheduler. Once a job is submitted, a unique job ID will be assigned to this job by the scheduler to check the job running status. Assume that it is always available to use this job ID to query the status of job $J_{i,j}$ from the scheduler. Most of the clusters use various symbols to indicate the job status which can be checked interactively by the users.

The strategy used here to form a fault-tolerant framework is that design a recoverable program R to execute an evaluating program E for each job on the cluster, and design another process M running on a stable machine to repeatedly call R simultaneously with different inputs. Since R only relates to an individual job, once the job submitting script file is ready, R should be able to start the evaluation job E on the cluster. Notice that the evaluating program E itself is not recoverable. It is R that provides the recovery protocol for each job. Program M

runs on the stable storage with stable processors. All the key conditions and variables related to the evaluating jobs are stored on the stable storage. Once a certain evaluation job E is failed, R can immediately detect the failed conditions and restart this job. The termination condition for M at each interpolation level is the set of all the termination conditions of all the running evaluation jobs. This means that when all the evaluation jobs are successfully finished, new interpolation level shall be started. Since program R plays crucial role in protecting a individual job running on the cluster, we call R a "monitoring unit", which also separates the deterministic solver apart from the allocator to let it run as individual job. Here we name this separated framework as "dynamic" linking strategy, to be distinguished with the "static" linking strategy. Here the program M is called the "manager process", which links the allocator and the deterministic solver in a dynamic way. There should be another important part in the "static" linking framework to serve for the communication between the manager process M and the monitoring process R. Details about the design and the implementation of such "dynamic" framework will be introduced in section 4.7 as following.

4.7 Fault-tolerant framework implementation methods

4.7.1 Architecture of the "dynamic" linking ASGC framework

Figure 4.1 presents the architecture of the "dynamic" linking ASGC framework on a computational cluster. A cluster, or a supercomputer, consists of many computational nodes, which are connected by an inner network. These computational nodes serve for performing the regular computation jobs. Both the ASGC allocator and the scientific solver are submitted to and run on these computational nodes through some cluster job submission protocols. However, as discussed in section 4.6.3, the manager process needs to be running on a stable storage. Most modern clusters have at least one host node, where the users build the executable and store the data. The host node can been viewed as a stable storage where there is no computational job being performed, and such node only serves for some light tasks. So the manager process is executed on the host node, which is indicated by the dashed box in figure 4.1. Besides with the manager process, the monitoring units also stay on the host node, since they do not monitor

themselves. There is also a database on the host node, storing the hierarchical surplus data for each level, which will be retrieved by the allocator for generating the collocation points for the next interpolation level. The database is developed using HDF5, of which the implementation details will be given later in this paper.

At beginning of each level, the manager process starts the ASGC allocator to generate the collocation points at this level which serve as the parametric inputs for the scientific solvers. This allocator is also submitted to be running on a computational node. To ensure the successful completion of the allocator, we also employ a monitoring unit, as a protection layer, between the manager and the allocator. There is no direct communication between the manager process and the allocator existed on a separate computational node. Once the manager starts the monitoring unit with the allocator, it only hears from the signal from the monitoring unit on the completion of the execution before starting next step of actions. The monitoring unit repeatedly queries the status of the scientific solver under its monitoring, then reports a success signal back to the manager if the job is completed, or restarts the job if it detected as failed. The detailed monitoring logics will be discussed later in this section. The allocator reads the surplus data from the database and prepare the allocation points set for the next level. These newly generated points are stored in a file on the host node, and will be read by the manager process to create evaluation jobs for next level.

Once the manager process has detected the completion of the allocator, it starts the next level of interpolation using the file containing the collocation points generated by the allocator. Each collocation point is related to one single evaluation job, which is running on the computational nodes. Notice that there is no restriction on the number of processors than can be used by an individual job, which provides the possibility to run very large scale simulations, and is a big improvement compared with the "static" linking framework. Starting next interpolation level requires the completion of all evaluation jobs at current level. Therefore, each evaluation job is equipped with a monitoring unit. This design is more efficient than making the manager process itself to monitor all running jobs. The monitoring unit stays with the scientific solver for its whole life cycle. The monitoring unit restarts the failed job if any sign of failure is detected. After the completion of the evaluation procedure, the scientific solver stores data

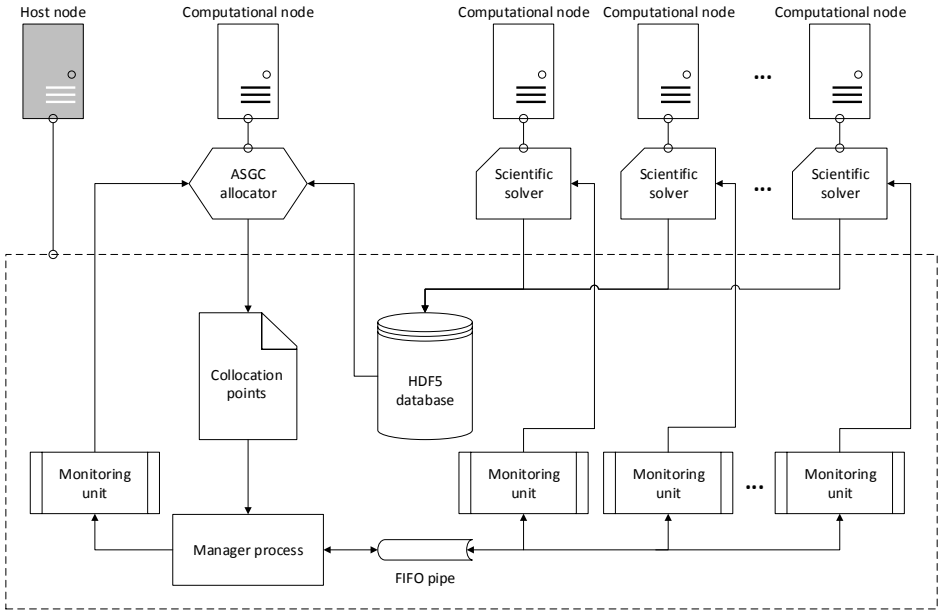


Figure 4.1 Architecture of the "dynamic" linking framework for implementing the ASGC method.

back to a data pool on the host node, which is bound with the HDF5 database to be later retrieved by the allocator for preparing next interpolation level.

4.7.2 Communication between manager process and sub-processes

Notice that the manager process and the monitoring units run in parallel as independent processes on the host node. We need to have a communication channel through which the manager process can hear from the monitoring processes and the monitoring processes can report to the manager process. On the other hand, the number of evaluation jobs can be considerable at each level, especially for high dimensional (very large N) stochastic system. However, the total number of processors that can be used at the same time is usually restricted for most clusters, which means that the number of evaluation jobs can be executed simultaneously is limited. These two problems can be resolved by using the "first-in-first-out" (FIFO) pipe (also known as named pipe), which is a system-persistent tool serving for inter-process communication (IPC) and being available on most operation systems. Such pipe also supports two-way writing and reading, and will hold the reading attempt if the pipe is empty until contents being

available again in the pipe. We can use this feature to control the number of simultaneously running jobs on a cluster.

The communication process between the manager process and the monitoring units is described in Figure 4.2. (1) At beginning of an interpolation level, a pipe is created with m tokens placed inside, where m is the number of scientific solvers that can be executed at the same time. (2) The manager process M keeps taking the tokens out from the pipe, and starts one monitoring unit R as well as a related evaluation job E once a token is taken. After m tokens being taken, the pipe becomes empty, and the reading process from the manager process will be held. At this moment, there are exactly m scientific solvers running on the computational nodes. (3) When the monitoring process detects the successful completion of the evaluation job, it will place the token back into the pipe and terminate itself. (4) Once a token is returned back to the pipe, the held reading action from the manager process is immediately released. Then the manager takes the token again from the pipe to start a new evaluation job, which makes the pipe empty once more and will hold the next reading action from the manager process. This process repeats until all evaluation jobs are submitted. At the end, the manager waits for collecting m tokens back from the pipe then call a completion of this interpolation level.

4.7.3 Functions of the monitoring unit

In this section we introduce the working logic of the monitoring unit. It is difficult to build direct communication between the monitoring unit and the scientific solver, since they exist on different nodes. However, modern clusters always provide commands to query the status of jobs, though the commands may vary due to the type of scheduler installed on the cluster. After a querying command is sent, the scheduler prints detailed information to the request. A scheduler also uses some symbols to express the status of a job. For instance, a PBS type scheduler usually uses the following symbols for related status: “Q” for queuing jobs, “R” for running jobs, “E” for exiting jobs, “C” for cancelled jobs, etc. Then the monitoring unit utilizes these querying commands with the information returned by the scheduler to monitor the status of a scientific solver. The working logic of a monitoring cycle is shown in Figure 4.3.

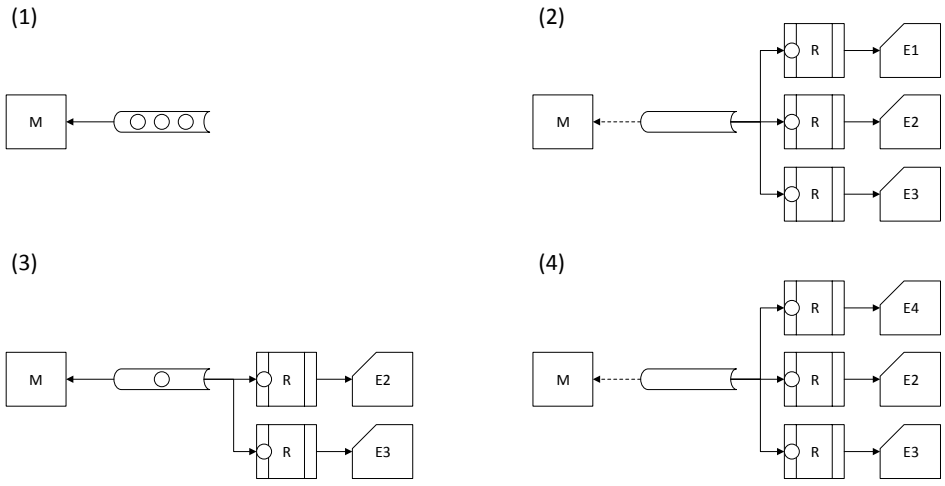


Figure 4.2 Communication between the manager process and the monitoring units, where M represents the manager process, R represents the monitoring unit, and E represents the scientific solver. (1) The pipe is filled with tokens; (2) The manager process takes tokens from the pipe and starts independent monitoring units as well as the scientific solvers. The monitoring units take away all tokens and then the pipe becomes empty, which holds the reading action from the manager process; (3) When a monitoring unit detects the completion of its evaluation job, it places the token back to the pipe and terminates itself; (4) The manager process takes the token and starts a new job, which empties the pipe again and holds the next reading action from the manager.

We introduce the action for each status made by the monitoring unit in the following. Without loss of generality, here we use the PBS scheduler symbols set mentioned above for example.

Checking for completion. To simplify the design of the deterministic solver, it is not required to output any check point during its run-time. The most convenient way to check whether a job has been finished is by checking whether the result file has been generated on the hard disk. It is assumed that outputting the result file is the last step during a jobs execution period. The first step of a monitoring period is checking whether such a result file has been generated. For the first time the monitor detecting the existence of the result file of its monitored job, it will start a timer to wait for a short period, T_c, to let the whole job finish writing to the result file and completely quit. If during T_c, the job normally quits and the job status turns to be "C", then the monitor considers this execution as a successful run and places the token back into the pipe. Otherwise, the monitor views this execution as a failed run and will restart it.

If the job is in the waiting list (status "Q"). At the moment when a job is submitted on the cluster, the scheduler may put the job in a waiting list if the load of a queue exceeds its capacity. Assume that the waiting period, T_W, of a job depends on the conditions of the running jobs in the queue, and also on the required running time of this job. It is difficult to accurately predict T_W since the scheduler may use various protocols. However, the worst case is that a job has to wait until the completion of all the running jobs in the queue before its submission. At the moment when a job is submitted, note the number of running jobs in the queue as m_Q, and their corresponding required running time as T_R^j, $j = 1, 2, \ldots, m_Q$. Then the worst case gives an estimation of the upper bound $\overline{T_W}$ of T_W as $\overline{T_W} \approx \sum_{j=1}^{m_q} T_R^j$. By giving an elastic waiting time T_E, e.g. $T_E = 5\,\text{min}$, if the real waiting time T_W exceeds this upper bound plus the elastic time, $\overline{T_W} + T_E$, then the monitoring unit regards this job as a failed submission and will resubmit the job.

If the job is running (status "R"). All schedulers require users to provide a *maximum run-time* (usually called "wall-time") for each job. The real run-time of a job cannot exceed this maximum run-time. If a job fails to complete when the maximum run-time is reached, the scheduler will force this job to quit. So users should give a good prediction of this max run-time, not too short or too long, otherwise the job may not finish running or will be waiting

for too long time to be started, respectively. However, sometimes when a job crashes but the scheduler fails to kill this job, then the job will still show a running state "R" but actually has become a defunct process. Thus the monitoring unit should be able to detect this failure and restart a failed job. To do this, the monitoring unit also creates a maximum run-time timer when a job is started, which can be viewed as a backup aid of the scheduler in case the scheduler itself fails. Then if a job crashes and its run-time has exceeded the maximum run-time but the scheduler fails to detect this case, the monitoring unit is still able to detect this failure, then marks this job with an ill-state and restarts it.

If the job is abnormally canceled (status "C" without output). During one monitoring period, if the output file has not been generated yet but the job status turns to be "C", it means that the job has been abnormally terminated. This failure may result from either a software exception or a hardware issue, or both. If this situation is detected, the monitoring unit will immediately restart the failed job. However, sometimes this failure might be caused by some bugs or wrong configurations of the scientific solver itself. If this is true, even if the job is restarted, the failure will occur repeatedly. To prevent this, we use a maximum resubmission counts to restrict the number of attempts of resubmission of a single job, e.g. 5 times at most to resubmit a single job. If any job has been resubmitted for such number of counts, it is very possible that bugs or wrong configurations exist within the scientific solver, and the monitoring unit will refuse the next submission attempt for this job and reports a failure signal to the manager process. When the manager process receives this signal, it will force to terminate all related processes and inform the user to check the scientific solver and its configurations.

Otherwise. If no ill-status is detected during one monitoring period, the monitoring unit will do nothing but just simply wait for a certain period, then starts the next monitoring period until an ill-status is found or the job successfully completes.

4.7.4 Data storage optimization

A frequent operation for the ASGC method is checking whether a newly generated sparse grid point has already been generated by other parent points from upper level according to the algorithm introduced in section 4.4. Each grid point can been represented by two integer

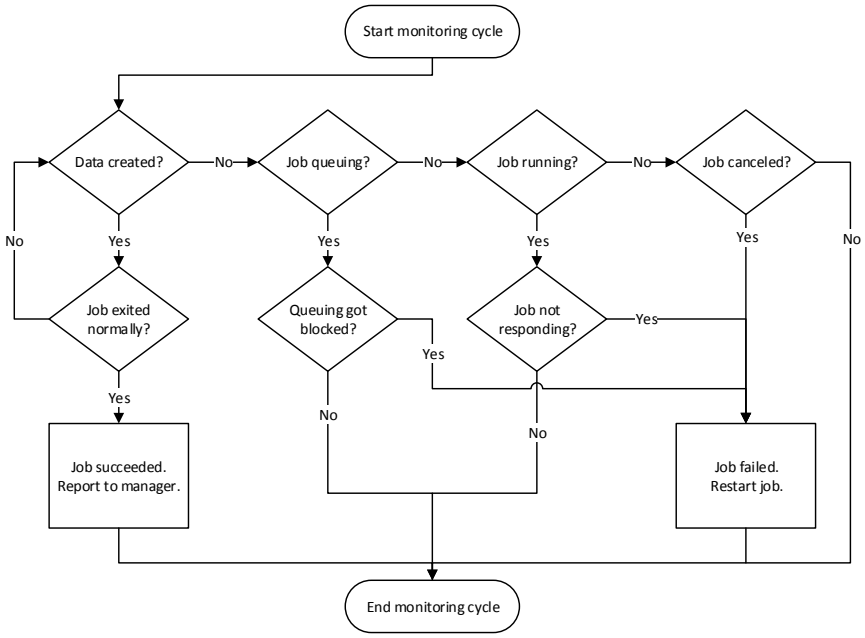


Figure 4.3 Flowcahrt of the monitoring unit working procedure.

vectors, $\boldsymbol{i}$ and $\boldsymbol{j}$, where $\boldsymbol{i}$ represents the interpolation depth in each dimension and $\boldsymbol{j}$ indicates the grid point position in each dimension. All the newly generated points are unique at current level, which form a set. In the "static" design, the C++ data structure "set" was utilized as a natural container for the storage purpose, which hosts in RAM. The C++ set container automatically checks the existence of a given point before the inserting operation. However, when the stochastic dimension becomes considerable, the RAM based data structure cannot be used due to the limited capacity of RAM. Then we need a well organized database to support stable data storage and efficient data inquiry operations. The $\boldsymbol{i}$-$\boldsymbol{j}$ indices couple serves as the key in the database for the simulation result from the scientific solver at each sparse grid point.

The key issue focuses on searching the existence of a given index, $\boldsymbol{i}$-$\boldsymbol{j}$, for a newly generated sparse grid point. For a given $\boldsymbol{i}$, there can be multiple $\boldsymbol{j}$ indices. So one searching strategy is: search the existence of $\boldsymbol{i}$ first, if this $\boldsymbol{i}$ does not exist, then we are sure that this is a new point, so stop searching and insert this new point, otherwise search for the existence of $\boldsymbol{j}$ under the found $\boldsymbol{i}$. This means that all the $\boldsymbol{j}$'s related to the same $\boldsymbol{i}$ should be piled together as a group, and the index $\boldsymbol{i}$ is the key to access this group. Notice that the number of $\boldsymbol{j}$'s for a certain $\boldsymbol{i}$ is

at most $2N$, where N is the stochastic dimension. So the searching inside the "bucket" of such $\boldsymbol{i}$ can be trivial: looping over all the $\boldsymbol{j}$'s in the bucket to check the existence. The complexity of looping the bucket can be viewed as $O(1)$. Unfortunately, this looped searching cannot be applied on searching the $\boldsymbol{i}$ indices for the large quantity at deeper interpolation level. The looped searching has complexity of $O(N^2)$, which cannot be accepted for high performance of simulations. Thus a smarter searching strategy needs to be designed to manage and search for the $\boldsymbol{i}$ indices.

The HDF5 library is utilized to manage the indices on hard disk. The HDF5 library manage data in a similar way as the operation systems manage files. The HDF5 organizes data on hard disk into groups. Each group has a unique name, as the key to access to the data in this group. HDF5 builds a separate index for all the names, which is organized using tree data structure providing almost $O(Nlog(N))$ searching complexity. This is a significant improvement compared with $O(N^2)$. The way HDF5 organizing data is suitable for organizing the indices of sparse grid points, where each group contains all the $\boldsymbol{j}$'s of a certain $\boldsymbol{i}$, of which the group name is a representation of $\boldsymbol{i}$. Each $\boldsymbol{i}$ index is an N integers vector, $\boldsymbol{i} = (i_1, i_2, \cdots, i_N)$. It is inappropriate that using $\boldsymbol{i}$ directly as the group name, since that the string can be rather lengthy for high stochastic dimensions. However, the sparse structure of $\boldsymbol{i}$ indices provides the possibility to compress the name string.

Notice that all the $\boldsymbol{i}$ indices are grew from the staring index $(1, 1, \cdots, 1)$ (N 1's in the vector). As the interpolation depth increases by 1, a certain position among the N elements in this vector also obtains one unit increment. If $\boldsymbol{i}$ is subtracted by the identity vector, then most of the elements are zeros. The number of left non-zero digits in the vector is less than the interpolation depth, which is a relatively small number. Thus the expression of group name can be compressed by using these non-zero digits in the vector. For the non-zero digit $i_m = k$, $k \leqslant L$, where m is the position of this non-zero digit in the vector and k is the interpolation depth, it means k 1's have been added to such position m. So the name can be represented in such a way as how many 1s have been added to a certain position. For the index$\boldsymbol{i}$, denote $\boldsymbol{i}' = (i'_1, i'_2, \cdots, i'_N) = (i_1, i_2, \cdots, i_N) - (1, 1, \cdots, 1)$, which forms a sparse vector. Create a set $I = \{(m_1, k_1), (m_2, k_2), \cdots, (m_p, k_p)\}$, where m_i represents the non-zero

positions, and k_i represents the value of i_{m_i}, $i_{m_i} = k_i$, $i = 1, 2, \cdots, p$, p is the number of non-zero digits in $\boldsymbol{i}'$, and we have $k_1 + k_2 + \cdots + k_p = L$. The name string of $\boldsymbol{i}$ can be expressed as $m_1 \cdots m_1 m_2 \cdots m_2 \cdots m_p \cdots m_p$, where there are k_1 m_1's, k_2 m_2's, ..., and k_p m_p's. For example, if $N = 5$, $\boldsymbol{i} = (1, 2, 1, 3, 1)$, then $\boldsymbol{i}' = (0, 1, 0, 2, 0)$. This gives that $I = \{(2, 1), (4, 2)\}$. Then the name string becomes "244". it is seen that when the stochastic dimension is very high (~ 100), and the interpolation depth is at regular level (~ 10), the name string is quite compressed in such a way.

4.8 Numerical examples

4.8.1 Stochastic elliptic problem

As an example of utilizing the ASGC method with the above fault-tolerant framework to solve stochastic partial differential equations, here we consider a stochastic elliptic problem which has been discussed by Nobile et al. (2008); Ma and Zabaras (2009). This equation governs a heat diffusion process on a closed domain where the thermal diffusivity is a stochastic field. The governing equation with the boundary condition is given as:

$$-\nabla \cdot (a_N(\omega, \boldsymbol{x}) \nabla u(\omega, \boldsymbol{x})) = f_N(\omega, \boldsymbol{x}) \quad \boldsymbol{x} \in D, \omega \in \Omega, \tag{4.9}$$

$$u(\omega, \boldsymbol{x}) = 0 \quad \boldsymbol{x} \in \partial D, \omega \in \Omega, \tag{4.10}$$

where the physical domain $D = [0, 1] \times [0, 1]$ is a 2D square domain, $u(\omega, \boldsymbol{x})$ is the temperature field, $a_N(\omega, \boldsymbol{x})$ is the stochastic thermal diffusivity with N the stochastic dimension, and $f_N(\omega, \boldsymbol{x})$ is a source term taking the form $f_N(\omega, x, y) = \cos(x)\sin(y)$. Notice that the source term is selected as a deterministic equation inside the physical domain, which avoids introducing extra errors from domain discretization. The stochastic field $a_N(\omega, \boldsymbol{x})$ can be expressed by a series of random variables $\{Y_n(\omega)\}_{n=1}^N$:

$$\log(a_N(\omega, x) - 0.5) = 1 + \sum_{n=1}^{N} \left(\frac{\sqrt{\pi} L}{2}\right)^{1/2} \exp\left(\frac{-(n-1)^2 \pi^2 L^2}{8}\right) \cos(2\pi x(n-1)) Y_n(\omega), \tag{4.11}$$

where $Y_n(\cdot)$ are independent uniformly distributed random variables on $[-\sqrt{3}, \sqrt{3}]$. Notice that this expansion is simplified as a 1D equation. This expansion makes the random field a_N similar

to the random field with exponential covariance kernel. In this expansion, L is the correlation length, for which smaller value gives a slower convergence rate.

One significant feature of our fault-tolerant framework is its capability of solving stochastic partial equations with large physical scales, which requires more computational resources to numerically solve each deterministic equation. Ma and Zabaras (2009) solved the deterministic problem using finite element method, where the physical domain is discretized into 30×30 bilinear quadrilateral elements, where the size of discretization is quite small. According to the static linking between the allocator and the scientific solver in their "one-program" design, the deterministic solver is restricted to execute only on a single processor, which does meet the requirements of most of large scale engineering problems. Besides, the scientific solver has to be hard-coded in to the executable before compiling and executing, which increases the difficulty of applying such a framework as an agile toolkit for stochastic analysis. By defining the dynamic linking strategy, this restriction of computational resources on the deterministic solver is removed. The size of the solver is user defined and theoretically can be any large, as long as there is enough computational resources. Besides, by employing the fault-tolerant method, successful execution of the stochastic interpolation is guaranteed.

To showcase the performance of the fault-tolerant framework, we increase the number of elements to 300 by 300, and use 4 processors to solve one deterministic equation. To simplify the problem, here we run this problem for low stochastic dimension, $N = 5$, and pick $L = 1$ and $L = 1/8$ to illustrate the propagation of sparse grid points and evolution of interpolation errors with levels. Figure 4.4 plots the mean and variance of the temperature fields for the case of $L = 1$. Here we see that both the mean and variance of the temperature fields have a higher distribution around the center of the domain. Figure 4.5 gives the mean and temperature distributions over the physical domain, which show similar structure as those of the case of $L = 1$. The convergence histories of both $L = 1$ and $L = 1/8$ are plotted in Figure 4.6.

Another feature of the fault-tolerant framework is that it is appropriate for solving stochastic partial differential equations with very high stochastic dimensions (large N). First, as the stochastic dimension is high, the number of sparse grid collocation points can be extremely considerable, which requires using more computational resources to solve these deterministic

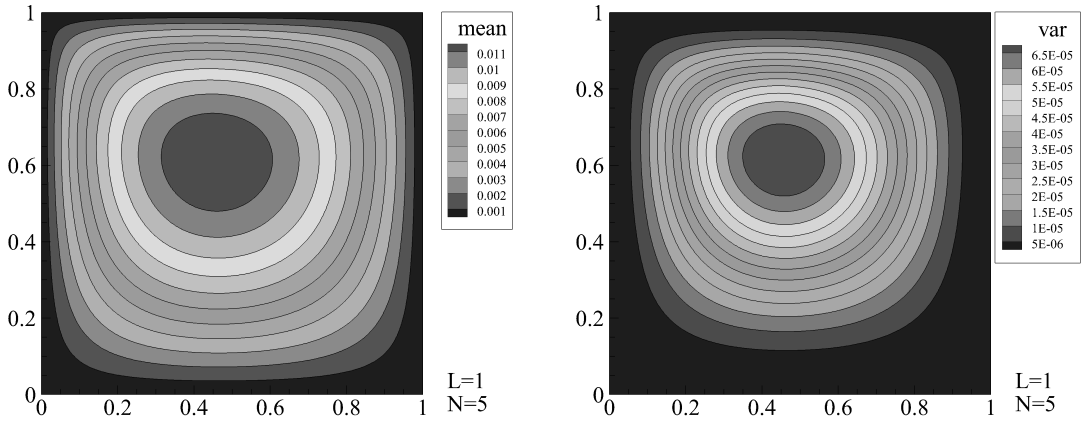


Figure 4.4 Mean and variance of temperature field for stochastic elliptic problem with correlation length $L = 1$. Stochastic dimension $N = 5$, discretization with 300×300 elements.

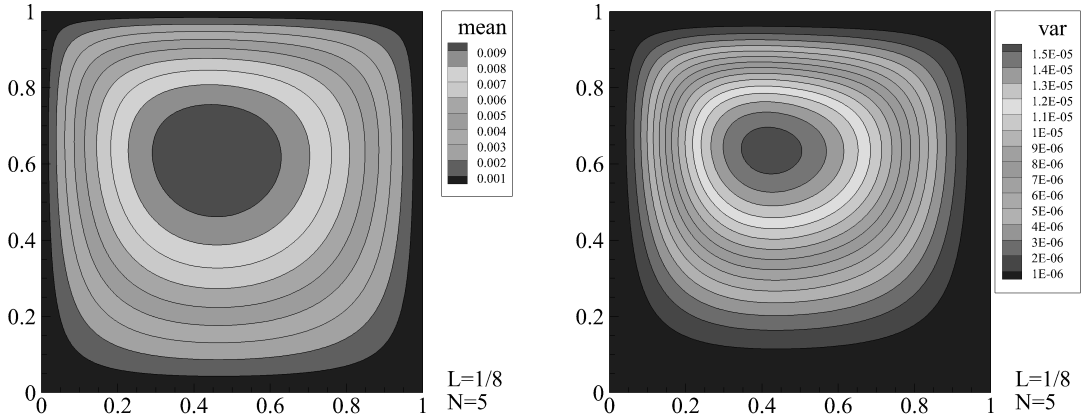


Figure 4.5 Mean and variance of temperature field for stochastic elliptic problem with correlation length $L = 1/8$. Stochastic dimension $N = 5$, discretization with 300×300 elements.

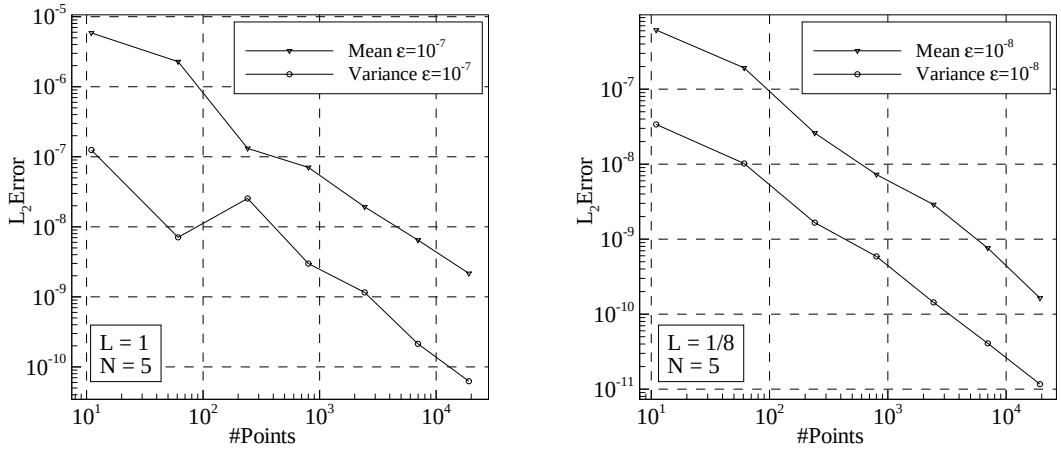


Figure 4.6 Convergence histories for the stochastic elliptic problems with large physical scales. Stochastic dimension $N = 5$, 300×300 elements. Left:$L = 1$; right: $L = 1/8$.

problems. By the dynamic linking strategy, since that the deterministic solver is separated from the sparse grid allocator, the number of processors can be kept at a constant level. Second, for high stochastic dimension interpolations, the quickly increased solutions to these deterministic equations bring the necessity for safe and efficient large data manipulations. This challenge is successfully resolved by utilizing the HDF5 data library with its hard disk based database and our index related fast searching algorithm.

In previous studies, Ma and Zabaras (2009) chose the highest stochastic dimension N for equations (4.9) as 75, and solved the problem with their "one-program" static linking framework. Even higher stochastic dimension may bring the potential risk as memory over flow for their RAM based storage strategy. Here we showcase the capability of our design by increasing the dimension to 100, and even higher, to 200 to showcase the capability of dealing with very high throughput simulations using our framework. For such high dimension, the number of collocation points can be considerable at each level, which increases the chance of the occurrence of failed jobs. This makes the fault-tolerability necessary to ensure the completion of the hierarchical process. To focus on the high dimensionality, we reduce the number of elements to 30×30, to save run-time for a single execution. For making the load of the two tests comparable, we select $L = 1/2$ for $N = 100$, and $L = 1$ for $N = 200$. Besides, we only showcase the capability by using the first 4 interpolation levels for both cases.

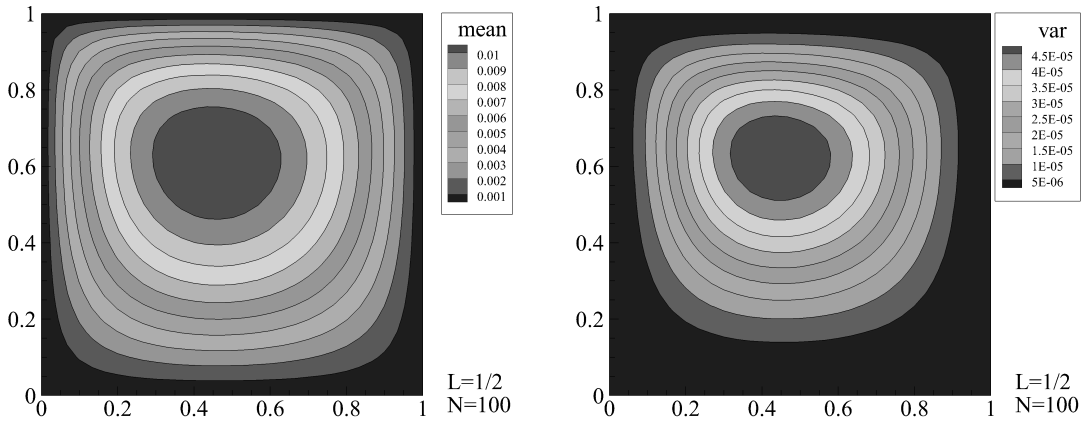


Figure 4.7 Mean and variance of temperature for stochastic elliptic problem with high stochastic dimensions, $N = 100$.

Mean and variance of the stochastic solution for $N = 100$ are plotted in Figure 4.7, and mean and variance of the case of $N = 200$ are shown in Figure 4.8. Figure 4.9 shows the convergence histories of these two high stochastic dimension simulations. As seen from the plots, the mean and variance fields have similar structure as those of the lower stochastic dimension cases. However, the convergence histories clearly tell that the numbers of interpolation points at level 4 for both the two cases have already exceeded the ones at the highest levels of the lower stochastic dimension cases. Thus the results demonstrate that our fault-tolerant ASGC framework has no difficulty in handling the simulations with high throughput interpolations, as stochastic dimensional safe.

As a highlight of the current framework, its fault-tolerability is investigated in this paper. The tested problem is chosen as the above stochastic elliptic equation with small physical scale (30×30 elements), low stochastic dimension ($N = 2$ and $N = 3$) with 5 interpolation levels. The programs are run on a stable local computer with four cores. The purpose of using small problems and stable machine is to reduce the probability of natural processor failure to a negligible level. Thus the processor exceptions can be imitated by adding artificial failure to the running processors, which makes the strength of processor exceptions controllable. The strength of processor exceptions of a given machine can be described by the probability p of the occurrence of processor failures on a given processor, assuming all the nodes are identical. In

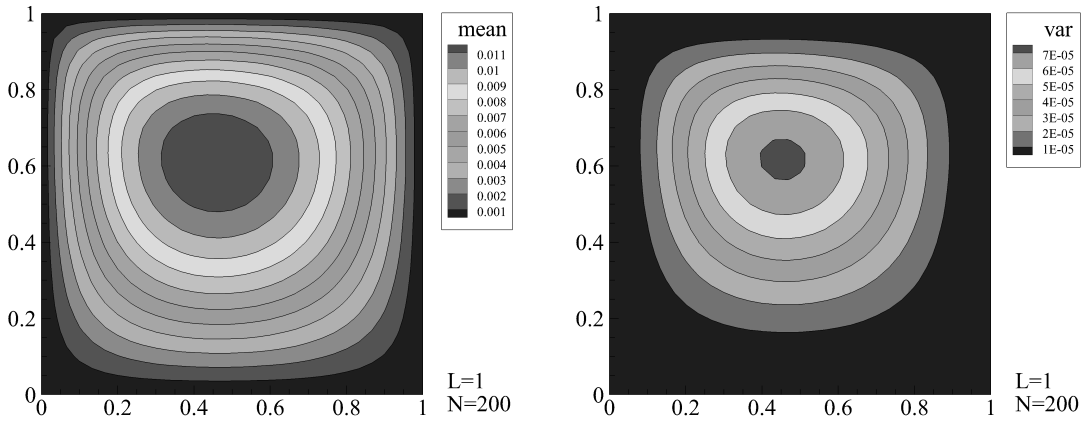


Figure 4.8 Mean and variance of temperature for stochastic elliptic problem with high stochastic dimensions, $N = 200$.

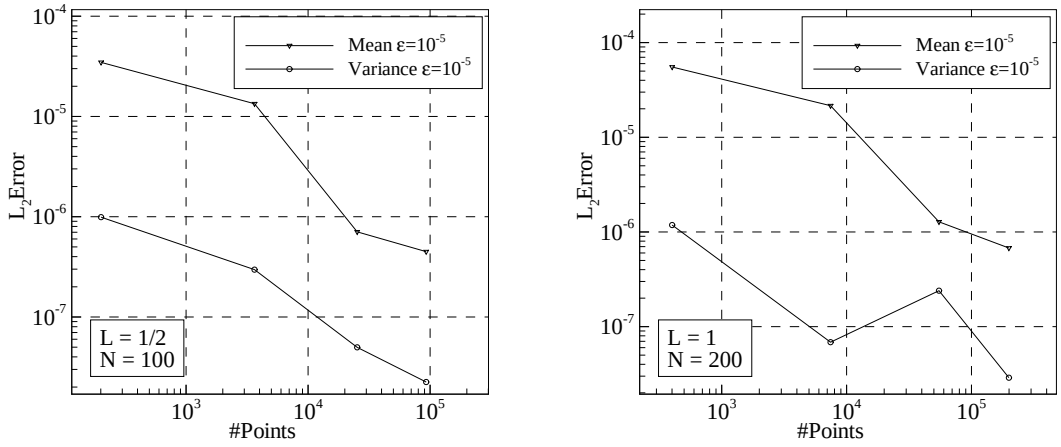


Figure 4.9 Convergence histories for stochastic elliptic problems with high stochastic dimensions. Left:$N = 100$; right: $N = 200$.

this test, p is chosen as 0.0625 to 0.75, to represent relatively stable machine and very unstable machine. As a mimic of the natural processor failure, at the beginning of the execution of each job, a random number r is assigned to the processor where the job is running. In each test, p is fixed. The situation $r < p$ represents a failure occur on the processor and the job will be forced to quit. Then the monitor process is able to detect this failure and restarts the job. For each p, three simulations are run and the average run time $E[T]$ is recorded. This $E[T]$ is then scaled by the run time recorded for the case $p = 0$, as the stable machine average run time $E[T_0]$.

The evolution of run-time with p is shown in Figure 4.10. The tests are parallelized in both 2 jobs running simultaneously and 4 jobs running simultaneously, to investigate the effect of parallelization on the tolerability. As seen in the figure, the average run time grows fast as the instability of machine increase. However, even for the most unstable condition, the increase of run time is able to be controlled in a limited bound, compared with $E[T_0]$. Thus the fault-tolerability of the current framework is guaranteed. The figure also reflects that the increase of run time is almost independent with the stochastic dimension and the parallelization level. These curves can be approximated by the expression of p, $\widehat{E[T]/E[T_0]} = 1/(1-p)^{\alpha}$, where the parameter α reflects the strength of the tolerability. For the current problem, $\alpha = 1.75$ gives a good approximation. This interpolation of the run time with machine failure probability p is plotted as the dashed curve in Figure 4.10, where very good agreement is obtained. Therefore, once the condition of a real cluster is known, the plot in Figure 4.10 can be used to estimate the run time of the simulations.

4.8.2 High discontinuity interpolation

The dynamic linking strategy provides our framework with the flexibility of handling almost all the parametric driven high throughput simulations, which is saying, as a versatile tool, this framework has not to be restricted on its primary usage for stochastic analysis, but is able to be linked to various types of engineering solvers. The fault-tolerant framework defines clear input-output format for users. The input to the framework at each interpolation level is a list of parameters for the scientific solvers, and the output can be any type generated by the scientific

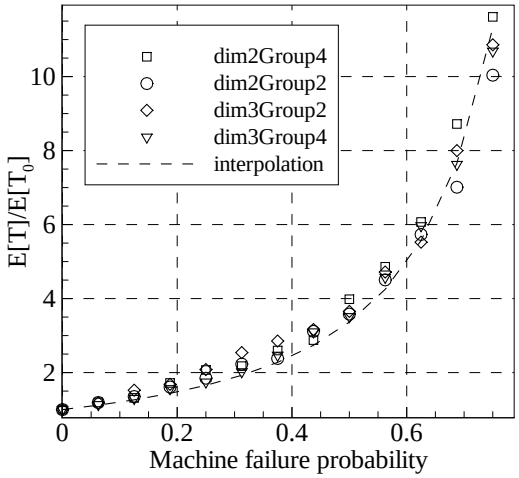


Figure 4.10 Increase of average run time with the probability of the occurrence of artificial processor exceptions. The dashed curve is the interpolation of the run time with machine failure probability p.

solver according to its design. Besides, the scientific solver is soft linked to the framework, not hard coded, which provides convenience for most types of engineering solvers - either user implemented or commercial solvers. One can also use this framework as a batch processing tool to manage large number of simulations to save a lot of manual work. Currently our framework support local mode, cluster mode and hybrid mode for selecting the way of executing. Local mode allows users to run all jobs on a local computer. Cluster mode allows users to run the simulations on a cluster, but each job has to be executed on at least node (a node may have multiple processors). The hybrid mode allows users to run the simulations on a cluster where each node can run a bunch of jobs. The local mode is appropriate for light scientific solvers with small number of jobs. The cluster mode is suitable for relatively heavy duty scientific solvers requires multiple processors to perform. The hybrid mode is designed for very large number of light scientific solvers which requires about one or two processors to run.

In this paper we illustrate the flexibility of this framework with an interesting problem. Remind that the sparse grid collocation method is appropriate for high discontinuity for which more grid points will be located around the discontinuity curves [Ganapathysubramanian and Zabaras (2007)]. Here we impose a “color map” on a square domain and use the ASGC method

Figure 4.11 A five-colored map in a square domain. Each block is assigned with a color of unique gray scale.

to detect the boundaries between two blocks with different gray scales. Figure 4.11 shows the color map with 5 blocks with different gray scales. At each given point on the map, the gray scale can be extracted from the image with any image handling library, as the scientific solver for this problem. Thus we can use ASGC method to interpolate this map. As interpolation level goes deeper, more grid points will be located around the discontinuity borders to form clear boundaries between two blocks. Figure 4.12 shows the growth of the grid points as interpolation level increases. We see that more collocation points are placed along the borders between two blocks. On the other hand, the number of grid points almost stops increasing inside each block away from the borders at higher interpolation levels.

4.9 Conclusions

From the analysis and discussion shown in this paper, we see that the fault-tolerant framework improve the use of sparse grid collocation method on stochastic interpolation. We first discussed the several shortcomings of the previous static linking framework, to show that it is not scalable to large scale deterministic problems, and is vulnerable or fragile to high stochastic dimensions. To cope with these difficulties, we introduced the fault-tolerant system design theories. A series of criterion conduct the design of a fault-tolerant framework. Then to implement this fault-tolerant framework, we defined the dynamic linking strategy. For the sparse

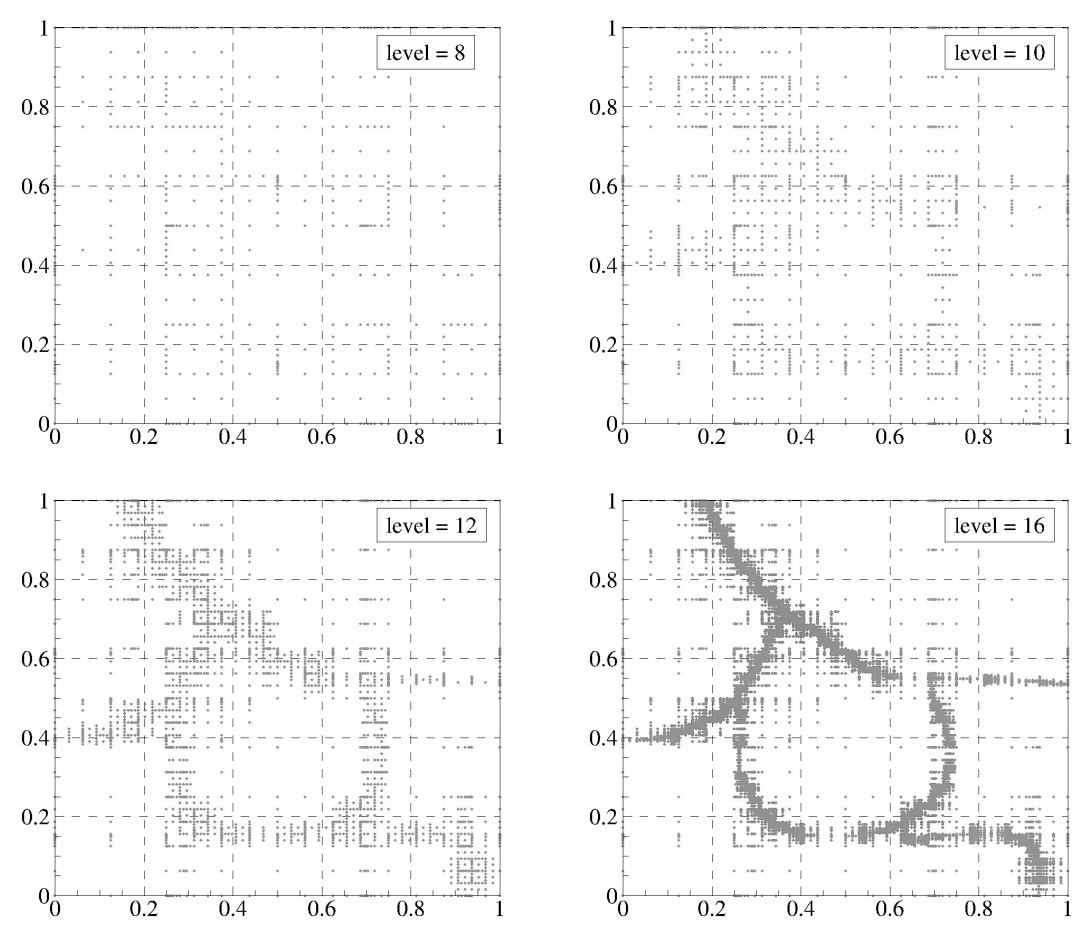


Figure 4.12 Growth of the ASGC collocation points for detecting the discontinuity in the five-colored map at different interpolation levels. More collocations points are placed along the borders between two blocks.

grid allocator, to avoid the risk of memory overflow as size of problem increases, the simulation results are stored on hard disk and organized using HDF5 data structures. The fault-tolerant functionality is implemented by a monitoring unit equipped for each deterministic solver. All possible failures have been defined in the monitoring unit. Whenever a failure occurs, the monitoring unit is able to detect the failure by actively querying the job status and immediately restarts the failed job on a healthful node. This guarantees the simulation process being able to complete for any level of stochastic simulations. At the end of this paper, several numerical examples were introduced to validate the performance of the fault-tolerant framework. These examples illustrated that: 1) the current framework is competent for large scale simulations; 2) it is capable to deal with high stochastic dimension; 3) it provides high flexibility for any other types of scientific solvers for not only stochastic analyzing usage. Furthermore, the allocator can also be designed with different sampling strategies, e.g. using Monte Carlo sampling points generator. The current framework can link between the input parameters generator and the deterministic solver in a very convenient way, as a versatile tool, to serve for various engineering problems

CHAPTER 5. GENERAL CONCLUSION

5.1 General discussions

In this thesis, we have discussed the details of deriving the finite element formula for solving coupled equation systems, especially for the complex fluid systems. We have provided solutions to both the numerical scheme to each equation set and the coupling between the two physical systems. For the Navier-Stokes equation, after comparing the accuracy and efficiency of several finite element implementations, we finally adopt the four-step fractional method which separates the pressure equation from the momentum equation thus reduces the degree of freedoms for solving the flow equations. The SUPG term is incorporated in all the convection dominant equations to eliminate the spurious oscillations of velocity in the solutions, which enables us to treat wider range of Reynolds numbers for laminar flows. Each nonlinear equation is linearized by the Newton-Raphson scheme at each time step and is then solved by PETSc SNES solver. The coupled equation sets are then treated in a semi-coupled way, where each equation set is solved solely and sequentially, and then the solution is updated in an iterative pattern at each time step until convergence of all unknown variables has been reached.

We applied this finite element numerical scheme on investigating the behaviors of two coupled equation systems. For the problem of flow past a heated cylinder, we numerically verified the existence of three vortex shedding patterns behind a heated cylinder, as the staggered shedding pattern (also named as the "von Karman vortex street") for lower cylinder surface temperature, the simultaneous shedding pattern (also known as the "Kelvin-Helmholtz vortex pattern") for higher cylinder surface temperature, and a transition state where a long quiescent wake zone is formed behind the cylinder (also called the "dead-flow zone" pattern) which exists in a short intermediate range of cylinder surface temperature. We further studied the

relation between the vortex shedding pattern and the cylinder aspect ratio, and found that the velocity gradient along the cylinder axis direction plays important role on affecting the formation of vortex shedding patterns behind the heated cylinder. We further demonstrated the capability of this numerical scheme on solving coupled equation systems by showcasing its application on simulating the multiphase flows. We verified that there should be at least four elements through the interface to guarantee the accuracy of the numerical solution to the interface motion. We also proved the capability of the numerical framework on solving 3D flows on complicated geometries.

Finally we introduced the design and implementation of the fault-tolerant framework for handling high throughput simulation tasks, especially its usage on the stochastic analysis on complex systems. The key idea of the design of this framework is separating the scientific solver apart from the input parameters generator (the allocator) as an independent process, which enables the scientific solver to use multiple processors for large scale problems. This "dynamic linking" design (between the scientific solver and the allocator) also breaks the direct risk escalation path from the scientific solver to the whole project as in the old-fashioned "static linking" strategy - where the failure of a single process will cause the termination of the whole project according to the MPI based design. This means that in the new design, the failed scientific solver is recoverable. This recovery feature is enabled by incorporating the monitoring unit into the framework. The challenge of large data set management is resolved by utilizing the HDF5 database and its managing library, which provides us efficient data storing and retrieving functionality. This batch processing framework also has a user friendly interface through which users are able to link their own solvers without knowing the details inside this tool. This feature greatly extends its potential usage on various engineering problems.

Before the completion of this thesis, our numerical framework as well as the finite element solvers have been widely used on analyzing cutting-edge scientific and engineering problems. Our framework helped Amini et al. (2013) exploring a new way of controlling the fluid streams in a microfluidic channel by simply placing a sequence of cylindrical pillars within the channel. The researchers found that the fluid particles are able to form several patterns after passing a single pillar under different controlling conditions. Our numerical analysis expanded the sim-

ulations on a wide range of these controlling parameters then found more streaming patterns and obtained a phase portrait of the pattern transition map. Guo (2013) applied the stochastic analysis toolkit AdaGiO on investigating the reliability of wind turbine blade designs affected by the uncertainties from the wind turbulence and the fabrication process of blade composite layers. Jaeger et al. (2012) numerically investigated the effect of nano-scaled surface roughness on the perturbation of the velocity of the low Reynolds number laminar flows in the microfluidic channels. They found that velocity profile can be significantly perturbed by the surface roughness as deep to the central position inside the channel. It is difficult to obtain these velocity profiles only from experimental observations for such small length scales. Other successful examples of utilizing our framework include, but not limited to, studying the stochastic behavior of air flow in the buildings and obtaining a phase portrait of morphology patterns of organic solar cells during fabrication process. Interested readers may refer to these papers for further details.

5.2 Potential future work

(1) Simulations of non-Newtonian fluids have attracted more and more attentions. Human blood is a typical non-Newtonian fluid. It is usually very difficult to experimentally observe the behavior of blood flow due to its small scale and its nontransparent color. Thus incorporating the capability of simulating non-Newtonian fluid flow into the current framework is highly valuable and has many engineering potentials.

(2) From the study of simulating multiphase flows we have seen that we usually need to face the challenging from the requirement of very fine resolution of the computational grid, since it significantly increases the consumption of computational resources. However, most of time we don't require such fine spatial resolution over the whole domain. For example, only the interface between two fluid components of multiphase flow needs to be discretized finely, but the far outer region does not. Therefore it will be great to incorporate the adaptive mesh refinement (AMR) technique into the current finite element solver where we can dynamically adjust the spatial grid resolution according to local requirement of accuracy.

(3) Amini et al. (2013) have pointed out an exciting direction for flow manipulation in microfluidic channels. It is possible to form complicated streaming configurations by just simply placing sequential cylindrical pillars in the channel. Thanks to our finite element solver and the high throughput batch processing tool, we have accumulated a huge number of simulation data for the flow deformation under various controlling conditions, such as Reynolds number, pillar position, pillar size, and channel cross-section aspect ratio. With these data, we can better understand how the fluid particles react to these control parameters, and we are aiming at establishing a data library for mapping the flow deformation to the control parameters. It is possible to build the inverse engineering to such problem where we are able to find the appropriate control parameter given an arbitrary desired flow pattern. Stoecklein et al. (2014) have taken the first step out onto this exciting journey.

BIBLIOGRAPHY

Ahmed, A. and Rangel, R. (2002). Metal droplet deposition on non-flat surfaces: effect of substrate morphology. *International journal of heat and mass transfer*, 45(5):1077–1091.

Amini, H., Sollier, E., Masaeli, M., Xie, Y., Ganapathysubramanian, B., Stone, H. A., and Di Carlo, D. (2013). Engineering fluid flow using sequenced microstructures. *Nature communications*, 4:1826.

Anderson, D., McFadden, G. B., and Wheeler, A. (1998). Diffuse-interface methods in fluid mechanics. *Annual review of fluid mechanics*, 30(1):139–165.

Anderson, J. D. et al. (1995). *Computational fluid dynamics*, volume 206. Springer.

Balay, S., Brown, J., , Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2013a). PETSc users manual. Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory.

Balay, S., Brown, J., Buschelman, K., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H. (2013b). PETSc Web page. http://www.mcs.anl.gov/petsc. Accessed on Jun 5, 2012.

Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. (1997). Efficient management of parallelism in object oriented numerical software libraries. In Arge, E., Bruaset, A. M., and Langtangen, H. P., editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press.

Boussinesq, J. (1897). *Théorie de l'écoulmnent tourbillonnant et tumultuex des liquides dans les lits rectilignes à grande section*, volume 1. Gauthier-Villars.

Boyer, F. (2002). A theoretical and numerical model for the study of incompressible mixture flows. *Computers & fluids*, 31(1):41–68.

Brackbill, J., Kothe, D. B., and Zemach, C. (1992). A continuum method for modeling surface tension. *Journal of computational physics*, 100(2):335–354.

Bray, A. (1994). Theory of phase-ordering kinetics. *Advances in Physics*, 43(3):357–459.

Brooks, A. N. and Hughes, T. J. (1982). Streamline upwind/petrov-galerkin formulations for convection dominated flows with particular emphasis on the incompressible navier-stokes equations. *Computer methods in applied mechanics and engineering*, 32(1):199–259.

Bungartz, H.-J. and Griebel, M. (2004). Sparse grids. *Acta numerica*, 13(1):147–269.

Bussmann, M., Mostaghimi, J., and Chandra, S. (1999). On a three-dimensional volume tracking model of droplet impact. *Physics of Fluids*, 11:1406.

Caginalp, G. and Chen, X. (1998). Convergence of the phase field model to its sharp interface limits. *European Journal of Applied Mathematics*, 9(4):417–445.

Cahn, J. W. and Hilliard, J. E. (1958). Free energy of a nonuniform system. i. interfacial free energy. *The Journal of Chemical Physics*, 28(2):258–267.

Ceniceros, H. D., Nós, R. L., and Roma, A. M. (2010). Three-dimensional, fully adaptive simulations of phase-field fluid models. *Journal of Computational Physics*, 229(17):6135–6155.

Cervera, M., Codina, R., and Galindo, M. (1996). On the computational efficiency and implementation of block-iterative algorithms for nonlinear coupled problems. *Engineering computations*, 13(6):4–30.

Chang, K.-S. and Sa, J.-Y. (1990). The effect of buoyancy on vortex shedding in the near wake of a circular cylinder. *Journal of Fluid Mechanics*, 220:253–266.

Chang, Y.-C., Hou, T., Merriman, B., and Osher, S. (1996). A level set formulation of eulerian interface capturing methods for incompressible fluid flows. *Journal of computational Physics*, 124(2):449–464.

Choi, H., Choi, H., and Yoo, J. (1997). A fractional four-step finite element formulation of the unsteady incompressible navier-stokes equations using supg and linear equal-order element methods. *Computer Methods in Applied Mechanics and Engineering*, 143(3):333–348.

Chorin, A. J. (1968). Numerical solution of the navier-stokes equations. *Mathematics of computation*, 22(104):745–762.

Clarkson, K. L. (1994). An algorithm for approximate closest-point queries. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 160–164. ACM.

Deng, Q., Anilkumar, A., and Wang, T. (2007). The role of viscosity and surface tension in bubble entrapment during drop impact onto a deep liquid pool. *Journal of Fluid Mechanics*, 578(1):119–138.

Ding, H., Spelt, P. D., and Shu, C. (2007). Diffuse interface model for incompressible two-phase flows with large density ratios. *Journal of Computational Physics*, 226(2):2078–2095.

Dong, S. and Shen, J. (2012). A time-stepping scheme involving constant coefficient matrices for phase-field simulations of two-phase incompressible flows with large density ratios. *Journal of Computational Physics*, 231(17):5788–5804.

Edgerton, H. E. and Killian, J. R. (1954). *Flash!: Seeing the unseen by ultra high-speed photography*. CT Branford Co.

Feng, X. (2006). Fully discrete finite element approximations of the navier–stokes–cahn-hilliard diffuse interface model for two-phase fluid flows. *SIAM journal on numerical analysis*, 44(3):1049–1072.

Feng, X., He, Y., and Liu, C. (2007). Analysis of finite element approximations of a phase field model for two-phase fluids. *Mathematics of Computation*, 76(258):539–571.

Ganapathysubramanian, B. and Zabaras, N. (2007). Sparse grid collocation schemes for stochastic natural convection problems. *Journal of Computational Physics*, 225(1):652–685.

Ganapathysubramanian, B. and Zabaras, N. (2008). A seamless approach towards stochastic modeling: Sparse grid collocation and data driven input models. *Finite Elements in Analysis and Design*, 44(5):298–320.

Ghanem, R. G. and Spanos, P. D. (1991). *Stochastic finite elements: a spectral approach*, volume 41. Springer.

Green, R. and Gerrard, J. (1993). Vorticity measurements in the near wake of a circular cylinder at low reynolds numbers. *Journal of Fluid Mechanics*, 246:675–691.

Griebel, M. (1998). Adaptive sparse grid multilevel methods for elliptic pdes based on finite differences. *Computing*, 61(2):151–179.

Guermond, J.-L. and Quartapelle, L. (2000). A projection fem for variable density incompressible flows. *Journal of Computational Physics*, 165(1):167–188.

Gueyffier, D., Li, J., Nadim, A., Scardovelli, R., and Zaleski, S. (1999). Volume-of-fluid interface tracking with smoothed surface stress methods for three-dimensional flows. *Journal of Computational Physics*, 152(2):423–456.

Guo, Q. (2013). Incorporating stochastic analysis in wind turbine design: data-driven random temporal-spatial parameterization and uncertainty quantication.

Hasan, O. and Prosperetti, A. (1990). Bubble entrainment by the impact of drops on liquid surfaces. *J. Fluid Mech*, 1219:143.

Hatton, A., James, D., and Swire, H. (1970). Combined forced and natural convection with low-speed air flow over horizontal cylinders. *Journal of Fluid Mechanics*, 42(01):17–31.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580.

Hohenberg, P. C. and Halperin, B. I. (1977). Theory of dynamic critical phenomena. *Reviews of Modern Physics*, 49(3):435.

Hu, H. and Koochesfahani, M. (2005). The wake behavior behind a heated cylinder in forced and mixed convection regimes. In *ASME 2005 Summer Heat Transfer Conference collocated with the ASME 2005 Pacific Rim Technical Conference and Exhibition on Integration and Packaging of MEMS, NEMS, and Electronic Systems*, pages 745–756. American Society of Mechanical Engineers.

Hu, H. and Koochesfahani, M. (2011). Thermal effects on the wake of a heated circular cylinder operating in mixed convection regime. *Journal of Fluid Mechanics*, 685:235–270.

Hughes, T. J. (2012). *The finite element method: linear static and dynamic finite element analysis*. Courier Dover Publications.

Incropera, F. P. and DeWitt, D. P. (2002). *Fundamentals of heat and mass transfer*. John Wiley & Sons.

Jacqmin, D. (1999). Calculation of two-phase navier–stokes flows using phase-field modeling. *Journal of Computational Physics*, 155(1):96–127.

Jaeger, R., Ren, J., Xie, Y., Sundararajan, S., Olsen, M., and Ganapathysubramanian, B. (2012). Nanoscale surface roughness affects low reynolds number flow: Experiments and modeling. *Applied Physics Letters*, 101(18):184102–184102.

Kannan, R. and Sivakumar, D. (2008). Drop impact process on a hydrophobic grooved surface. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 317(1):694–704.

Karniadakis, G. E. and Triantafyllou, G. S. (1989). Frequency selection and asymptotic states in laminar wakes. *Journal of Fluid Mechanics*, 199:441–469.

Kieft, R. N., Rindt, C., Van Steenhoven, A., and Van Heijst, G. (2003). On the wake structure behind a heated horizontal cylinder in cross-flow. *Journal of Fluid Mechanics*, 486:189–211.

Kim, J. (2005). A continuous surface tension force formulation for diffuse-interface models. *Journal of Computational Physics*, 204(2):784–804.

Krechetnikov, R. and Homsy, G. M. (2009). Crown-forming instability phenomena in the drop splash problem. *Journal of Colloid and Interface Science*, 331(2):555–559.

Lecordier, J.-C., Browne, L., Le Masson, S., Dumouchel, F., and Paranthoen, P. (2000). Control of vortex shedding by thermal effect at low reynolds numbers. *Experimental Thermal and Fluid Science*, 21(4):227–237.

Li, X., Lowengrub, J., Rätz, A., and Voigt, A. (2009). Solving pdes in complex geometries: a diffuse domain approach. *Communications in mathematical sciences*, 7(1):81.

Liu, Y. and Li, G. (2012). A new method for producing lotus effect on a biomimetic shark skin. *Journal of colloid and interface science*.

Lowengrub, J. and Truskinovsky, L. (1998). Quasi–incompressible cahn–hilliard fluids and topological transitions. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1978):2617–2654.

Lunkad, S. F., Buwa, V. V., and Nigam, K. (2007). Numerical simulations of drop impact and spreading on horizontal and inclined surfaces. *Chemical Engineering Science*, 62(24):7214–7224.

Ma, X. and Zabaras, N. (2009). An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113.

Michaux-Leblond, N. and Belorgey, M. (1997). Near-wake behavior of a heated circular cylinder: viscosity-buoyancy duality. *Experimental thermal and fluid science*, 15(2):91–100.

Minjeaud, S. (2012). An unconditionally stable uncoupled scheme for a triphasic cahn–hilliard/navier–stokes model. *Numerical Methods for Partial Differential Equations*.

Mittal, S. (2001). Computation of three-dimensional flows past circular cylinder of low aspect ratio. *Physics of Fluids (1994-present)*, 13(1):177–191.

Morton, K. W. and Mayers, D. F. (2005). *Numerical solution of partial differential equations: an introduction*. Cambridge university press.

Nobile, F., Tempone, R., and Webster, C. G. (2008). A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345.

Noto, K. and Matsumoto, M. (1991). Generation and suppression of the karman vortex street upon controlling surface temperature of a cylinder. *Proceedings of numerical methods laminar and turbulent flows*, 7:671–678.

Oden, J. (1973). The finite element method in fluid mechanics. *Finite element methods in continuum mechanics*, pages 151–186.

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49.

Parizi, H. B., Rosenzweig, L., Mostaghimi, J., Chandra, S., Coyle, T., Salimi, H., Pershin, L., McDonald, A., and Moreau, C. (2007). Numerical simulation of droplet impact on patterned surfaces. *Journal of Thermal Spray Technology*, 16(5-6):713–721.

Prosperetti, A. (1981). Motion of two superposed viscous fluids. *Physics of fluids*, 24(7):1217.

Roshko, A. (1954). On the development of turbulent wakes from vortex streets.

Rübenkönig, O. (2006). The finite difference method (fdm)-an introduction. *Albert Ludwigs University of Freiburg*, page 139.

Schlichting, R. D. and Schneider, F. B. (1983). Fail-stop processors: an approach to designing fault-tolerant computing systems. *ACM Transactions on Computer Systems (TOCS)*, 1(3):222–238.

Shepherd, J. et al. (2000). Cubit mesh generation toolkit. *SAND2000-2647, Sandia National Laboratories, Albuquerque, New Mexico*.

Šikalo, Š., Tropea, C., and Ganić, E. (2005). Dynamic wetting angle of a spreading droplet. *Experimental thermal and fluid science*, 29(7):795–802.

Slaouti, A. and Gerrard, J. (1981). An experimental investigation of the end effects on the wake of a circular cylinder towed through water at low reynolds numbers. *Journal of Fluid Mechanics*, 112:297–314.

Smolyak, S. A. (1963). Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Dokl. Akad. Nauk SSSR*, volume 4, page 111.

Stoecklein, D., Wu, C.-Y., Owsley, K., Xie, Y., Di Carlo, D., and Ganapathysubramanian, B. (2014). Micropillar sequence designs for fundamental inertial flow transformations. *Lab on a Chip*, 14(21):4197–4204.

Tezduyar, T. E., Mittal, S., Ray, S., and Shih, R. (1992). Incompressible flow computations with stabilized bilinear and linear equal-order-interpolation velocity-pressure elements. *Computer Methods in Applied Mechanics and Engineering*, 95(2):221–242.

Tezduyar, T. E. and Sathe, S. (2007). Modelling of fluid–structure interactions with the space–time finite elements: solution techniques. *International Journal for Numerical Methods in Fluids*, 54(6-8):855–900.

The HDF Group (2000-2014). Hierarchical data format version 5.

Tryggvason, G. (1988). Numerical simulations of the rayleigh-taylor instability. *Journal of Computational Physics*, 75(2):253–282.

Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., and Jan, Y.-J. (2001). A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708–759.

Varma, K. B., Hu, H., and Wang, Z. (2007). Numerical investigation of effect of buoyancy on the wake instability of a heated cylinder in contra flow.

Versteeg, H. K. and Malalasekera, W. (2007). *An introduction to computational fluid dynamics: the finite volume method*. Pearson Education.

Von Karman, T. and Rubach, H. (1912). Über den mechanismus des flüssigkeits-und luftwiderstandes. *Phys. Z*, 13(2).

Williamson, C. (1988). The existence of two stages in the transition to three-dimensionality of a cylinder wake. Technical report, DTIC Document.

Williamson, C. (1989). Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low reynolds numbers. *Journal of Fluid Mechanics*, 206:579–627.

Williamson, C. H. (1996). Vortex dynamics in the cylinder wake. *Annual review of fluid mechanics*, 28(1):477–539.

Wodo, O. and Ganapathysubramanian, B. (2011). Computationally efficient solution to the cahnhilliard equation: Adaptive implicit time schemes, mesh sensitivity analysis and the 3d isoperimetric problem. *Journal of Computational Physics*, 230(15):6037 – 6060.

Wodo, O. and Ganapathysubramanian, B. (2012). Modeling morphology evolution during solvent-based fabrication of organic solar cells. *Computational Materials Science*, 55:113–126.

Xiu, D. and Karniadakis, G. E. (2002). The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644.

Xu, L. (2007). Liquid drop splashing on smooth, rough, and textured surfaces. *Physical Review E*, 75(5):056316.

Yarin, A. (2006). Drop impact dynamics: splashing, spreading, receding, bouncing. *Annu. Rev. Fluid Mech.*, 38:159–192.

Yue, P. and Feng, J. (2011). Can diffuse-interface models quantitatively describe moving contact lines? *The European Physical Journal Special Topics*, 197(1):37–46.

Yue, P., Zhou, C., and Feng, J. J. (2010). Sharp-interface limit of the cahn–hilliard model for moving contact lines. *Journal of Fluid Mechanics*, 645:279–294.

Yue, P., Zhou, C., Feng, J. J., Ollivier-Gooch, C. F., and Hu, H. H. (2006). Phase-field simulations of interfacial dynamics in viscoelastic fluids using finite elements with adaptive meshing. *Journal of Computational Physics*, 219(1):47–67.

Zabaras, N. and Ganapathysubramanian, B. (2008). A scalable framework for the solution of stochastic inverse problems using a sparse grid collocation approach. *Journal of Computational Physics*, 227(9):4697–4735.

Zhou, C., Yue, P., Feng, J. J., Ollivier-Gooch, C. F., and Hu, H. H. (2010). 3d phase-field simulations of interfacial dynamics in newtonian and viscoelastic fluids. *Journal of Computational Physics*, 229(2):498–511.