8-29-2012

Interstitial-Scale Modeling of Packed-Bed Reactors

Daniel Parks Combest
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Energy, Environmental, and Chemical Engineering

Interstitial-Scale Modeling of Packed-Bed Reactors

by

Daniel P. Combest

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2012
Saint Louis, Missouri

ABSTRACT OF THE THESIS

Interstitial-Scale Modeling of Packed-Bed Reactors

by

Daniel P. Combest

Doctor of Philosophy in Chemical Engineering

Washington University in St. Louis, 2012

Research Advisor: Palghat A. Ramachandran

Packed-beds are common to adsorption scrubbers, packed bed reactors, and trickle-bed reactors widely used across the petroleum, petrochemical, and chemical industries [106]. The micro structure of these packed beds is generally very complex and has tremendous influence on heat, mass, and momentum transport phenomena on the micro and macro length scales within the bed. On a reactor scale, bed geometry strongly influences overall pressure drop, residence time distribution, and conversion of species through domain-fluid interactions. On the interstitial scale, particle boundary layer formation, fluid to particle mass transfer, and local mixing are controlled by turbulence and dissipation existing around packed particles. In the present research, a CFD model is developed using OpenFOAM (www.openfoam.org) to directly resolve momentum and scalar transport in both laminar and turbulent flow-fields, where the interstitial velocity field is resolved using the Navier-Stokes equations (i.e. no pseudo-continuum based assumptions [88]). A discussion detailing the process of generating the complex domain using a Monte-Carlo packing algorithm is provided, along with relevant details required to generate an arbitrary polyhedral mesh describing the packed-bed. Lastly, an algorithm coupling OpenFOAM with a linear system solver using the graphics processing unit (GPU) computing paradigm was developed and will be discussed in detail.

Acknowledgments

It is an honor for me to have been able to work with Dr. Palghat A. Ramachandran so closely during my time in graduate school. His feedback and continued curiosity enabled me to gain a much deeper understanding of the subject of this dissertation. Additionally, I would like to thank my co-advisor Dr. Milorad P. Dudukovic for his effort to empower me to be a better engineer.

I would also like to acknowledge the industrial partners of the Chemical reaction Engineering Laboratory (CREL), especially the partners of the Multiphase Reaction Engineering (MRE) fund for the financial support throughout my graduate work. Additionally, I would like to thank MAGEEP, the American Chemical Society, and Nvidia for their support through project seed funds, travel awards, and equipment donations. Lastly, I would like to thank the engineering scholars program for the continued financial support during my undergraduate and graduate education in the form of the Gloria & Rubin Feldman and James M. McKelvey Scholarships.

A special thanks goes to the many graduate students, faculty, and undergraduate students that have provided valuable lessons and feedback on my research, writing, and teaching throughout graduate school.

Daniel P. Combest

Washington University in Saint Louis
August 2012

Dedicated to my teachers and former advisors: Jackie Collier, my high school chemistry teacher who inspired (convinced) me to go to college • Drs. Larry Blair, Lee Roecker, Jay Baltisberger, Paul Smithson, Matthew Saderholm, Michael Crescimanno, Amer Lahamer and J.P. Lee for their challenging and thought-provoking courses during my time spent at Berea College • Dr. Dennis Jacobs and group for teaching me about the beauty in experimentation • Dr. Dan Gezelter and group (Pat C., Chris F., Matt M., Megan S., and Charles V.) for giving me my first real experience in theoretical research and scientific computing. Without your group, I would not have thought about graduate school. • Drs. Gero Decher, Gregory Schneider, Albert Izquierdo, and Shoko Ono from my time spent at Institute Charles Sadron in Strasbourg France, for their patience in teaching me about polymer chemistry • Drs. James McKelvey, John L. Kardos and Professors Bob Heider and James Ballard, for their courses, advising, quotable phrases, many hours of homework, and constant reminders of the importance of the fundamentals.

Dedicated to my friends and colleagues: The Peace Corps, for teaching me how to dedicate my life to the world. For my fellow Americans in Malawi (Noah M., Noah Mc., Will, Sarah, Justin, Amanda, Jennifer, Amy, Zacki, Shane and Ron), thank you for the experience of a lifetime. • My labmates from CREL (Mehmet, Yujian, Tim, Vesna, Sean, Radmila, Mohamed, Arnaud, Zeljko, Bia, Nayak, Rajneesh, Ahmed, Onkar, Tim L., Melissa (honorary), and Evgeniy) for your friendship and conversations about science and life • Corey S., Chad G., Jason M., Edd H., and Mark W. for the time away from the dorm, lab, and classroom that kept me sane.

Dedicated to my family: Elisabeth, for your continued encouragement and love throughout graduate school and after • My Grandfather, for teaching me that nothing in life worth doing comes easy and that hard work will pay off if you pour your heart into achieving your goals; my Father, for teaching me the concept of delayed gratification and planning for the future; my Brother, for teaching me that the only way to get to Carnegie Hall is to Practice Practice Practice!; and my Mother, for inspiring me to be curious about the world and to be passionate about life

The list goes on ...

Contents

Abstract . . . . **ii**

Acknowledgments . . . . **iv**

List of Tables . . . . **ix**

List of Figures . . . . **x**

1 Introduction **1**
- 1.1 Background and Motivation 1
  - 1.1.1 Objectives of this Work 5
- 1.2 Previous and Related Studies 6
  - 1.2.1 Random Packing of Cylinders 6
  - 1.2.2 Interstitial-Scale Modeling of Packed Bed Reactors 9
  - 1.2.3 Sparse Linear System Solvers in CUDA 13
- 1.3 Present Contributions 14
- 1.4 Thesis Outline 16

2 A Brief Background in Computational Fluid Dynamics **18**
- 2.1 Introduction 18
- 2.2 The Process of Using Computational Fluid Dynamics 19
- 2.3 Domain and Equation Discretization 21
  - 2.3.1 An Introduction to Domain Discretization 21
  - 2.3.2 Equation Discretization: The Finite Volume Method 24
- 2.4 Closure 33

3 Constructing Randomly Packed Beds of Cylindrical Particles . . . **35**
- 3.1 Introduction 35
- 3.2 Geometry Generation Using a Monte-Carlo Packing Algorithm 36
  - 3.2.1 Packing Methodology 36
  - 3.2.2 Results and Discussion 44
- 3.3 Strategy for Mesh Generation 52
  - 3.3.1 Generating the Underlying Particle Geometry 53
  - 3.3.2 Determining the Initial Face Mesh 55
  - 3.3.3 Generating the Tetrahedral Volume Mesh 56
  - 3.3.4 Polyhedral Mesh Conversion 58
  - 3.3.5 Parallel Mesh Decomposition 59

3.3.6 Results and Discussion 59
3.4 Closure 62

4 Interstitial-Scale Momentum Transport Modeling 65
4.1 Introduction 65
4.2 Momentum Transport Modeling 66
4.2.1 Turbulent Momentum Transport Theory 67
4.2.2 Momentum Modeling Methodology 71
4.3 Results and Discussion 80
4.3.1 Pressure-Drop Calculations 80
4.3.2 Three-Dimensional Data 82
4.3.3 Distributions in the Radial Direction 88
4.3.4 Perpendicular Profiles 92
4.4 Closure 95

**5 Interstitial-Scale Scalar
Transport Modeling 98**
5.1 Introduction 98
5.2 Scalar Transport Modeling 99
5.2.1 Scalars and Turbulent Mixing 99
5.2.2 Turbulent Scalar Transport Theory 101
5.2.3 Scalar Transport Modeling Methodology 104
5.3 Results and Discussion 105
5.3.1 Transient Passive Scalar Transport 106
5.3.2 F and E Curve Analysis 109
5.3.3 Effect of Scalar-Flux Model 112
5.4 Closure 113

**6 Implementing Sparse Linear System Solvers Based on CUDA in
OpenFOAM 116**
6.1 Introduction 116
6.1.1 The Graphics Processing Unit: A Shift in Computing Paradigm 117
6.2 Theory and Implementation 119
6.2.1 Implemented Algebraic Operations 121
6.2.2 Parallel Preconditioned Conjugate Gradient Method with Normalized Residual 125
6.2.3 Computational Workflow of Cufflink Solvers 127
6.3 Results and Discussion 129
6.3.1 Steady-State Scalar Transport 129
6.3.2 Current Developments 131
6.4 Closure 131

7 Summary of Contributions and Future Work 133

7.1 Constructing Packed Beds of Cylindrical Particles 134
7.2 Interstitial-Scale Momentum Transport Modeling 136
7.3 Interstitial-Scale Scalar Transport Modeling 137
7.4 Implementing Sparse Linear System Solvers Based on CUDA in Open-FOAM 139
7.5 Overall Comments 140

Appendix A Miscellaneous Mathematical Matter 141
A.1 General Form of Gauss's Flux Theorem 141
A.2 Owner-Neighbor Relationship in FVM 142
A.3 Reynolds Averaging 142

Appendix B Programming Related Matter 143
B.1 The Cufflink Library 143
B.1.1 Multi-GPU BLAS Level 1 Operations 143
B.2 Courant Number Definitions 146

Appendix C Additional Background in Transport Phenomena Concepts 147
C.1 The k-ϵ Model 147
C.2 The Turbulent Schmidt and Prandtl Numbers 149
C.2.1 Churchill's Reinterpretation and A Look to the Future 152

References 155

List of Tables

1.1 RANS Turbulence Models in Interstitial-Scale Packed-Bed Modeling . 13

3.1 Comparison of Relative Mesh Cell Quality to cell Skewness 57
3.2 Mesh summary of Scotch decomposed mesh 62

List of Figures

1.1 Modeling strategies to capture momentum, based on knowledge of packed-bed structure. Schematic of a realistic packed-bed system of stationary packed catalyst particles (center); a. Bulk Porosity values enable low-dimensional models to based on an *a priori* velocity field to give global information; b. Radial Porosity Distribution functions (Equation 1.1) provide a more detailed bed description for determination of the velocity profile on the reactor-scale; and c. Complete Descriptions of Particles allow for extremely detailed modeling that leverage a fluid mechanical approach, capturing transport phenomena over small portions of the packed-bed at the particle-scale 2

1.2 Visual examples of the the results by Caulkin et. al. [21] as obtained by the program DigiPac for: (a) Pall ring packing, (b) packing of 4-mm alumina pellets, (c) Hama beads packing 8

1.3 An example of the results using the overlap determination algorithm by Blaak [18], discussed in more detail in Chapter 3. 9

1.4 Mesh structure of a unit-cell simulation used by Gunjal et. al. [54] for (a) simple-cubic; (b) and (d) Rhombohedral; and (c) face-centered cubic structures. 10

2.1 polyhedral control volume from [61] 22

2.2 Two and three dimensional cells shapes 23

2.3 Survey of Mesh Generation Algorithms from Owen [96] 23

3.1 Particle representation (a) Definition of local (x',y', and z') and global position (x, y, and z) in relation to position vector V_{pos} with an orientation vector U; (b) Definition of the orientation vector projections $U_{py'x'}, U_{px'z'}$ and $U_{pz'y'}$ around local axes with respect to the angles θ, ϕ, and τ using the right hand rule. 37

3.2 Relation of local (x', y', and z') and global (x,y, and z) coordinate system using a particle position vector V_{pos} for (a) Simple trilobed particle with member orientation vectors U_1, U_2 and U_3; (b) Simple quadlobed particle with member orientation vectors U_1, U_2, U_3 and U_4. 38

3.3 The intersection ($\cap$) of a cylindrical slice S_r and packed bed domain D yields an object whose volume equals the volume of the void space at distance r from the bed center, through an angle from 0 to 2π around the bed, over the entire height of the bed. 43

3.4 Bulk porosity convergence history and cycle time data for 1000 packed cylinders packed into a tube container for 250 packing cycles using proposed algorithm. 45
3.5 Bulk porosity convergence history and cycle time data with 95 CI bands based on 5 simulations with the same initial particle and container shape, for 1000 packed cylinders packed into a tube container for 250 packing cycles using proposed algorithm. 47
3.6 Radial porosity distribution evolution starting at an initial condition (step 0) through 245 packing cycles (step 245). 49
3.7 Comparison between radial porosity distributions produced by the algorithm and experimental results by Roblee et al. [110] 51
3.8 Packed bed of 500 trilobe particles with bulk porosity of 0.75 (a) Side view of three dimensional geometric representation of packed bed of trilobe particles constructed in GAMBIT meshing software; (b) Radial porosity distribution of the trilobe particle bed. 52
3.9 Side view of three dimensional domain of 1000 packed cylinders with bulk porosity of 0.65 constructed using GAMBIT meshing software. . 54
3.10 Representative domain of interstitial spaces that has been meshed and had a simulation performed. 54
3.11 Example of a reduced domain prior to face and volume meshing. . . . 55
3.12 Example of a triangular element that is (a) maintains Delaunay criterion; and (b) violates Delaunay criterion. 57
3.13 Typical parallel processor distribution on a packed-bed mesh 59
3.14 Example of the surface mesh on a particle (a) full triangular faces prior to arbitrary polyhedral conversion; and (b) polyhedral faces that smoothed out remaining grain boundaries in created triangular mesh. 61

4.1 Random point velocity fluctuations for (a) steady-state turbulent flow and (b) transient turbulent flow 68
4.2 Typical pressure drop data from the simulations that is fitted to Ergun's equation (Equation 4.25) using the fv^3 version of the Spalart-Allmaras turbulent closure model 82
4.3 Presentation of three-dimensional data along the (a) (1,-1,0) and (b) (0,0,1) plane through the bed using Miller index notation. 83
4.4 Velocity magnitude slices along (0,0,1) plane through the packed bed for (a) laminar flow at Re$_p$=10; (b) laminar flow at Re$_p$=100; (c) fully turbulent flow at Re$_p$=1500; and (d) fully turbulent flow at Re$_p$=2500 84
4.5 Velocity magnitude slices along (1,-1,0) plane through the packed bed for (a) laminar flow at Re$_p$=10; (b) laminar flow at Re$_p$=100; (c) fully turbulent flow at Re$_p$=1500; and (d) fully turbulent flow at Re$_p$=2500. The entrance and exit regions of the beds have been removed for improved visualization. 85

4.6 Comparison of velocity magnitude (top row) and vorticity magnitude (bottom row) for laminar (left column) and turbulent (right column) flows around a particular particle in the bed. 86
4.7 Surface plots of turbulent kinetic energy (left) and turbulent kinetic energy dissipation (right) near a particle surface. 86
4.8 Surface and contour plot of Q-criterion used to visualize the presence of vortexes around the packed particles. 87
4.9 Dimensionless (a) radial velocity component $U_r/\bar{U}_r$, (b) azumuthal velocity component $U_\theta/\bar{U}_\theta$, (c) axial velocity component $U_z/\bar{U}_z$, and (d) velocity magnitude $||\mathbf{U}||/||\bar{\mathbf{U}}||$ vs. dimensionless radial distance from the center (r/$d_{particle}$) for laminar and turbulent flows. 89
4.10 Dimensionless (a) radial velocity component $U_r/\bar{U}_r$, (b) azumuthal velocity component $U_\theta/\bar{U}_\theta$, (c) axial velocity component $U_z/\bar{U}_z$, and (d) velocity magnitude $||\mathbf{U}||/||\bar{\mathbf{U}}||$ vs. dimensionless radial distance from the center (r/$d_{particle}$) for laminar and turbulent flows. 91
4.11 Dimensionless velocity magnitude scaled with (a) the average velocity of the lowest laminar Reynolds number of Re$_p$ = 10 and (b) the average velocity at each Reynolds number, related to dimensionless perpendicular distance from the nearest wall (y) scaled with particle diameter. 93
4.12 Dimensionless (a) perpendicular turbulent kinetic energy profiles scaled with the lowest turbulent velocity magnitude and (b) perpendicular turbulent kinetic energy dissipation rate scaled with the lowest turbulent ϵ. 94

5.1 Laminar ($Re_p = 10$, $Sc = 0.79$, $C0 = 0.8$, and $Bo = 7.9$) step-tracer surface plot through the (1,-1,0 plane) cutting plain with $\bar{t} = 26.8$ [sec] and $\sigma^2 = 78.01$ [sec^2]. 107
5.2 Turbulent ($Re_p = 1500$, $Sc = 0.79$, $Sc_t = 0.7$, $Co = 0.8$, and $Bo = 1188$) step-tracer surface plot through the (1,-1,0 plane) cutting plain with $\bar{t} = 5.25 \cdot 10^{-2}$ [sec] and $\sigma^2 = 4.74 \cdot 10^{-5}$ [sec^2]. 108
5.3 Presentation of (a) an age-distribution curve for a laminar flow with $Re_p = 10$ and (b) a comparison of dimensionless age-distribution (E_θ) versus dimensionless time ($\bar{t}$) between a laminar flow and fully turbulent step-tracer. 110
5.4 E curves for flows with $Re_p = 1500$ comparing (a) the effect of turbulent Schmidt and (b) the effect of Courant number. 112

6.1 Intel CPU vs. Nvidia GPU [32] (a) GFlops per second performance comparison for single and double precision (b) Memory bandwidth comparison 118
6.2 Packed-Bed mesh decomposition using Scotch 123

6.3 Simple domain decomposition across six nodes, each containing processor interfaces between sub-meshes 123
6.4 A global representation of our decomposed $A\mathbf{x} = \mathbf{b}$ system with coefficient (A_{ii}) and interface (A_{ij}) matrices for each sub-mesh 124
6.5 Data flow diagram outlining the movement and conversion of data from OpenFOAM to the Cufflink solvers. The process is broken up into three portions where (I.) takes place in OpenFOAM C++ code, (II.) leverages Thrust and CUSP to convert and move the information to the GPU, and (III.) the system is solved on the GPU using Cufflink with the solution transferred back to OpenFOAM. 128
6.6 Speed-up comparison of geometric algebraic multigrid solver in OpenFOAM over 6 processors (GAMG6) with the several Cufflink solvers using one and two GPUs, where CG = conjugate gradient, DPCG = diagonal preconditioned conjugate gradient, SmAPCG = smoothed aggregation multigrid preconditioned conjugate gradient, _Parallel indicates two GPUs, and cufflink indicates a GPU solver. 130

Chapter 1

Introduction

1.1 Background and Motivation

Packed beds are described as a collection of closely placed solid particles in a container in which a flowing phase is contacted with a stationary solid phase. In the chemical industry, packed beds are most commonly used in absorption scrubbers, packed bed reactors, and trickle bed reactors. Absorption scrubbers utilize the high surface area of the non-porous particles in the packed bed to increase the rate of species mass transfer between different phases present in the unit. Packed-bed reactors (PBR) are used to carry out single phase catalyzed reactions such as methane steam reforming, hydrocracking, and partial oxidation, often using a porous heterogeneous catalyst. Similarly, trickle-bed reactors (TBR) are used to combine mass transfer of species between a thin liquid film and gas phase with a reaction using a heterogeneous catalyst [106]. The absorption scrubber, PBR, and TBR are an integral part of the chemical industry that rely on the intricate micro structure within packed beds.

The structure of packing plays an important role, affecting many physical phenomena including dispersion, pressure-drop, interstitial velocity, and local boundary layer formation over particle surfaces, is often difficult to characterized. Due to the intricate of arrangements of the packed particles, the method by which bed structure can be described dictates the method used to model momentum transport in a PBR or TBR, as well as the information gained from these models. Shown pictorially in Figure 1.1, momentum modeling strategies can be broken down into three categories based on the description of bed structure:

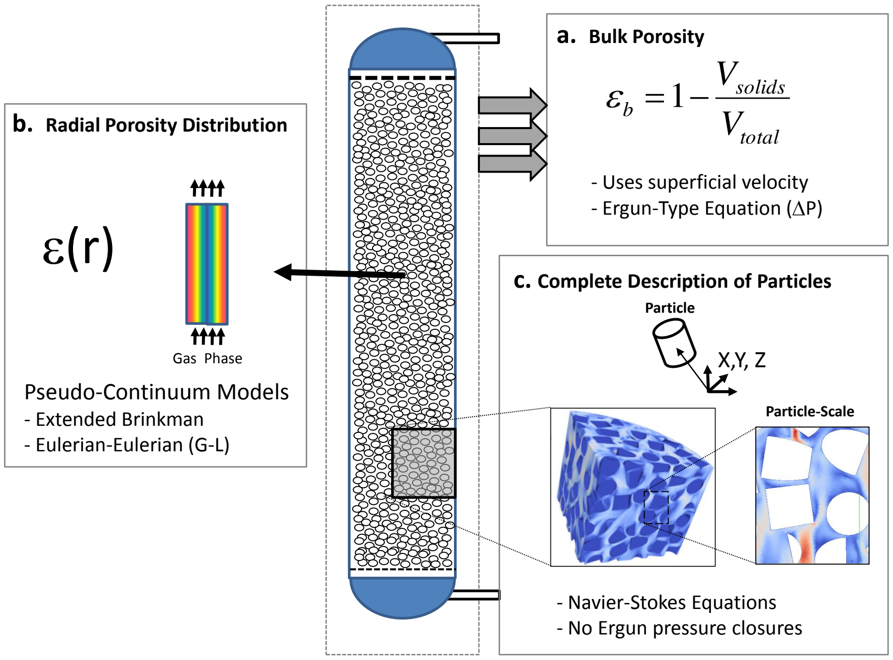

a. Bulk Porosity
$\varepsilon_b = 1 - \frac{V_{solids}}{V_{total}}$
- Uses superficial velocity
- Ergun-Type Equation (ΔP)
b. Radial Porosity Distribution
ε(r)
Gas Phase
Pseudo-Continuum Models
- Extended Brinkman
- Eulerian-Eulerian (G-L)
c. Complete Description of Particles
Particle
X,Y, Z
Particle-Scale
- Navier-Stokes Equations
- No Ergun pressure closures

Figure 1.1: Modeling strategies to capture momentum, based on knowledge of packed-bed structure. Schematic of a realistic packed-bed system of stationary packed catalyst particles (center); a. Bulk Porosity values enable low-dimensional models to based on an *a priori* velocity field to give global information; b. Radial Porosity Distribution functions (Equation 1.1) provide a more detailed bed description for determination of the velocity profile on the reactor-scale; and c. Complete Descriptions of Particles allow for extremely detailed modeling that leverage a fluid mechanical approach, capturing transport phenomena over small portions of the packed-bed at the particle-scale

a. **Bulk Porosity:** Low-dimensional models based on bulk porosity (ϵ_b) must also lump complex velocity flow fields into a superficial fluid velocity ($G_0 = \dot{V}/A_{tube}$) by assuming plug-flow behavior. Although it is seen experimentally that there is a local flow maxima near the tube wall due to the higher local bed porosity [115], the velocity profile in packed-beds is generally modeled as a plug-flow [49]. Though these models are empirical in nature, they can provide overall estimates of pressure drop for power consumption estimations. The benefit is that low order models have been thoroughly tested over the last century and is widely used by industry and academia. The drawback is that the description of complex flow and detailed physical phenomena are lumped together in relations that are fitted with empirical parameters (i.e. Ergun's constants).

b. **Radial Porosity Distribution:** Higher-dimensional models replace a bulk porosity with a more descriptive representation of the bed structure in the form of a porosity distribution function. The most common is the radial porosity distribution for uniformly sized spherical particles from Mueller [90], with dependency on radial position r, reactor diameter D, and particle diameter d, such that

$$\epsilon(r) = \epsilon_b + (1 - \epsilon_b) J_0(ar^*) e^{-br^*}. \tag{1.1}$$

where

$$\begin{aligned}
a &= 8.243 - \frac{12.98}{(D/d + 3.156)}, && \text{for } 2.61 \leq D/d \leq 13.0, \\
a &= 7.383 - \frac{2.932}{(D/d - 9.864)}, && \text{for } 13.0 < D/d, \\
b &= 0.304 - \frac{0.724}{D/d}, && \\
r^* &= r/d, && \text{for } 0 \leq r/d, \\
&\text{and} && \\
\epsilon_b &= 0.379 + \frac{0.078}{(D/d - 1.80)}. &&
\end{aligned}$$

In general, the class of single and multi-phase reactor models that leverage porosity distributions to characterize the solid packing are called pseudo-continuum models (PCM) and have been widely used throughout the literature [79, 88, 76, 100, 64, 52, 53, 107]. The main assumption of PCMs is that all of the phases present in the system can inter-penetrate and exchange momentum via additional lumped source terms in a transport equation [88]. For single phase flows, Vortmeyer and Schuster [130] extended the Brinkman equation to compute velocity profiles ($G(r)$) such that

$$\frac{\partial p}{\partial z} = -E_1 \frac{(1-\epsilon(r))^2}{\epsilon(r)^3} \frac{\eta_{eff}}{d_p^2} G(r) - E_2 \frac{(1-\epsilon(r))}{\epsilon(r)^3} \frac{\rho}{d_p^2} G(r)^2 - \frac{\eta_{eff}}{r} \frac{\partial}{\partial r}(r \frac{\partial G(r)}{\partial r}). \tag{1.2}$$

At the center of the tube is a finite velocity, while at the wall is a no-slip velocity condition [49]. However η_{eff} in Equation 1.2 requires iterative calibration to match model results to measured laboratory flow rates [49]. One of the most dramatic examples of the impact of such models was from Lerou and Froment [79], in which the researchers showed that a simple two dimensional models for heat and mass transport yielded significantly different results when constant and porosity depended velocity profiles were used. Specifically, the authors showed that for the exothermic oxidation of orthoxylene to pthalic anhydride,a non-uniform velocity profile (dependent on porosity) predicted reactor runaway while the constant velocity profile did not. A testament that the classical assumption of plug-flow can estimate a drastic difference in reactor performance.

For multiphase flows, computational fluid dynamics (CFD) models are based on an Eulerian-Eulerian (EE) approach which additionally incorporates the volume fraction of a gas or liquid phase and detailed inter-phase momentum exchange terms[1] for fluid-fluid and fluid-packing interactions first introduced by Attou, A. and Ferschneider [7]. In general, the EE approach behaves agreeably but can inadequately predict complex physical behavior such as hysteresis [75]. Overall, pseudo-continuum models provide a more detailed glimpse into superficial velocity profiles within packed-bed systems, ultimately providing better information to improve reactor design. To a detriment, PCMs are rooted in empiricism, but are considered hybrid models meant to bridge the fluid mechanics gap with traditional chemical engineering reactor-modeling whilst being computationally tractable.

c. **Complete Description of Particles:** By developing a complete description of the packed-bed, the location, orientation, and dimensions of each particle face is know *a priori*. With this level of geometric detail, a methodology based on fluid mechanics and computational fluid dynamics can be leveraged to model the complex fluid flow within the packed-bed to resolve momentum, heat and mass transport phenomena. Currently, obtaining a full description of a industrial-scale packed-bed reactor via tomographic methods is not possible. Obtaining this level of detail for pure fluid mechanical modeling is also impractical due to the amount of computing power required. What has been shown to be practical (Section 1.2), is to model a small section of the packed bed that is representative of the whole bed as shown in Figure 1.1. The particles captured in this representative volume are composed of either structured groups of particles (e.g.

[1]the equation is left out for brevity

face-centered, body-centered, or simple cubic), pseudo-random particles, or artificially generated random domains of particles. Though considered more precise from a first-principles perspective, wide acceptance of these modeling methods on this length scale in the chemical engineering domain have yet to take hold. In addition, models representing fully turbulent non-isothermal reacting flow with gas-solid heat and mass transfer quickly become computationally taxing. Strategies and assumptions to reduce such complications are given in later chapters.

The more detailed the characterization and numerical representation of the packed bed structure the more detailed the reactor model and type of results that can be given by the resulting simulation. Having complete information of the structure of the packed-bed allows for the determination of both the bulk porosity (ϵ_b) and radial porosity distributions (Equation 1.1) directly, subsequently allowing for the low-dimensional and pseudo-continuum models to be fully explored (Chapter 3). This fact could in essence allow for improved pseudo-continuum models in much the same way that direct numerical simulation (DNS) enables more precise large-eddy simulation (LES) and Reynolds-averages Navier-Stokes (RANS) turbulence models through better estimation of model parameters. The benefits and drawbacks of the full fluid mechanical approach to particle-scale modeling is discussed throughout this thesis, and represent a leap in bridging the fluid mechanics gap in chemical engineering.

1.1.1 Objectives of this Work

The overall objective of this work is to investigate heat, mass, and momentum transport on the length-scale of the particles within a randomly packed beds of particles and decipher an appropriate method to generate the random domain and capture various physical phenomena. The principle tasks that will be addressed in this thesis are to:

1. Develop a packing algorithm to randomly place cylindrical particles in a tubular domain while knowing the exact location, orientation, and dimensions of each particle.

2. Develop a strategy to generate computational meshes of these randomly packed particles that minimizes numerical errors.
3. Using computational fluid dynamics, simulate particle-scale momentum transport using the Navier-Stokes equations (laminar flow) and Reynolds-averaged Navier-Stokes methodology (turbulent flow) in OpenFOAM.
4. Using computational fluid dynamics, simulate mass transport in laminar and turbulent flow fields to investigate the sensitivity of turbulent mixing in the system to the choice of turbulent scalar-flux approximation methods.
5. Develop heterogeneous computing methodologies using graphics processing units to accelerate simulations and decrease overall simulation time.

All of these tasks will be completed through three overall phases that include developing a domain generation algorithm for random packing of cylinders, interstitial-scale modeling of packed beds, and developing sparse linear system solvers in CUDA. Previous studies related to each of these phases are discussed in the next section.

1.2 Previous and Related Studies

1.2.1 Random Packing of Cylinders

The development of a method to efficiently and randomly place objects in a container, while minimizing total volume in an optimal manner, has been the subject of much attention in mathematics, physics, chemistry, and engineering [62]. A full evaluation of the prior art in packing algorithms is beyond the scope of this work. However, a discussion of packing domain generation efforts towards packed-bed simulations is provided herein.

For many studies in interstitial-scale simulations of packed-beds, ordered structures of spheres were used to approximate the packed-bed, drastically simplifying the domain construction and meshing process[84, 51, 54, 88, 85, 86]. For more complex particles, Dixon et. al. began using groups of cylindrical particles in low particle-to-tube ratios [88]. Again, the method by which the domain of cylinders was created was simply selecting important structures from those seen in an experimental packing study and manually creating a representative arrangement of particles so that a periodic computational domain could be achieved [92]. The resulting structure is only pseudo-random but extremely well suited for simulation. However, the only randomly packed bed of particles to have fluid simulation to date have been created using the discrete element method (DEM), with fluid modeling being performed using a Lattice Bolztmann Method (LBM) [22]. Though DEM is now an attractive choice for packing particles, at the beginning of this project the methods for efficiently converting DEM results to a 3D computational mesh was difficult.

In the discrete element method, an arbitrarily shaped particle (e.g. cylinder, Pall ring, etc.) is represented as a collection of tiny composition spheres or three-dimensional pixels (Voxels) [91]. The resulting overlap determination of the complex particles is merely the assessment of whether the composition particles overlap. Each of the composition particles is subject to time, collision, gravity, adhesion and other force fields, modeled as Lagrangian particles. As the number of compositional particles grows and the simulation timescale lengthens, the simulation quickly becomes computationally intensive. So much so that in recent years, DEM has increasingly gained traction in the computing world for further application of graphics processing units (discussed in Chapter 6) in the rigid-body dynamics simulation [121].

With respect to the packed-bed simulation, the most relevant work is by Caulkin et. al. [21]. Caulkin and colleagues were able to pack arbitrary shaped particles including Pall rings, cylinders, and Hama beads in a tube using the DEM program DigiPac$^{\text{TM}}$(www.structurevision.com) [47]. The results of Caulkin et. al. are very impressive, as they compare well with X-Ray tomography images of packed-beds of Pall Rings, cylinders and Hama beads shown in Figure 1.2. Because the voxels are represented on a regular grid, more natural method of performing fluid simulation is to use LBM [22]. However, recent advances (2011) in image analysis have enabled more complex meshes to be constructed from voxilated images from CT and MRI scans, showing that the finite volume Navier-Stokes method can indeed be used on these domains from images [8].

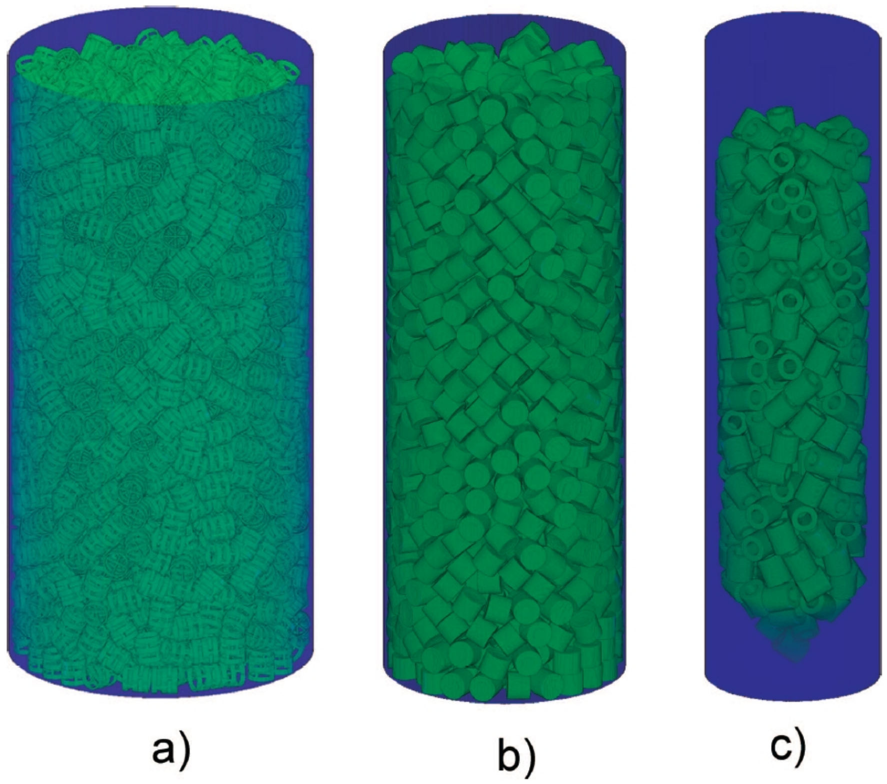


Figure 1.2: Visual examples of the the results by Caulkin et. al. [21] as obtained by the program DigiPac for: (a) Pall ring packing, (b) packing of 4-mm alumina pellets, (c) Hama beads packing

As an alternative to DEM, a method was devised at the beginning of this work, seeking a cylinder-cylinder overlap determination explicitly, so that each face of the particles could be known exactly and meshed. One such method was proposed by Blaak to investigate the formation of cubatic phases in the packing of cylinders [18]. Using the method of particle-particle overlap determination by Blaak, one can know the orientation and location of the faces of the cylindrical particles and generate a computational mesh relatively easily. As we shall see in Chapter 3, the algorithm by Blaak provided a more than adequate method for generating a packed-bed of particles as shown in Figure 1.3

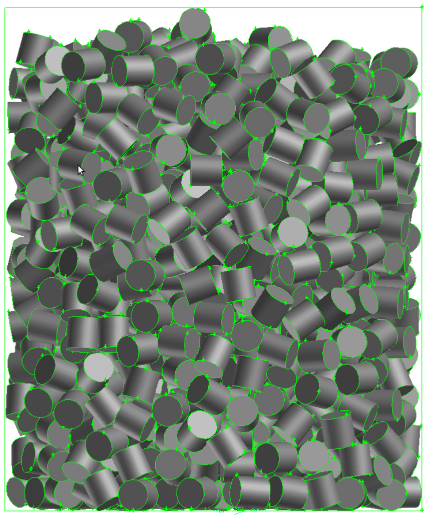


Figure 1.3: An example of the results using the overlap determination algorithm by Blaak [18], discussed in more detail in Chapter 3.

1.2.2 Interstitial-Scale Modeling of Packed Bed Reactors

As introduced in the background and motivation section, interstitial-scale modeling offers the most detailed look into the transport processes occurring in a packed-bed. Specifically, each particle surface is resolved, a detailed mesh approximating the domain is generated, and complete modeling of transport equations for heat, mass, and momentum are performed. A fairly complete review of the literature has been provided by Dixon et. al., navigating the field up until 2006 [88]. More recently, a discussion by Ranade, Chaudhari, and Gunjal provided further insight into the experiential and modeling of single and multiphase flow modeling in packed-beds[108]. The pertinent details of the literature review are left to the interested reader, but sufficient background of the topic is provided here to orient the reader as to the state-of-the art.

Simulations of interstitial phenomena in packed-beds generally leverages the finite volume method (Chapter 2), however some attention has been given to the Lattice Boltzmann Method (LBM), especially from packing algorithms leveraging DEM [23]. The limitation to the LBM is that flows are limited to lower Mach numbers and turbulence is handled through a relaxation time process [43]. For finite volume based

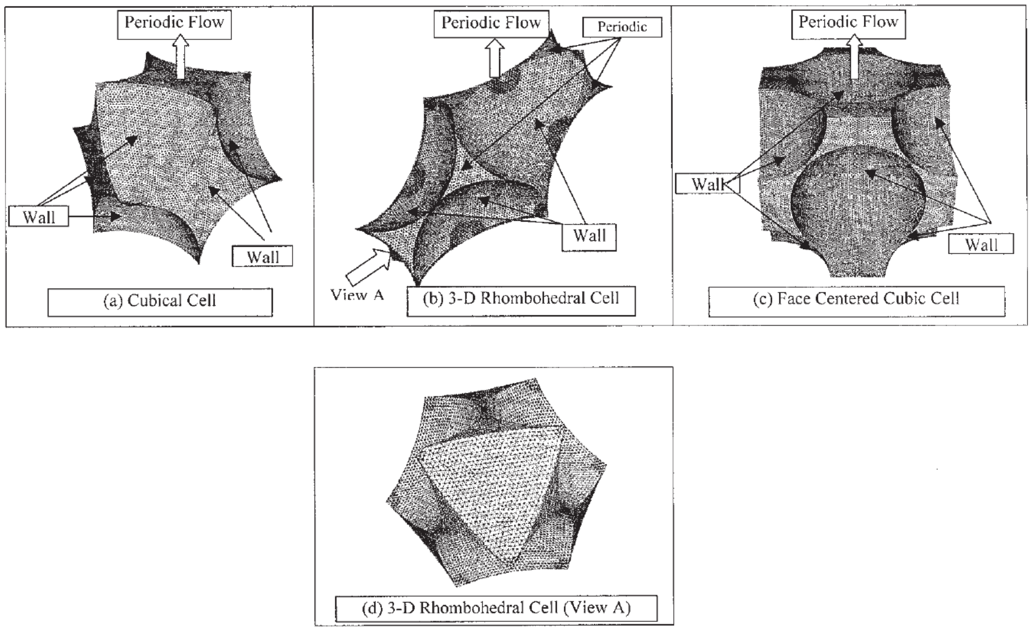


Figure 1.4: Mesh structure of a unit-cell simulation used by Gunjal et. al. [54] for (a) simple-cubic; (b) and (d) Rhombohedral; and (c) face-centered cubic structures.

methods, nearly the entire spectrum of published results in interstitial-scale CFD of packed beds leverages software by ANSYS (http://ansys.com/).

There are two main type of simulation on the interstitial-scale, a unit cell approach (UCA) and a macro cell approach (MCA). In the unit-cell approach, particles are arranged in an orderly fashion to form repeating units similar to crystal structure (face-centered cubic, body centered cubic,etc.)[54]. UCA uses fine meshes to capture near particle transport phenomena and periodic boundary conditions to remove inlet and exit effects, with typical meshes shown in Figure 1.4. Most importantly, the work by Gunjal et. al. sought to further validate the flow field through direct comparison of simulation to experimental results using non-invasive MRI imaging [54, 88]. The second method using the macro cell approach simulates transport phenomena over larger groups of particles that are a representative section of the bed. Macro cell approaches are currently dominant in the literature, and have been the most difficult to validate. Almost the entire collection of the literature has “validated” simulation results with boundary values (i.e. pressure drop, heat flux calculations, etc.), radial profiles of some sort (e.g. velocity, temperature, etc.), and qualitative comparison with trends. Full validation of the flow field and experiment in a one-to-one comparison is still lacking, since we are only recently able to generate meshes of experimental setups directly from MRI images and yet to determine a fully-resolved real-time flow field [8]. For macro cell simulations, complete validation is still an issue and an integral portion of the research on interstitial scale phenomena in packed beds. So much so that validation of flow around a single particle is still a focus of research [41].

Though the research in the field of interstitial-scale transport modeling has been admittedly academic in nature, it has gained increased attention of industry for research related to particle shape optimization. The work by Nijemeisland and collaborators, has focused on low particle to tube ratio systems for optimization of wall heat transfer [92]. In particular they have chosen to optimize the number and diameter of lengthwise holes in cylindrical catalysts along with small alterations to the overall cylindrical shape [123]. Recently they have worked to resolve intraparticle diffusion and reaction within their simulation for nonisothermal flows [125, 124, 40, 14]. Though there is a steady flow of publications continuing to be put out in the literature, there has yet to be a fundamental analysis of particle characteristics that optimize some objective for more general systems. More specifically, a list of questions could be addressed to further understand the link of particle structure and function towards optimizing a particular process in a packed-bed, including:

1. What is the optimal shape for reduced pressure drop in a packed bed?
2. What attribute of the particle causes dead-zones to form behind particles?
3. Can the near particle boundary layer be controlled and is this level of control important?
4. What does the shape of a particle look like to promote maximum heat removal from a particle surface or prevent run-away exothermic reactions?
5. Is there an preferred orientation of a particle to reduce undesired effects?
6. Can we make a particle that packs in a specific manner without much effort (i.e. packs itself in a desired configuration)?
7. What is the most accurate model for capturing the complex flow field and is large eddie simulation (LES) and direct numerical simulation (DNS) an option?

Partial answers to some of these questions can be found from previous studies mentioned earlier, however the complete particle structure to function link has yet to be fully elucidated. Doing so requires further study in packing algorithms, meshing studies, and turbulence modeling in fluid simulations specifically aimed at answering these questions in order to obtain a more fundamental understanding.

Meshing Of Packed Beds

The generation of meshes describing the packed bed structure at the interstitial scale requires special consideration that has been discussed in the literature[88, 108]. The unit cell modeling of Gunjal reiterates the fact that the mesh must be fine to capture hydrodynamics within the interstitial spaces, and in some cases require periodic boundary conditions to eliminate inlet and outlets effects [54]. For larger domains of particles, spaces between particles must be used to reduce the skewness of the mesh near particle-particle contact points, i.e. increase the uniformity of face area of the tetrahedron cells [93]. At contact points, the increased skewness of a mesh creates more dramatic changes in velocity resulting in solution instability at higher Reynolds numbers with and without turbulent closure [93]. In addition skewed cells make it very difficult to converge a solution. Meshes of packed bed structures generally use unstructured tetrahedron meshes to fully describe the intricate domain and may or may not include the internal geometry of the particles [88]. In order to reduce the effects of mesh skew and non-orthogonality, hybrid meshes composed of tetrahedral and hexahedral-prism cells were used to simulate single-particle dynamics, to resolve boundary layers, drag, and heat transfer from a particle [41]. What has not been addressed is the fact that **misalignment with the flow and mesh will always occur with tetrahedral meshes**, increasing the error in the calculation. Misalignment with the flow can be overcome by using unstructured arbitrary polyhedral meshes, a topic covered in later chapters.

Turbulence Modeling

For most of the literature in macro-cell interstitial-scale modeling, Reynolds averaged Navier-Stokes (RANS) methods have been used in order to minimize computational

Table 1.1: RANS Turbulence Models in Interstitial-Scale Packed-Bed Modeling

Author	Source	RANS Turbulence Model
Guardo et. al.	[50]	**Spalart-Allmaras** Standard, RNG, and Realizable $k-\epsilon$ Standard $k-\omega$ (Wilcox)
Gunjal, Ranade, and Chaudhari	[54]	Standard $k-\epsilon$
Dixon et. al.	[84, 94, 41]	standard and RNG $k-\epsilon$ and $k-\omega$
Combest, Ramachandran & Dudukovic	[27, 26]	Spalart-Allmaras fv3 version

efforts. The details of RANS modeling are addressed in Chapter 4, however the common turbulence models used in interstitial-scale modeling are summarized in Table 1.1. A comparison of several turbulent viscosity models was given by Guardo [50], stating that the optimal method was the Spalart-Allmaras model. However there was no specific mention of what version of the model was optimal. There is a wide variety of turbulence models used in the studies, mostly related to $k-\epsilon$, but interestingly the current results of the literature show that there is no real difference in the selection of the turbulence model as long as it is used correctly [92]. This conclusion is addressed in Chapter 4 during the selection of models used for this research project.

1.2.3 Sparse Linear System Solvers in CUDA

There have been several projects with varying levels of success that have linked the GPU to OpenFOAM. The earliest project that was available to the public was the Vratis SpeedIT plugin (`http://speedit.vratis.com/`) in November of 2010, offering a single precision CUDA based code with diagonal preconditioned conjugate gradient method. The SpeedIT plugin offered multi-gpu support only through apparent parallelism of the linear algebraic operations and not through domain decomposition. The second code offered to the public was provided by Symscape (`http://www.symscape.com/gpu-openfoam`) in April of 2011, giving diagonal preconditioned conjugate gradient and bi-conjugate gradient solvers without multi-GPU support. The third code was outlined in a proposal to Nvidia by Combest and Ramachandran [28] and conference presentation [27] in the summer of 2010. This last code is currently a portion of my thesis work and is being released under Gnu Public License Version 3 [1] under the name cufflink.

Additional CUDA Libraries

This work employs CUSP and THRUST, two additional libraries that are based on CUDA C/C++. CUSP is used for sparse linear algebra and graph computations, that contains a collection of BLAS functions and example algorithms for solving sparse linear systems of equations provided by the user [16]. THRUST is a collection of algorithms with an interface resembling the C++ Standard Template Library (STL) [120] that aims to combine the strength and ease of the template programming of the STL and parallelism of the GPU [56]. The basis of CUSP is a mature and highly efficient sparse matrix vector multiplication algorithm [15] that has been optimized for several sparse matrix storage schemes. Similarly, THRUST has received recent attention for sorting one-billion integer keys per second, a parallel algorithm that is faster than several CPU based sorting methods [89]. By leveraging the strength of CUSP and THRUST, this work utilizes highly active and cutting-edge parallel algorithms that are then tied into the OpenFOAM library.

1.3 Present Contributions

Below are the present contributions of this project that will be further outlined and discussed in detail throughout this document:

Random Packing of Cylinders

- Created a Monte-Carlo method for generating random domains of cylindrical-based particles for interstitial-scale simulations.
- Developed a method to create a mirrored mesh so that meshes of higher resolution can be created with limited computational resources.

Particle Scale Modeling of Packed-Bed Reactors

- Provided justification for the use of arbitrary polyhedral meshes, and showed that they are ideally suited for interstitial-scale flow simulations due to the presence of extremely complicated domains.
- Was the first to showed that the low Reynolds number Lam-Bremhorst $k - \epsilon$ model is ideally suited for packed bed modeling due to the additional dissipation terms in the Lam-Bremhorst model, requiring no additional wall-functions.
- Provided insight into the need for improved scalar-flux solvers in RANS based simulations of scalar transport in a turbulent field

Sparse Linear System Solvers in CUDA

- Showed more than 100x serial speedup compared to OpenFOAM solvers for Smoothed Aggregate Algebraic Multigrid Preconditioned Conjugate Gradient solvers for the heat equation.
- Was the **first** to use Multiple-GPUs leveraging course-grained parallelism through domain decomposition method to solve massively parallel CFD simulations
- Provided open source code back to OpenFOAM community in the form of a library tied to both CUSP and THRUST using the CUDA language.

General Contributions

- Developed a framework for modeling interstitial-scale phenomena in packed-beds
- Initiated the use of OpenFOAM at Washington University
- Provided support for new users in OpenFOAM in CREL, WUStL, and at other universities.

1.4 Thesis Outline

With the overall goal of using the OpenFOAM library to model the interstitial-scale transport phenomena in a packed-bed, each chapter will explain in sufficient details the following:

Chapter 2: A Brief Background in Computational Fluid Dynamics
This chapter provides and introduction to computational fluid dynamics (CFD), including the overall workflow and general considerations in the using CFD as an analysis and modeling tool. Both domain and equation discretization will be discussed, with special attention towards the discretization methods used in this research. The reader will be left with a general sense of the complicated process of using CFD.

Chapter 3: Constructing Packed Beds of Cylindrical Particles
This chapter will thoroughly discuss the methods used to generate the randomly packed particles for the CFD simulations. Specifically, the domain generation and subsequent meshing strategy to create meshes of arbitrary polyhedral cells is covered.

Chapter 4: Interstitial-Scale Momentum Transport Modeling
This chapter will contain key information about the process of modeling momentum transport in laminar and turbulent systems, and is the majority of the efforts during this project. Covering pertinent information on the Lam-Bremhorst low Reynolds number turbulence model used in this study, model equations, boundary condition considerations, and results and discussion of flow-field results will be given.

Chapter 5: Interstitial-Scale Scalar Transport Modeling
Building upon the prior chapter, modeling passive scalar transport in laminar and turbulent flow fields is presented. Specific discussion of the treatment of the scalar-flux term ($\langle u'C' \rangle$) is provided, as well as the simulations of step-tracer simulations.

Chapter 6: Implementing Sparse Linear System Solvers Based on CUDA in OpenFOAM
As a method to accelerate the OpenFOAM library, graphics processing units (GPUs) were used to parallelize Krylov subspace linear system solvers. Specifically, a library coupled with OpenFOAM was created, that utilizes both CUSP and THRUST CUDA libraries is covered in detail.

Chapter 7: Summary of Contributions and Future Work
Concluding the thesis, this is a discussion of the contributions made throughout the project with final thoughts as to future work.

Chapter 2

A Brief Background in Computational Fluid Dynamics

2.1 Introduction

Computational Fluid Dynamics (CFD) in its most basic form is the study of partial differential equations related to the description of fluid mechanics and transport phenomena. The study of which involves methods to transform the physical system in question from continuous space to discrete coordinate space. Most importantly, CFD utilizes numerical analysis to develop approximate solutions to the continuous problem.

CFD addresses questions that cannot be solved analytically, and can provide complete information of transport phenomena occurring within a system to enable improved engineering design. CFD can be used to analyze the design of full aircraft [4], evaluate full scale trickle-bed reactor performance [108], improve hip joint design through contact stresses analysis [20], and improve the fundamental understanding of micro-scale phenomena in packed bed reactors [26, 27]. Ultimately it is a tool that is used to provide a glimpse into the difficult to measure length and time scales and scenarios where it is impractical to build many prototypes.

In this chapter, a brief background of computational fluid dynamics (CFD) is given to familiarize the reader with the methods used in later chapters. A discussion of the key steps involved in the workflow of solving a problem using CFD are provided. The chapter continues with the cursory explanation of both space and equation discretization. Beginning with domain discretization, the idea of the computational mesh is presented and mesh generation algorithms are mentioned. The reader should be left with a broad sense of composition of a computational mesh the level of detail that must be considered. Next, in Section 2.3.2, the reader is given a brief introduction to the finite volume method (FVM) so that a further understanding of overall CFD solution procedure may be achieved. Face interpolation schemes that are mentioned in subsequent chapters are presented entirely, leaving the reader with an informed understanding of the importance of local continuity, the paramount characteristic of the FVM. Finally, closure is given to prepare the reader for a discussion of the methods used to model complex turbulent fluid flow in packed bed reactors in Chapter 4. Relevant sources are provided throughout and the reader is encouraged to pursue cited texts for additional depth and clarity.

2.2 The Process of Using Computational Fluid Dynamics

The process of using CFD to arrive at an accurate and physically meaningful answer is an extremely complex task that requires careful consideration of many details [104, 44, 128, 59, 81, 134]. It is extremely beneficial to establish a systematic approach for a specific type of simulation (e.g. aerospace, ship hydrodynamics, turbo machinery, multiphase reactor design, combustion modeling, etc.), and consistently apply these methods in a precise manner. One such set of "best practices" guidelines are published by the European Research Community on Flow, Turbulence, and Combustion (ERCOFTAC) [2]. Editors Casey and Wintergerste offer a thorough and somewhat non-technical discussion on the process of using CFD, and how to judge the efficacy of such simulation work. The editors discuss a "typical" CFD calculation and note that there are several important steps, including:

- Training of CFD users
- Problem definition
- Selection of solution strategy
- Choice of numerical procedure
- Validation of models
- Selection of turbulence models
- Geometry definition
- Boundary condition definition

- Definition of initial guess/condition
- Solution to the numerical equations
- Assessment of errors and accuracy
- Post-processing and visualization
- Analysis and interpretation
- Documentation and archiving of results
- Communication with code developer

For the discussion at hand, the entire process is divided into a simulation cycle of preprocessing, model development, code execution, and post-processing and is briefly discussed below.

1. **Preprocessing:** During preprocessing, the physical phenomena to be modeled is isolated through the choice of model equations and boundary conditions; domain measurements along with the computational mesh is decided upon, constructed, and possibly decomposed for parallel execution; physiochemical properties for the system in question are retrieved from the literature or experimental data; computational requirements are estimated for the unknown system; and specific objectives of data collection are outlined for post-processing. Careful planning at this stage will result in a quicker cycle of simulations.

2. **Model Development:** During model development, the model equations are derived and programmed for code execution. A solution strategy to solve the model equations in an efficient manner is explored. The code must be thoroughly debugged and tested against canonical cases or analytical solutions to provide confidence in the final result on an unknown geometry and fluid system.

3. **Code Execution:** Code execution requires that computational requirements be reaccessed and evaluated for efficiency (i.e. is it more efficient to run our case on 10 nodes or 100 nodes?). Data collection, preliminary error analysis, solution quality assessment, and a solution improvement strategy (e.g. change of discretization method to improve stability) are performed simultaneously and may lead require revisions in previous steps in the simulation cycle.

4. **Post-Processing:** During post processing, the results of the simulation are evaluated, summarized, and put into a form more conducive to sharing and understanding. Conclusions about the results are drawn and may require the simulation cycle to be repeated until errors have been minimized or the model produces data that matches experimental results.

Overall, the process of using computational fluid dynamics to investigate complicated fluid phenomena has been summarized. It is important to glean that the process is iterative and requires many steps in order to achieve a final result. Returning to a previous phase in the simulation cycle requires the remaining steps to be completed again. The amount of time of each simulation cycle is directly related to practitioner experience, choice of model (e.g. DNS, RANS, LES, multiphase flow models, moving mesh models, etc.), computational resources available, mesh quality and composition, equation discretization, and may depend on several other factors not mentioned. Ultimately, if the proper model equations to describe the physical phenomena are chosen correctly; error analysis is performed and errors minimized; and care is taken to ensure validity through proper choice of discretization, the practitioner will increase their chances of producing a successful simulation.

2.3 Domain and Equation Discretization

2.3.1 An Introduction to Domain Discretization

In terms of the finite volume method discussed in section 2.3.2, domain discretization is the practice of approximating continuous computational space with a finite number of control volumes. The accuracy of the final solution, stability of the solution procedure, and detail of physical phenomena that is captured by the model equations is directly proportional to mesh size (i.e. number of cells), quality, and composition (i.e. cell shapes). As a result, the generation of a mesh over the desired physical domain is integral to the solution process.

In OpenFOAM, discretized space is represented as a collection of a finite number of arbitrary polyhedron control volumes (cells) that are described fully using the polyMesh[2] format [81]. As an example of a polyhedron control volume, Figure 2.1 shows a cell with a centroid at point P, a face-area vector $\mathbf{S}_f$ at the area-centroid of face f, and the location of the neighbor cell centroid at point N. Unfortunately,

[2]The polyMesh format is specific to OpenFOAM

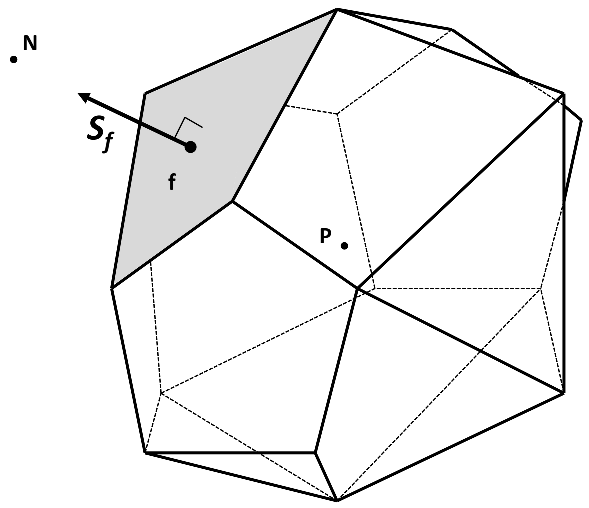


Figure 2.1: polyhedral control volume from [61]

there is no direct method available to produce pure arbitrary polyhedral meshes[3]. In order to produce an arbitrary polyhedral mesh, a mesh consisting of simplified volume elements must be first generated and then eventually converted to an arbitrary polyhedral mesh.

The simplest area elements of a two-dimensional mesh are the triangle and quadrilateral. All surfaces in the discretized domains begin with these two shapes and are the basis of the mesh cells that are created in the space within computational domain. If we draw a point away from the triangular and quadrilateral faces and connect each vertex to this point, we can create the simplest three-dimensional elements [101] – the tetrahedron and pyramid – shown in Figure 2.2. If we extrude the triangle and quadrilateral faces in a single direction, we then create wedges (or prisms) and hexahedron cells. From these basis elements – tetrahedron, pyramid, wedge/prism, and hexahedron – we can create an entire computational mesh or grid of a finite number of non-overlapping control volumes to completely approximate the physical domain.

Using the volume elements presented in Figure 2.2, the size, shape (i.e. level of deformation) and quantity of each element must be determined during the domain space approximation. The choice of mesh generation algorithms is immense, and can be partially represented by Figure 2.3 that has been compiled by Owen [95] and

[3]To the best of my knowledge at the time of writing, this is correct. This was a question that I posed to a PointWise developer at the 6th OpenFOAM Workshop

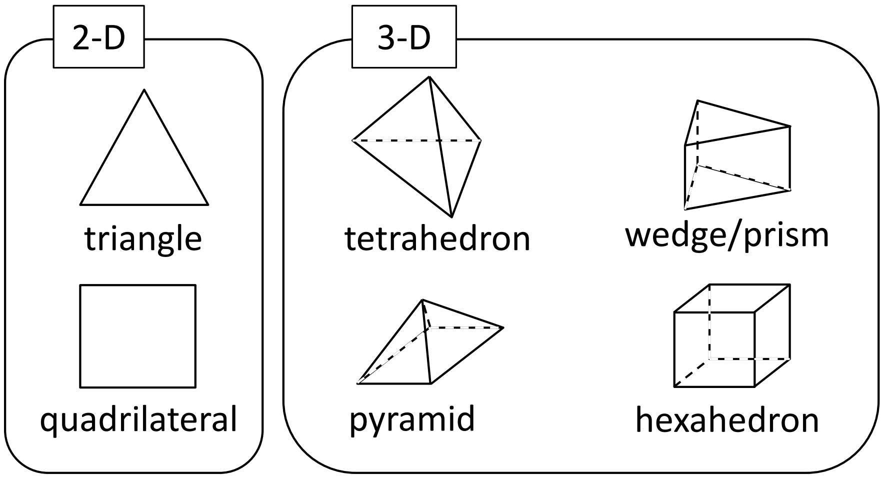


Figure 2.2: Two and three dimensional cells shapes

taken directly from [96]. The details of each of these algorithms are not covered in

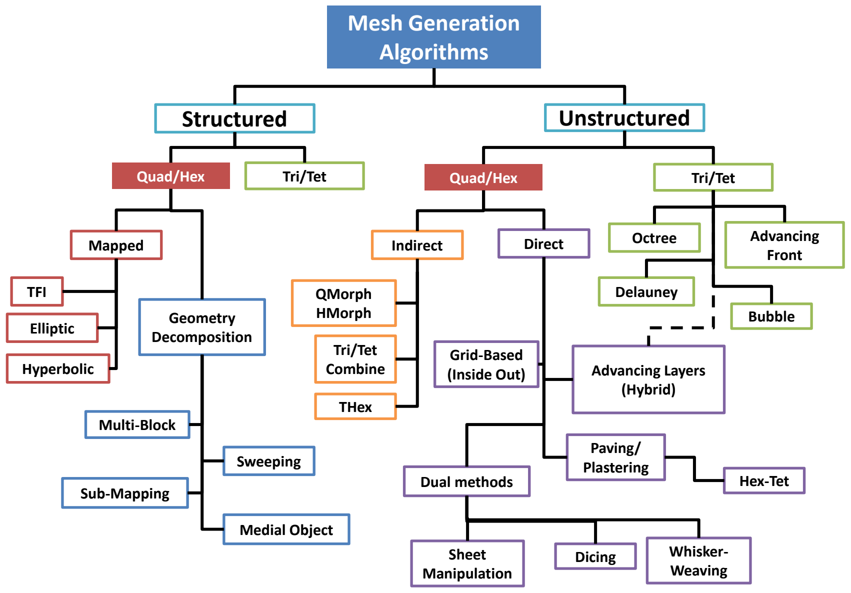


Figure 2.3: Survey of Mesh Generation Algorithms from Owen [96]

the present discussion, and are left for the motivated reader.

2.3.2 Equation Discretization: The Finite Volume Method

The goal of equation discretization is to approximate a partial differential equation (PDE) on a continuous domain with a set of algebraic equations over discretized space. For this discussion, the finite volume method (FVM) is briefly covered through a simple example. For a more detailed treatment, the reader may consult the work of Patankar [98] or Jasak [61].

As an example, consider the conservative differential form of the transport equation of a generic tensoral quantity ϕ (of rank 0, 1, or 2 in general) through time and space, such that:

$$\frac{\partial \rho \phi}{\partial t} + \nabla \bullet \left(\rho \mathbf{U} \phi + J_\phi \right) = S_\phi \left(\phi \right), \tag{2.1}$$

where the flux of ϕ in the convective term (J_ϕ) is commonly taken as Fick's Law of diffusion or the gradient diffusion hypothesis [30] (i.e. $J_\phi = -\Gamma \nabla \phi$). The resulting transport equation is given as:

$$\underbrace{\frac{\partial \rho \phi}{\partial t}}_{\text{time derivative}} + \underbrace{\nabla \bullet \rho \mathbf{U} \phi}_{\text{advection term}} = \underbrace{\nabla \bullet \Gamma \nabla \phi}_{\text{diffusion term}} + \underbrace{S_\phi \left(\phi \right)}_{\text{source term}} , \tag{2.2}$$

where ρ is the fluid density, $\mathbf{U}$ is the velocity vector of the fluid, Γ is the diffusivity of ϕ, and $S_\phi(\phi)$ is a source term. Equation 2.2 describes the rate of change per unit volume (time derivative), the rate of efflux per unit volume (advection term), the rate of transport due to random molecular motion (diffusive term), and the rate of consumption or generation of ϕ per unit volume (source term). The present form of Equation 2.2 describes the conserved transport of basic quantities such as momentum, mass, or enthalpy over a domain.

In the finite volume method, Equation 2.2 is transformed into its integral form and discretized using a generalized form of Gauss's Flux Theorem (Appendix A.1) to approximate the volume integrals with enclosed surface integrals. Converting Equation 2.2 to its integral form in time an space yields:

$$\int^{t+\Delta t} \left[\frac{\partial}{\partial t} \int_{V_P} \rho\phi \, \mathrm{dV} + \int_{V_P} \nabla \bullet (\rho \mathbf{U} \phi) \, \mathrm{dV} \right] dt = \int^{t+\Delta t} \left[\int_{V_P} \nabla \bullet \Gamma \nabla \phi \, \mathrm{dV} + \int_{V_P} S_\phi \, \mathrm{dV} \right] dt, \tag{2.3}$$

where V_P is the control volume. For the remainder of this section, each of the differential operators, source terms, and time integration schemes will be covered.

The Advection Term

In finite volume discretization, the advection term (often used interchangeably with the term convection) in Equations 2.2 and 2.3 is represented as:

$$\int_{V_P} \nabla \bullet (\rho \mathbf{U} \phi) \, \mathrm{dV} = \oint_{S_P} \mathrm{d}\mathbf{S} \bullet (\rho \mathbf{U} \phi). \tag{2.4}$$

Equation 2.4 can further transformed into a linear combination of values around the discretized control volume by:

$$\oint_{S_P} \mathrm{d}\mathbf{S} \bullet (\rho \mathbf{U} \phi) = \sum_f \mathbf{S}_f \bullet (\rho \mathbf{U})_f \phi_f = \sum_f F_f \phi_f, \tag{2.5}$$

where ϕ_f is the face value of quantity ϕ that is evaluated using numerous face interpolation methods including:

- **Upwind Differencing (UD):** This method is first order accurate and bounded using the scheme:

$$\phi_f = \begin{cases} \phi_P & \text{for } F_f \geq 0 \\ \phi_N & \text{for } F_f < 0 \end{cases} \tag{2.6}$$

where P and N are shown in Figure 2.1. Equation 2.6 is built on the assumption that the gradient in ϕ from the cell center to the face from the upwind direction is negligible. The UD scheme is known to be numerically diffusive [44], especially on tetrahedral meshes, decreasing in diffusiveness on polyhedral meshes. Reducing numerical diffusiveness of a solution using UD requires careful choice of meshing structures. Normally, UD should be used as a first approximation, with diffusive errors minimized by reducing the cell size (increasing the total cell count) of the mesh.

- **Linear Upwind Differencing (LUD):** As a method to correct the UD scheme and increase its order of accuracy, the linear upwind scheme can be used. This method is second order accurate and unbounded [104, 44], unless a limiter in the gradient portion of Equation 2.7 is used. The face value of ϕ is calculated by:

$$\phi_f = \phi_i + \nabla\phi_i \bullet \mathbf{r}, \tag{2.7}$$

  where ϕ_i is the upwind cell centered value, that is corrected by the gradient ($\nabla\phi_i$) in the direction from the upwind cell center to the face of the cell (i.e. $\mathbf{r} = \mathbf{x}_f - \mathbf{x}_i$) [58, 104, 3]. The usage of this scheme in OpenFOAM requires the user to prescribe a gradient scheme for each entry using the "linearUpwind" divergence scheme.

- **Linear Differencing (LD):** Equivalent to the Central Differencing Scheme (CDS), this method is second order accurate and can be unbounded [61, 111]. It is basically a distance weighted average of the cell-centered values of the owner and neighbor cells of the face f, determined by:

$$\phi_f = \lambda_f \phi_P + (1 - \lambda_f)\,\phi_N, \tag{2.8}$$

  where

$$\lambda_f = \frac{|\mathbf{x}_f - \mathbf{x}_N|}{|\mathbf{x}_f - \mathbf{x}_N| + |\mathbf{x}_f - \mathbf{x}_P|}. \tag{2.9}$$

  The pitfall of LD scheme is that out of bounds oscillating results can be returned (i.e. unphysical densities, concentrations, temperatures) [44]. If the ratio of convective flux to diffusive flux (cell Peclet number) is less than 2, the LD scheme can be used and generally gives bounded results [44].

- **Limited Linear Differencing (LLD):** This is a total variational diminishing (TVD) method [5], that is second order accurate and artificially bounded and stabilized through a Sweby limiter, such that:

$$\phi_f = (\phi_f)_{UD} + \Psi(r)\left[(\phi_f)_{LD} - (\phi_f)_{UD}\right], \tag{2.10}$$

  where $(\phi_f)_{UD}$ and $(\phi_f)_{UD}$ are the values of ϕ at the cell face determined by upwind differencing and linear differencing respectively. The function $\Psi(r)$ is dependent on the ratio of gradients (r), with:

$$r = \frac{\phi_A - \phi_B}{\phi_C - \phi_A}, \tag{2.11}$$

  where points A, B, and C are chosen depending on the direction of the flow in the face f [61]. For a Sweby limiter [122], the limiting function $\Psi(r)$ in Equation 2.10 is:

$$\Psi(r) = \max\left(\min\left(\frac{2r}{k}, 1\right), 0\right), \tag{2.12}$$

  and $0 < k \leq 1$ is specified *a priori*. In general this “limitedLinear” scheme should be less diffuse than the “linearUpwind”.

The Gradient

The gradient (∇) is used in some of the second order schemes and the non-conservative form of Equation 2.2. In this project, several methods were used to calculate gradient including:

- **Gauss Integration :** Using Gauss's Flux Theorem (Appendix A.1), the gradient term is expressed as an outer product of the surface normal vector and the face value of ϕ using:

$$\int_{V_P} \nabla\phi \,\mathrm{dV} = \oint_{S_P} \mathrm{d}\mathbf{S}\phi = \sum_f \mathbf{S}_f \phi_f, \tag{2.13}$$

where ϕ_f is determined using the methods described in the advection term discretization subsection.

- **Linear Least Squares Gradient:** Discussed by Jasak and Weller [60] and Muzaferija[38], the cell centered linear least squares gradient (LLSG) method represents the gradient term as:

$$\int_{V_P} \nabla\phi \,\mathrm{dV} = (\nabla\phi)_P \, V_P. \tag{2.14}$$

The LLSG approximation is derived by extrapolating the gradient from point P to its neighbor N, and comparing the extrapolated value of ϕ_N with the actual value of ϕ_N. The error at point N is:

$$e_N = \phi_N - \left(\phi_P + \mathbf{r}_{PN} \bullet (\nabla\phi)_P\right), \tag{2.15}$$

where $\mathbf{r}_{PN} = \mathbf{x}_N - \mathbf{x}_P$. We then minimize of the error at point P with:

$$e_P^2 = \sum_N (w_N e_N)^2, \tag{2.16}$$

where the weighting function $w_N = 1/|\mathbf{r}_{PN}|$ leads to the cell-centered gradient expression:

$$(\nabla\phi)_P = \sum_N w_N^2 \mathbf{G}_P^{-1} \bullet \mathbf{r}_{PN} \left(\phi_N - \phi_P\right). \tag{2.17}$$

The symmetric tensor $\mathbf{G}_P$ needs to be calculated at every cell on the mesh using:

$$\mathbf{G}_P = \sum_N w_N^2 \mathbf{r}_{PN}^2. \tag{2.18}$$

$\mathbf{G}_P$ is then inverted, stored locally for future use, and used in Equation 2.17 to give an accurate second order cell centered gradient approximation. For a face centered value of gradient, the $(\nabla\phi)_P$ value from Equation 2.17 can be interpolated between the owner cell P and the neighbor cell N to the common face using:

$$(\nabla\phi)_f = \lambda_f (\nabla\phi)_P + (1-\lambda_f)(\nabla\phi)_N \tag{2.19}$$

where

$$\lambda_f = \frac{|\mathbf{x}_f - \mathbf{x}_N|}{|\mathbf{x}_f - \mathbf{x}_N| + |\mathbf{x}_f - \mathbf{x}_P|}. \tag{2.20}$$

- **Gradient Limiters:** Used in conjunction with Gauss integration or the LLSG methods, gradient limiters dampen oscillations that might appear near rapid changes in the flow field [59]. OpenFOAM provides cell and face based limiters that are either standard or multi-directional. Cell based methods limit the extrapolated face values between the maximum and minimum cell and cell neighbor values and are applied to all components of the gradient vector [4]. Face based methods limit the extrapolated face values between the face-neighbor cell values and are applied to all components of the gradient vector [5]. Face based methods are more numerically diffuse than their cell based counterparts [59]. The multi-directional version of the cell and face based methods work by being applied only to the normal component of the gradient at each separate face of the cell[6] and are less numerically diffuse than the standard limiters [59].

The Diffusion Term

For the diffusion term in Equations 2.2 and 2.3, Gauss's Flux Theorem (Appendix A.1) can again be invoked to linearize the Laplacian (∇^2) operator such that:

$$\int_{V_P} \nabla \bullet \Gamma\nabla\phi \, \mathrm{dV} = \oint_{S_P} \mathrm{d}\mathbf{S} \bullet (\Gamma\nabla\phi) = \sum_f \Gamma_f \left[\mathbf{S}_f \bullet (\nabla\phi)_f\right]. \tag{2.21}$$

For orthogonal meshes, Jasak [61] and Rusche [111] note that the face normal gradient $\mathbf{S}_f \bullet (\nabla\phi)_f$ is simply defined as:

[4]read almost verbatim from cellLimitedGrad.H in the OpenFOAM code
[5]read almost verbatim from faceLimitedGrad.H in the OpenFOAM code
[6] from cellMDLimitedGrad.H in the OpenFOAM code

$$\mathbf{S}_f \bullet (\nabla\phi)_f = |\mathbf{S}_f| \frac{\phi_N - \phi_P}{\mathbf{r}_{PN}}. \tag{2.22}$$

An alternative that is again noted by Jasak [61], is the use of Equation 2.19 to interpolate a cell-centered gradient to the face and dot this vector with $\mathbf{S}_f$, but is ultimately undesirable due to its larger computational molecule.

For non-orthogonal meshes, the accuracy of Equation 2.22 is decreased and a correction approach where:

$$\mathbf{S}_f \bullet (\nabla\phi)_f = \underbrace{\mathbf{\Delta}\,(\nabla\phi)_f}_{\text{orthogonal contribution}} + \underbrace{\mathbf{k} \bullet (\nabla\phi)_f}_{\text{non-orthogonal contribution}} \tag{2.23}$$

must be used to maintain accuracy. Vectors $\mathbf{\Delta}$ and $\mathbf{k}$ must sum to equal $\mathbf{S}_f$, and are determined using the non-orthogonality treatment covered by Jasak [61]. For brevity, they are only mentioned as the minimum correction approach, orthogonal correction approach, and over-relaxed approach.

The Source Term

The treatment of source terms is an integral portion of the finite volume discretization process, discussed in more detail by Patankar [98]. Ultimately, using an implicit formulation of the FVM builds a set of linearized equations that are iteratively solved with guesses from previous solution steps. Since the source term is a cell centered value, it can contribute directly to the matrix diagonal, thus having a strong influence on stability in the solution process. In much the same way that we have linearize the differential operators, we must linearize source terms to accurately describe them physically and provide stability to the solution process. The source term in Equation 2.3 is represented as a combination of an implicit and explicit component such that:

$$\int_{V_P} S_\phi \, \mathrm{dV} = \underbrace{S_E V_P}_{\text{explicit source}} + \underbrace{S_I V_P \phi_P}_{\text{implicit source}} \tag{2.24}$$

where V_P is the control volume, with S_E and S_I being explicit and implicit source constants that may themselves depend on previous values of ϕ. For extremely complicated sources (e.g. stiff or coupled reaction rates), the only option may be to represent S_ϕ as a fully explicit $S_E(\phi^*)$ term that is determined from a separate ordinary differential equation solver. Lastly, in order to maintain stability, S_E must always be negative [98].

Time Derivatives

In this research, both steady-state momentum and time-dependent scalar transport simulations were performed. In the case of the steady state solutions, no treatment of the time derivatives is necessary. However, for the time dependent scalar transport equations, a first order unconditionally stable Euler implicit method was used. Referring to Equations 2.2 and 2.3, the time derivative term can be approximated as:

$$\int_{V_P} \frac{\partial \rho\phi}{\partial t} \mathrm{dV} = \frac{\rho_P^n \phi_P^n - \rho_P^0 \phi_P^0}{\delta t} V_P, \tag{2.25}$$

where the implicit $\phi_P^n = \phi_P\left(t + \delta t\right)$ is being solved as a system of linear equations and the $\phi_P^0 = \phi_P\left(t\right)$ is the solution from the previous time step. The final discretized[7] form of Equation 2.3 now becomes:

$$\underbrace{\rho_P \frac{\phi_P^n}{\delta t} V_P}_{\text{implicit time portion}} + \underbrace{\sum_f F_f \phi_f^n}_{\text{advection term}} - \underbrace{\sum_f \Gamma_f \left[\mathbf{S}_f \bullet \left(\nabla \phi^n\right)_f\right]}_{\text{diffusion term}} - \underbrace{S_I(\phi^0) V_P \phi_P^n}_{\text{implicit source}} = \underbrace{S_E(\phi^0) V_P}_{\text{explicit source}} + \underbrace{\rho_P \frac{\phi_P^0}{\delta t} V_P}_{\text{existing time solution}} . \tag{2.26}$$

[7]It is "customary" to neglect the change of ϕ, the face values, and gradient terms over the infinitesimal time-step[111], but the assumption is what leads to the large errors and first-order nature of the implicit Euler scheme.

Using the face interpolation (UP, LUD, LD, LLD, etc.) and gradient approximation (Gauss Integration, LLSG, etc.) methods previously discussed to transform ϕ_f values to values of ϕ in terms of owners (P) and neighbors (N), the simplified general form of Equation 2.26 now becomes:

$$A_P \phi_P^n + \sum_N A_N \phi_N^n = b_P. \tag{2.27}$$

Under-Relaxation in Steady-State

For most situations in CFD, a steady-state solution is sought after as an initial condition for transient calculations or used to completely decouple the Navier-Stokes equations from scalar transport (e.g. one-way coupling). As a method to reduce instability during the iterative solving process, all segregated solvers (i.e. components are solved separately and coupled explicitly) in OpenFOAM will use under-relaxation through adding an additional term to both side of the algebraic system of equation such that

$$\frac{A_P}{\alpha} \phi_P^n + \sum_N A_N \phi_N^n = b_P + \frac{1-\alpha}{\alpha} A_P^0 \phi_P^0, \tag{2.28}$$

where $\alpha = (0, 1)$. In Equation 2.28, the superscript 0 represents the solution from the previous iteration. The benefit of under-relaxation is that additional diagonal dominance is added to the linear system, thus improving stability. However, at low value of α the solution progression is stable, yet the rate of convergence is drastically slowed. It is wise to choose values of α that ensure stability, at a quick rate of convergence. Finally, Equation 2.28 can be completely contracted into:

$$A\phi = b. \tag{2.29}$$

Although there are many methods used to solve such systems of equations, because the A matrix is very large and sparse, experience has shown that iterative solvers can be extremely efficient compared to direct solvers. Moreover, the potential for increasing the speed of finite volume solvers resides in the acceleration of iterative linear system solvers. This final thought is addressed in Chapter 6 in detail.

2.4 Closure

A brief discussion defining computational fluid dynamics (CFD) and introducing discretization was provided. An overview of the entire process of using CFD as a design and investigation tool was given. Several key points can be distilled from the provided discussion:

- **Computational Fluid Dynamics (Section 2.2)** CFD has become a widely accepted tool for investigating complex fluid flows and physical phenomena. In its purest form, CFD is a set of computational tools and numerical methods to provide approximate solutions to systems of partial differential equations encountered in the study of fluids and transport phenomena. Throughout this thesis, preprocessing, model development, code execution, and post processing steps are reoccurring discussion topics. Moreover, strategies of improving these steps are given with a specific slant in solving the complex fluid fields in packed-beds.

- **Domain Discretization (Section 2.3.1)** The search for a more perfect approximation of continuous space in the form of a mesh is the subject of much research in mathematics and engineering [96]. As a result, there is a variety of mesh cell types and descriptions across many applications and in some cases are problem specific [98]. For example, hexahedron meshes are known to produce accurate solutions for boundary layer flows due to the unidirectional nature of the flow near the solid surface in the base of the boundary layer. Likewise, tetrahedron meshes are considered to be a very poor choice for highly turbulent flows due to orthogonality and numerical diffusion issues, yet may be the only choice of mesh for extremely complex geometries. For this project, as we shall see in Chapters 3 and 4, errors were minimized and results were improved by using arbitrary polyhedron meshes. Consequently, a higher level of confidence in the solution is gained in the packed bed modeling work presented.

- **Equation Discretization (Section 2.3.2)** As Chapter 4 will reiterate, solution algorithm stability and accuracy is tremendously influenced by the choice of discretization schemes. For the most part, second order upwind schemes have been used in the advection schemes while least squares gradient schemes have been used for gradient approximations. The current introduction to the reader of the finite volume method was provided for reference in later chapters. The interested reader should consult works by Patankar [98], Versteeg [128], Jasak [61] and the appendix of this thesis for additional explanation.

Chapter 3

Constructing Randomly Packed Beds of Cylindrical Particles

3.1 Introduction

One of the most important and time-consuming portions of CFD research is the domain and mesh generation steps. In this chapter, a Monte-Carlo packing algorithm is discussed at length with particular attention to the particle-particle overlap determination. If the reader is familiar with packing algorithms, Section 3.2 may be skipped. The remaining sections in the chapter (Section 3.3) cover the steps for generating a Delaunay mesh for conversion to an arbitrary polyhedral mesh and eventual parallel decomposition.

3.2 Geometry Generation Using a Monte-Carlo Packing Algorithm

3.2.1 Packing Methodology

Individual Particle Representation

The representation of the particle structure must be simple, require little computer memory, and be easily translated into other formats for computational mesh generation. The representation of the particles used in this algorithm describes the dimensions, location, and orientation of each particle in three-dimensional Cartesian space, and is shown in Figure 3.1(a). The dimensions of each particle are represented as a cylinder length and radius.

In Figure 3.1(a), the location of each cylinder is described by the position vector V_{pos}, containing the x, y, and z-location of the centroid (located at $r = 0$ and $z = L/2$) of each cylinder with respect to a global coordinate system. The orientation of each cylinder is described by an orientation vector (U) aligned with the primary axis of the cylinder in addition to the angles θ, ϕ, and τ. All particle rotations are performed with respect to a local coordinate system (x', y' and z'), with a local origin at the particle centroid. Angle θ is the amount of rotation around the local x'-axis of a projection of U onto the plane z'-y' represented as $U_{pz'y'}$; angle ϕ is the amount of rotation around the local y'-axis of a projection of U onto the plane x'-z' represented as $U_{px'z'}$; and angle τ is the amount of rotation around the local z'-axis of a projection of U onto the plane y'-x' represented as $U_{py'x'}$ about the centroid. The sign of the rotation follows the right hand rule around each of the axis, and is shown in Figure 3.1(b).

The description of more complex particles, such as a trilobe (Figure 3.2(a)) or quadlobe (Figure 3.2(b)), is merely a superset of member cylinders with a specified length and radius. The global position of the particle is described with a vector V_{pos} from the global origin to the particle centroid in terms of x, y and z coordinates. Each of the member cylinder is defined by the location of each member centroid on a local particle coordinate x', y', and z' similar to Figure 3.1(a). The orientation of each

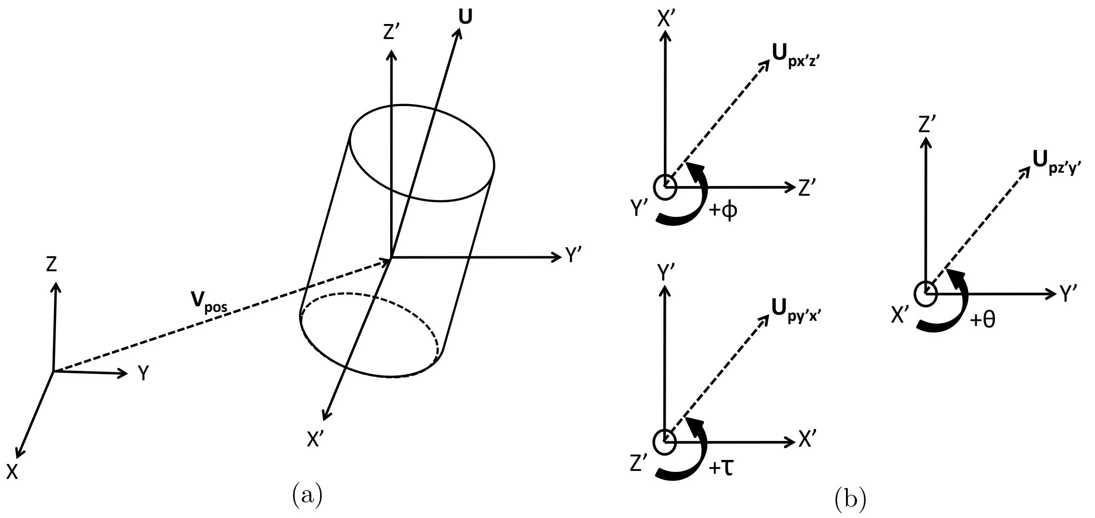


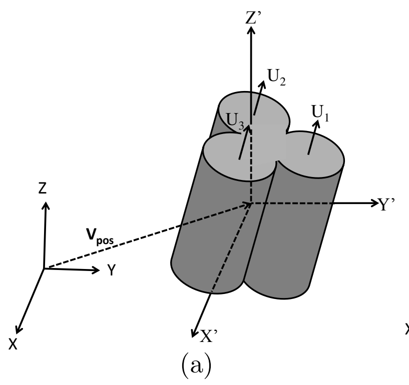
Figure 3.1: Particle representation (a) Definition of local (x',y', and z') and global position (x, y, and z) in relation to position vector V_{pos} with an orientation vector U; (b) Definition of the orientation vector projections $U_{py'x'}, U_{px'z'}$ and $U_{pz'y'}$ around local axes with respect to the angles θ, ϕ, and τ using the right hand rule.

member cylinder is described by a member orientation vector U_n oriented along the primary axis of each member cylinder in addition to the angles θ_n, ϕ_n, and τ_n. The description of angles θ_n, ϕ_n, and τ_n with respect to the vector U_n is similar to the relation between vector U and angles θ, ϕ, and τ shown in Figure 3.1(b).

Packing Algorithm

The goal of the algorithm is to randomly pack particles within a specified container. In addition, the particle population was to have a normal distribution of particle lengths and radii described by a mean length and radius as well as a standard deviation in both measurements. Lastly, the algorithm was to "compress" the particles in the negative z-direction without including external forces or a force field in the simulation. The key steps of the developed procedure are given in Algorithm 1.

The first step is to adjust each individual particle's dimensions so that the population has a normal distribution of lengths and radii. The second step is to determine an



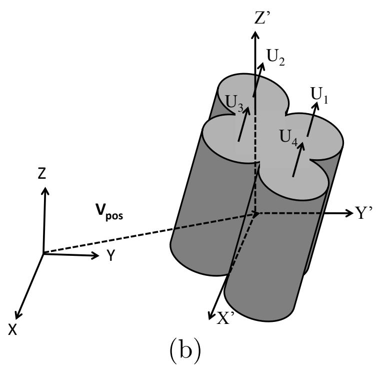


Figure 3.2: Relation of local (x', y', and z') and global (x,y, and z) coordinate system using a particle position vector V_{pos} for (a) Simple trilobed particle with member orientation vectors U_1, U_2 and U_3; (b) Simple quad-lobed particle with member orientation vectors U_1, U_2, U_3 and U_4.

Algorithm 1: Proposed Monte-Carlo packing algorithm

```
1. Adjust particle dimensions
2. Generate random initial condition
3. Perform packing cycles
    (a) Sort particles and create a list
    (b) Move through sorted list and place particle
        i. Rotate and translate particle in random manner
        ii. Check that the particle is within the packed bed boundaries
        iii. Check for particle-particle overlap
    (c) Lower the domain boundary
4. Output final location of particles and generate computational mesh
```

initial condition by placing each particle in the packing domain. Additionally, the particle-particle overlap during the initialization step only considers packing spherocylinders and is explained in the overlap determination section more thoroughly. In the case of a packed bed reactor, a cylindrical container is chosen. This cylindrical container requires a radius and initial height. A key point is that in order to produce an initial condition quickly, an initial bed height many hundreds or thousands of times greater then the final anticipated bed height is required. The key is to determine an initial condition and let the bulk of the packing and compression to take place during the packing cycles.

The third step is the core of the packing algorithm and consists of performing packing cycles. First, an ordered list of particles in the initial condition from lowest z-location to highest is constructed. Moving through the list, starting from the particle closest to the bottom of the bed to top, each particle is packed one at a time. The use of the list ensures that there is space below the particle being moved. Each particle is rotated a random amount around the x', y', and z'-axis and then randomly translated in the +/-x and +/-y directions and only the negative z direction in Cartesian space. By only allowing downward motion in the z direction, we are ensuring that the bed is becoming more packed with each packing cycle. As each particle is moved, it must be checked to see that it is within the boundaries of the packed bed and also that it is not overlapping with any other particle. Further discussion of the specific overlap determination algorithm appears in overlap determination section. Finally, at the end of each packing cycle the height of the boundary of the packing domain is lowered, resulting in a decrease in the overall porosity.

The fourth step of the packing algorithm is to output the final location and orientation of the particles in the packed bed. This output is used as an input file for meshing or computer aided drafting software to create a mesh of the packed domain. For this project, GAMBIT version 2.3.16 by Ansys Inc. (www.fluent.com) was used to create the initial meshes. As input, GAMBIT uses a journal file script to place each particle in the domain and then scales the particles to the appropriate length to produce a mesh of the packed bed. Before the meshing process is discussed, the overlap determination method needs to be addressed.

Overlap Determination

The packing algorithm section outlined a general method to pack particles of any shape. The difficulty, is to determine if two particles overlap each other. In the case of spheres, the overlap determination is trivial. If the center of sphere one and sphere two are closer than the sum of radius one and radius two, there is an overlap. For the case of more complex shapes (cylinders, trilobes, and quadlobes), the determination is more difficult. For cylinders, the overlap determination follows the algorithm proposed by Blaak et al.[18] with an additional large distance exclusion step. The overlap between two cylinders (one being moved and one being stationary) is determined through the following steps:

1. **Large distance exclusion**: Given an average cylinder length of $\bar{L}$ and average cylinder radius $\bar{R}$. If $\bar{L} > \bar{R}$, treat both cylinders as spheres of radii $L_1/2$ and $L_2/2$ and calculate the distance between the spheres to determine overlap. Similarly, if $\bar{L} < \bar{R}$, treat both cylinders as spheres of radii R_1 and R_2 and calculate the distance between the spheres to determine overlap. If there is an overlap move to the next step to determine spherocylinder overlap, otherwise the move is accepted.

2. **Spherocylinder overlap**: Approximate the two cylinders as two separate stacks of overlapping spheres. Spheres are only allowed to overlap spheres approximating the same cylinder. If the spheres of one stack overlap any sphere from the second stack, move to the next step in the overlap determination. Otherwise, the two cylinders do not overlap and the move is accepted.

3. **Disk-disk overlap**: Treating only each end of the two cylinder as a separate thin ellipsoid, determine if the ellipsoids from one cylinder overlap the ellipsoids of another cylinder. If there is an overlap, then the move is not accepted, the particle is returned to it's previous position, and a new move of the same particle must be made. If no overlap is determined, then proceed to the next step.

4. **Disk-cylinder overlap**: This last step is used to determine if the ends of one cylinder overlap with the cylinder shaft of the second cylinder. Treat the ends of one cylinder as two separate thin ellipsoids and approximate the second cylinder as many stacked thin ellipsoids. Determine if the ellipsoids of the fist cylinder overlap the stack of ellipsoids in the second cylinder. If there is overlap, then the move is not accepted, the particle is returned to it's previous position, and a new move of the same particle must be made. Otherwise, there is no cylinder-cylinder overlap and the move is accepted. Repeat this last step again by approximating the first cylinder as a stack of thin ellipsoids and only the ends of the second.

The determination of ellipsoid-ellipsoid overlap is performed using the method outlined by Perrem [102, 103]. Originally used in modeling molecular fluids, the method can detect ellipsoid-ellipsoid overlap without calculating points on the surface of the two hard objects.

The particle-particle overlap determination method outlined is capable of detecting the overlap of cylindrical particles or any particle consisting of a subset of cylindrical particles. This includes trilobes and quadlobes or more complex particles. What must be known is the location and orientation of each member (subset) cylinder in relation to each other on a local (to the particle) Cartesian coordinate system. As shown in Figure 3.2(a) and 3.2(b), each particle has a V_{pos} vector for the entire particle, but separate U_n vectors for each cylindrical particle member where the subscript n describes member n. Additionally, each particle member is identified by a location in the local x', y', and z' particle coordinate system. The determination of particle-particle overlap for more complex particles relies on the rule that members of the same particle may overlap, while members of two different particles may never overlap each other.

Radial Porosity Distributions

Porosity distributions are used to describe the structure of a packed bed. Often these distributions are expressed with respect to a particular direction along a line or axis in a packed bed (eg. r or z-direction). A simple mathematical definition of porosity along a line is described using a point void fraction ϵ_p, where a value of 0 is assigned if the point is located in a void space between the particles and 1 if the point is located inside a particle [110]. The values of point void fraction along the line is the porosity distribution along the line L. Summing all of these point void fractions along a line L and dividing by the length of the line will give a line void fraction described by:

$$\epsilon_L = \frac{1}{L}\int_L \epsilon_p \, dL \tag{3.1}$$

with L being the total length of the line. Similarly, an area void fraction ϵ_A and volume void fraction ϵ_V (equivalent to bulk porosity ϵ_{bulk}) are given as:

$$\epsilon_A = \frac{1}{A}\int_A \epsilon_p \, dA \tag{3.2}$$

$$\epsilon_V = \epsilon_{bulk} = \frac{1}{V}\int_V \epsilon_p \, dV \tag{3.3}$$

These types of descriptions using point voidage and distribution are convenient for direct experimental investigation of bed porosity, but requires many points to reduce overall standard deviation in measurements. However, for a packing simulation, information about particle location and orientation is readily available as a set of data. Given a set C that describes the container of the packed bed and the set P that describes N particles, the set describing the packed bed domain D is given as:

$$D = C - P \tag{3.4}$$

where:

$$P = \{P_0, ..., P_N\} \tag{3.5}$$

and P_N describes particle N in the set of particle P.

Similarly, given S_r as a cylindrical slice at r that is described as a hollow cylinder without caps and a wall thickness of dr, the value of radial porosity ϵ at r is given as:

$$\epsilon_r = \frac{Volume(S_r \cap D)}{Volume(S_r)} \tag{3.6}$$

where $Volume(S_r \cap D)$ is the volume of the intersecting domains S_r and D, and $Volume(S_r)$ is the original volume of the slice S_r. Shown in Figure 3.3, the intersection of S_r and D yields a slice capturing the void space between the particles in the bed at distance r from the center, through an angle from 0 to 2π around the bed, over the entire height of the bed. The hollow portions seen in $S_r \cap D$ labeled object in Figure 3.3 represent the internal volumes of the particles that are excluded from domain D. The volume of the slice S_r is determined by:

$$Volume(S_r) = Z\pi(R_{outer}^2 - R_{inner}^2) \tag{3.7}$$

where Z, R_{outer}, and R_{inner} are the bed height, slice outer radius, and slice inner radius respectively with $dr = R_{outer} - R_{inner}$.

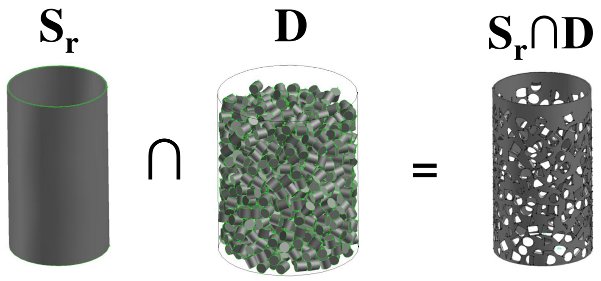


Figure 3.3: The intersection ($\cap$) of a cylindrical slice S_r and packed bed domain D yields an object whose volume equals the volume of the void space at distance r from the bed center, through an angle from 0 to 2π around the bed, over the entire height of the bed.

The radial porosity distribution $\epsilon(r)$ is described as the set of porosity values from zero to bed diameter (R) or:

$$\epsilon(r) = \{\epsilon_0, ..., \epsilon_r, ..., \epsilon_R\} \tag{3.8}$$

with the bulk porosity given as:

$$\epsilon_{bulk} = \frac{1}{R}\int_0^R \epsilon(r)dr \tag{3.9}$$

Descriptions of computational domains are often expressed as discrete representations of continuous objects agglomerated as data points into data sets, making the definition of a radial porosity distribution using sets a much more appropriate method. This definition, rather than using a point void fraction definition, reduces overall standard deviation in measurement of radial porosity. Furthermore, since many commercial computer aided drafting and meshing software uses sets to describe a domain, it is more natural to use and can easily be integrated in the meshing process.

3.2.2 Results and Discussion

The overall process of the creating a packed bed with the packing algorithm involves providing input information, executing the algorithm, and producing a file that is used to generate a computational mesh for CFD simulation. Initially, information describing the particle dimensions and particle quantity, along with the dimensions of the container are given. The algorithm moves each individual particle in a random fashion over a certain number of steps to arrive at a predetermined ending point. At the conclusion of the packing algorithm, the location and orientation of each particle is known and used in a meshing software to generate a computational mesh. Details of this process have been given in previous sections. Here, a brief case study will be covered to show typical results, compared to experimental work, with an explanation of the behavior during the packing process.

Initial Condition

For this case study, 1000 cylindrical particles of radius 0.25 +/- 0.00 units (i.e. zero standard deviation in particle radius) with a length of 0.50 +/- 0.00 units are packed into a container of radius 3.35 units with an initial height of 50,000 units. The units of the packed bed are arbitrary since the packed bed can be scaled easily after a mesh has been generated. The following sections will discuss the convergence, bulk porosity, and radial porosity distributions during the packing process.

Convergence

A convergence plot of bulk porosity or cycle time versus packing cycle is useful in analyzing the behavior of the algorithm as the packed bed is being created. Figure 3.4 exhibits typical behavior for a packing run. Looking at the bulk porosity series, there are three main stages A-B, B-C, and C-D. These three stages are also seen in the cycle time series as A'-B', B'-C', and C'D'.

The initial stage of packing from point A to B and A' to B' is characterized by a large container with large spacing between particles. Each move to compress and lower the

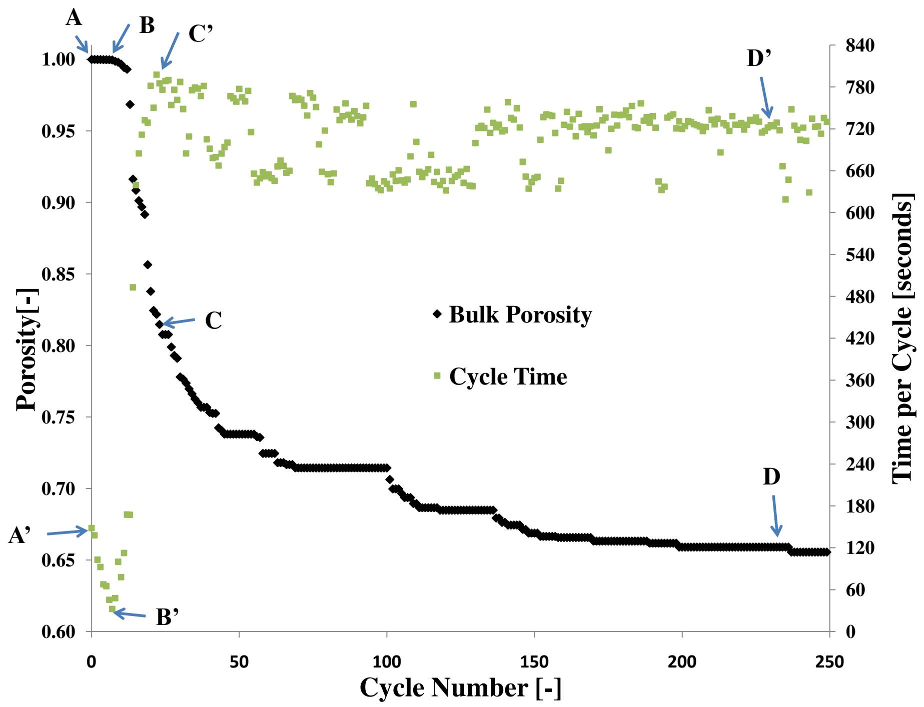


Figure 3.4: Bulk porosity convergence history and cycle time data for 1000 packed cylinders packed into a tube container for 250 packing cycles using proposed algorithm.

upper boundary of the container is fast and does not readily effect the bulk porosity since the container was initially set to be much higher than the expected final bed height. Having a large initial height is necessary in obtaining an initial condition in a timely manner rather than trying to obtain an initial condition close to the final packing structure. Performing the latter would require tremendous time that is better spent on fast packing cycles seen in the second stage. At B' there is a global minimum in time per cycle seen as a tipping point at B, beginning rapid convergence of bulk porosity.

The second stage, seen in the bulk porosity series as B to C shows rapid convergence in which particles are moved and the container height is lowered with each packing cycle very rapidly. With each move, the degrees of freedom for each particle become more limited and the cycle time increases as seen in the cycle time series from B' to C'.

The final stage of the packing process is seen as a long tail from C to D and C' to D'. Because there is little room for movement, small translational and rotational movements of the particles are performed to further pack the particles in the bed. During this stage, nearly 96 percent of the total time is spent decreasing bulk porosity by 49 percent of overall change in bulk porosity from 1 to 0.65. The cycle time series in Figure 3.4 shows that some packing cycles are completed in less time than others, seen as oscillations from C' to D'. It is also noted that there are decreasing oscillations as the number of packing cycles progress. This decrease in oscillations in the cycle time series indicates that the algorithm produces less change in the bulk porosity, reaching an asymptotic limit of bulk porosity as seen in C to D in the bulk porosity series.

The overall shape of both the cycle time series and bulk porosity series are similar for different inputs (particle and container size and shape). Shifts in the maximum height (from B' to C') and total area under the curve of the cycle time series data (total packing time) are drastically changed with increasing and decreasing the number of particles in the system. This is consistent with the fact that it takes longer to complete a cycle with more particles present in the packed bed and the total time will be higher for more particles. In terms of the bulk porosity curve, the shape and rate of convergence are similar for all sizes (total numbers of particles) of packed beds in the same size container unless there are too few particles to fill the container bottom more than a single layer. In the latter case, the length of B to C will be shorter and the last stage from C to D will become much flatter. This comparison is only true if the size and shape of the particle and container are held constant, changing only the total number of particles. A detailed study of these effects is not presented in this discussion for the purpose of brevity.

The packing results are reproducible with little deviation in bulk porosity over the duration of the packing simulation. Figure 3.5 shows an averaged porosity convergence curve over 5 runs with 95 percent confidence intervals above and below the mean porosity value. The region of least variability is A-B. This is expected, since we are measuring the bulk porosity of an extremely large and evenly spread domain of particles not strongly influenced by the change in height of the container. The region for greatest variability is C-D, which is expected since the algorithm is stochastic process in which the final positions of the particles are achieved through random motion.

Though the simulation moves particles randomly, the commonality of the particle and container dimensions of the 5 runs should produce similar final bulk porosities. This is evident by a similar shape of the convergence curve for each of the runs, ending in a long tail that approaches a relatively steady state value.

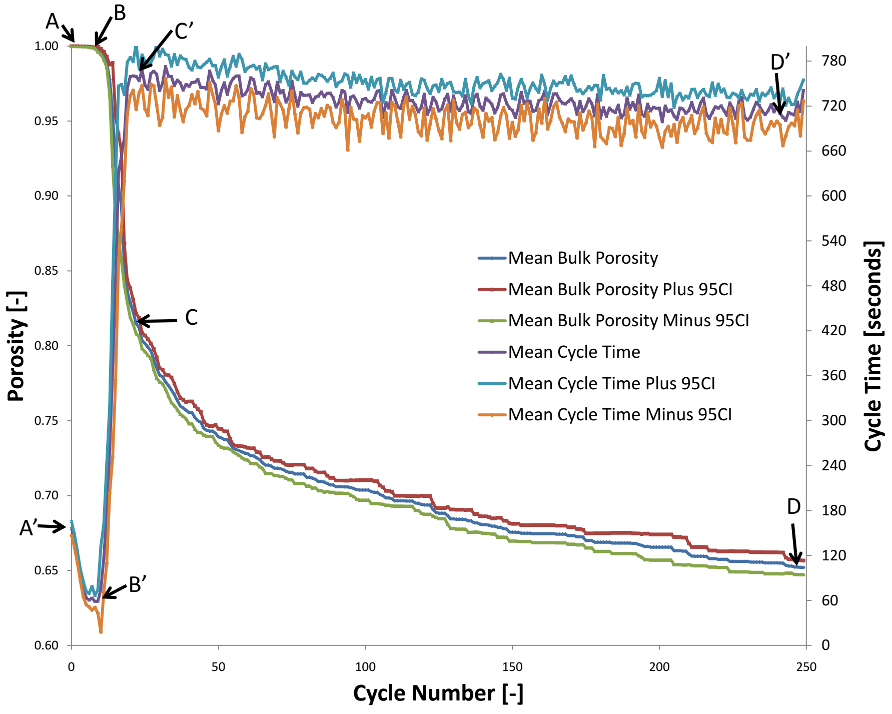


Figure 3.5: Bulk porosity convergence history and cycle time data with 95 CI bands based on 5 simulations with the same initial particle and container shape, for 1000 packed cylinders packed into a tube container for 250 packing cycles using proposed algorithm.

Similar to the porosity convergence curve, the cycle time series data shows little deviation over the duration of the packing simulation. Again, Figure 3.5 shows a cycle time curve averaged over 5 separate runs with 95 percent confidence intervals above and below the average value. The region of least variability is from B'-C', with a slight jump in variability at B', signaling the sharp drop in the convergence curve from B to C. The region of most variability is B'-C', due to the stochastic nature of the algorithm allowing faster cycle times in some cases and slower cycle times in others.

The last observation that can be made about the convergence curves in Figures 3.4 and 3.5 is a "stair-stepping" effect seen in the region C-D. This is characterized by several repeated identical bulk porosity values followed by a sudden drop in bulk porosity. This phenomena could be explained by the presence of several packing cycles being completed in which the upper most particles are not moved or rotated enough to lower the overall bed height. As a new packing cycle starts, lower particles are moved enough to allow higher particles to be rotated and moved into new positions. This effect then allows subsequent particle moves higher in the bed, eventually lowering the total bed height (i.e. reducing container volume) thus reducing the overall bulk porosity.

Radial Porosity Distributions

Figure 3.6 shows the packing-cycle evolution of the radial porosity distribution as the algorithm progresses through the different stages outlined in Figure 3.4. The abscissa represents dimensionless distance from the wall as $(R_{container} - r)/d_{particle}$, with $R_{container}$, r, and $d_{particle}$ being the container radius, radial position of measurement, and particle diameter respectively. The ordinate represents the value of porosity determined using the method outlined in a previous section in equations 3.4 through 3.6. As a note, small outliers from the smooth curve in Figures 3.6 and 3.7 are due to rounding errors produced during the intersection of the slice and domain ($S_r \cap D$) in Equation 3.6 using the GAMBIT software.

The initial condition in Figure 3.6, shown as the "Step 0" series, shows that the vast majority of the particles are packed along the centerline of the tube container. This is evident by the slightly decreasing radial porosity distribution very far from the wall. At step 5, the particles have spread apart more to create a uniform distribution throughout the tube container. As the algorithm progresses to step 21, the effects of the wall on the packing process are seen as a local minimum around 1.9 particle diameters from the wall. From steps 50 to 245, the effect of the wall is more apparent as the cylinders become more packed into the bed. The oscillations seen in the radial porosity distribution show that there are locally more dense rings followed by locally less dense rings of packed particles. This wall effect decreases away from the wall, evident by a decreasing oscillation amplitude until 5 or 6 particle diameters. This

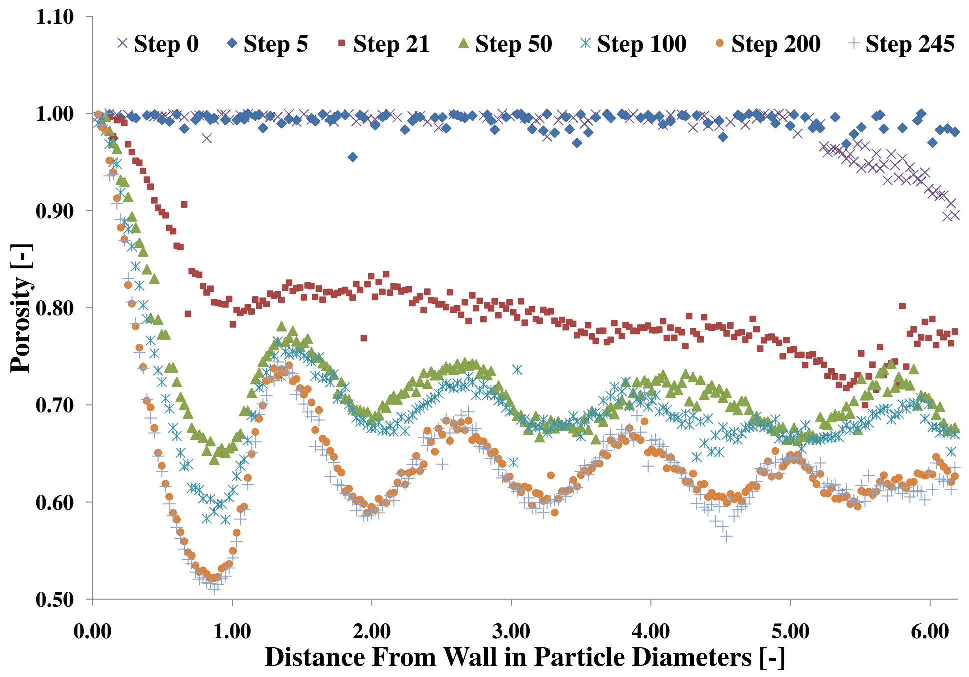


Figure 3.6: Radial porosity distribution evolution starting at an initial condition (step 0) through 245 packing cycles (step 245).

behavior has been thoroughly studied[37] in spherical packings in tubes of low tube to particle diameter, with similar wall effects and decreasing oscillations. It is important to note two main characteristics; 1) near the wall the bed is very porous compared to the bulk and; 2) less wall effects are prevalent in structures towards the center of the packed bed i.e. the oscillations are decreasing in magnitude. Both of these characteristics are seen both in low tube to particle diameter systems and larger industrial packed beds. However, oscillations eventually dissipate in larger packed tubes, showing no oscillations at the center.

Figure 3.7 shows a comparison of the radial porosity distribution at step 245 from Figure 3.6 with an experimentally determined radial porosity distribution by Roblee et al. [110] having identical particle and container dimensions between experiment and the algorithm input parameters. The most obvious characteristic is that the porosity distribution produced by the algorithm shows similar wall effects compared to the experimental curve in Figure 3.7. The influence of the wall decreases towards the center of the bed, evident by decreasing oscillations at greater distances from the wall seen in both experimental results and the results from the packing algorithm. The overall bulk porosity is smaller in the experimental curve compared to the algorithm, as evident by a smaller area under the radial porosity curve for the experimental data versus the area under the distribution produced by the present algorithm. The approximate bulk porosity of the data from Roblee is around 0.2578 determined using equation 3.9 and numerically integrated using the trapezoid rule. This level of porosity shows an extremely tight packed bed and is near to the porosity ($1-\pi/\sqrt{18} = 0.25952$) of spheres in a face centered cubic packing orientation [31]. The value of the bulk porosity reported by Roblee et al. (ϵ_{bulk}=0.2578) is much lower than those seen in Computed Tomography ($\epsilon_{bulk} = 0.354$) and the program DigiDEM ($\epsilon_{bulk} = 0.423$) for similar particle shapes reported in the literature [21]. It is also seen that the near wall porosity is greater for the algorithm curve. Simply put, the Algorithm produces a loosely packed bed that becomes slowly compressed over time as shown in Figure 3.6. This is partially due to the algorithm itself not including external forces (compressive forces) to pack the particles, but rather packing the particles by slowly limiting the degrees of freedom of the moving particles to approach a more packed state. External forces (gravity, manual compression, etc.) influence the direction that a particle is allowed to pack, rather than allowing a particle to vibrate into a lower energy position through random motion. For the algorithm to achieve a much lower void fraction there must be many more packing cycles or external forces (i.e. an additional force field) moving and compressing the particles in the bed. Lastly, it is seen that the experimental curve shows a much greater degree of packing near the wall, seen by a local minimum closer to the wall compared to the algorithm. This is due to the fact that a greater degree of packing is achieved experimentally, compared to the algorithm.

More Complex Particles

More complex trilobed and quadlobed particles are often used in industrial hydroprocessing to alter the diffusional characteristics of the particles in the process of catalyst tailoring [68]. Because of the additional level of complexity of the catalyst shape, subsequent domains and meshes are also more complicated. Figure 3.8(a) shows a packed bed geometry of 500 simple trilobed particles, each consisting of 3 member cylinders of radius 0.25 units and length 0.5 units placed in an equilateral triangle orientation spaced 0.125 units apart. After 50 packing cycles, the bulk porosity of the packed bed is around 0.75, where approximately 0.07 is attributed to the empty space above

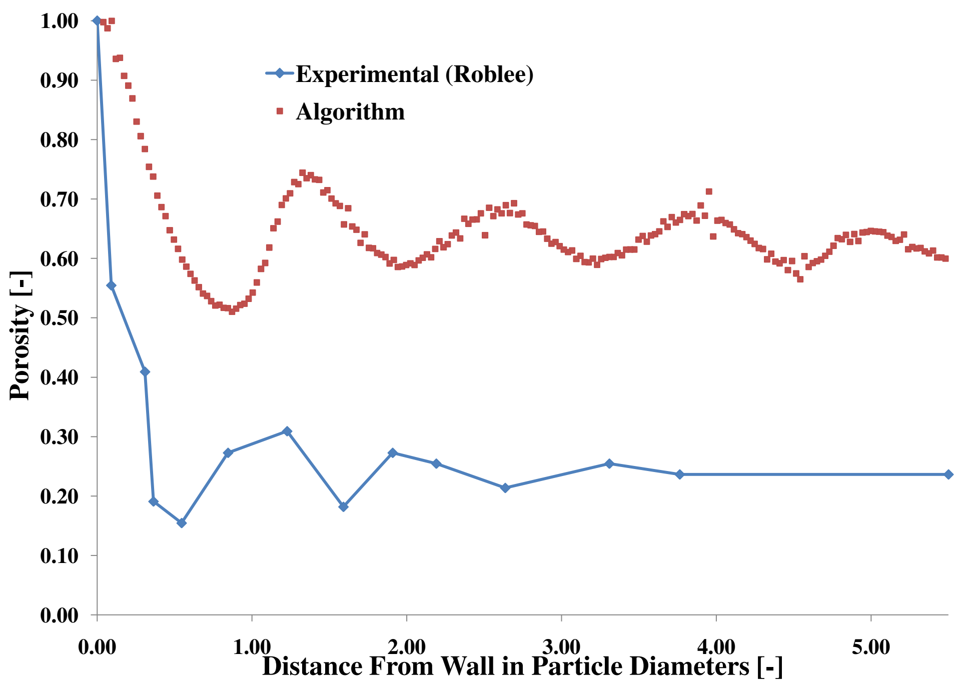
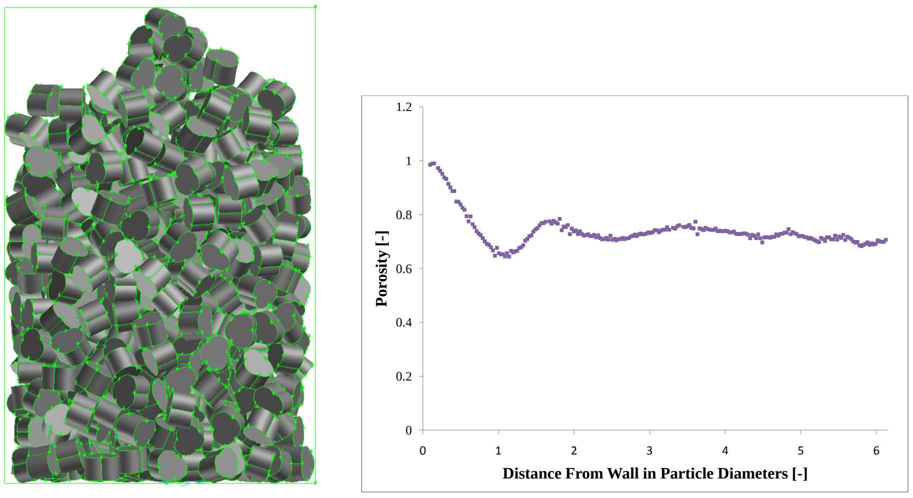


Figure 3.7: Comparison between radial porosity distributions produced by the algorithm and experimental results by Roblee et al. [110]

the particles. The bulk porosity of the packed bed of trilobes is greater than the previously reported bulk porosity of a packed bed of cylinders. This is due to the low number of packing cycles completed in this specific case. Also, the radial porosity distribution in Figure 3.8(b) shows similar wall effects seen previously in Figures 3.6 and 3.7 for a packed bed of cylinders. The main difference between Figure 3.8(b) and 3.6 is that the trilobe particles have not packed as tightly throughout the bed and near the wall of the container compared to the packed bed of cylinders. Once again this is attributed to the low number of packing cycles completed, but it is expected that a lower bulk porosity can be achieved with more packing cycles and show similar behavior to the results in Figure 3.6. Due to the increased complexity of the trilobed particles, the overall simulation time is greater compared to a packed bed of cylinders of the same number of particles. The increase in simulation time is due to the larger number of particle-particle overlap evaluations performed to complete a packing cycle. Additionally, the subsequent packed bed of the trilobe particle will also be more intricate, thus requiring more computational time to create the mesh.



(a) (b)

Figure 3.8: Packed bed of 500 trilobe particles with bulk porosity of 0.75 (a) Side view of three dimensional geometric representation of packed bed of trilobe particles constructed in GAMBIT meshing software; (b) Radial porosity distribution of the trilobe particle bed.

3.3 Strategy for Mesh Generation

One of the most important aspects of this research project was to develop a procedure for generating meshes in which CFD calculations could be performed efficiently in parallel. With that goal in mind, this section will outline the steps to produce high quality decomposed meshes using GAMBIT 2.4, TGRID 13.0.10, Fluent 13, and the OpenFOAM mesh decomposition utility decomposePar. The overall method to produce the computational meshes involves generating an underlying domain from the results of the packing algorithm; determining an initial face mesh from which the internal volume mesh will connect; generate a Delaunay tetrahedral volume mesh that occupies the interstitial space; converting the pure Delaunay triangulated tetrahedral meshes to an arbitrary polyhedral mesh; and finally, decomposing the mesh into separate sub-domains such that parallel solutions can be computed using a distributed computing approach.

3.3.1 Generating the Underlying Particle Geometry

The output from each run of the packing algorithm contains information about the dimensions, location, and orientation of each cylindrical particle. This information is in a form that is familiar to GAMBIT, and is called a journal file. This journal file is a script that guides GAMBIT through a sequence of steps to draw, rotate, and translate each cylindrical particle of specific dimensions to its final location determined by the aforementioned packing algorithm. For example, a cylinder of radius 0.98 and length 0.98 that is initially aligned along the z-axis, located at the coordinate location (0.76 1.72 12.27), and rotated 66.25° around the x-axis, 69.96° around the y-axis and 2.96° around the z-axis about the centroid of the particle can be achieved by using:

```
/---------drawing particle 1 ----------
volume create height 0.98 radius1 0.98 radius2 0.98 radius3 0.98 zaxis frustum
volume move "volume.1" offset 0.76 1.72 12.27
volume move "volume.1" dangle 66.25 vector 1 0 0 origin 0.76 1.72 12.27
volume move "volume.1" dangle 69.96 vector 0 1 0 origin 0.76 1.72 12.27
volume move "volume.1" dangle 2.96 vector 0 0 1 origin 0.76 1.72 12.27
```

This is performed on every particle in the packed bed, yielding the location of each particle face that will ultimately be meshed. At this point the orientation of each particle can be visualized and inspected. Figure 3.9 illustrates the typical result from the packing algorithm, creating an underlying geometry in GAMBIT that will be meshed for CFD.

In order to produce a mesh, **the particle sizes must be reduced by at most two-percent in order to even produce a mesh**. If the particles are in contact, the meshing algorithm will fail or will require overly fine meshes at contact points (where fluid velocities are low and a coarse mesh is desired). This has been addressed by Nijemeisland et al. and has shown that particle spacing has little effect on the fluid flow near the particle during a CFD fluid simulation [94]. Once the location of the particles is determined, the interstitial space is defined by subtracting each individual particle from a larger cylinder encompassing the packed bed that represents the wall of the tubular reactor. At this point, the overall mesh is reduced even further in order

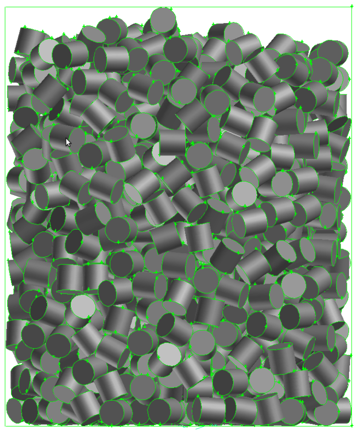


Figure 3.9: Side view of three dimensional domain of 1000 packed cylinders with bulk porosity of 0.65 constructed using GAMBIT meshing software.

to decrease the computational efforts to a level that can be run on a particular cluster or workstation. To achieve this, a representative domain is created by first, lowering the height of the interstitial domain and then, extracting an azimuthal section that is usually one-quarter or one-third in the theta direction, resembling Figure 3.10.

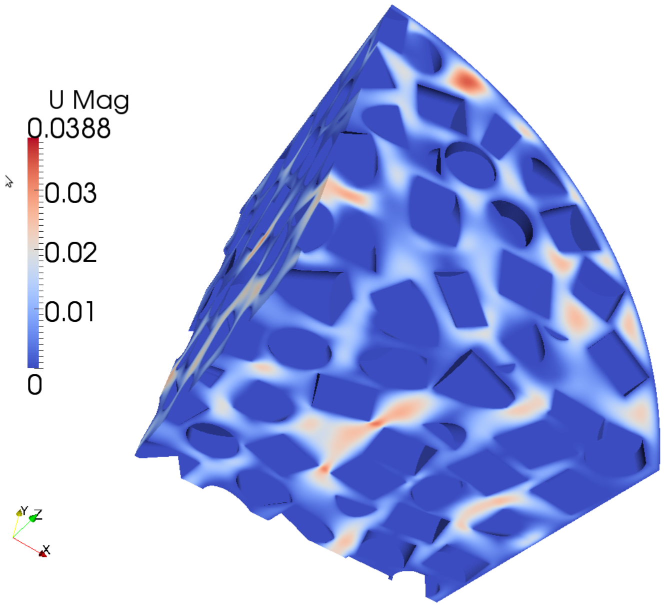


Figure 3.10: Representative domain of interstitial spaces that has been meshed and had a simulation performed.

Because we want to capture the momentum boundary layer, the mesh must possess high grid density near the particle surfaces. To achieve this, an additional step was used to take our already reduced mesh and cut the domain in half for eventual "mirroring", shown in Figure 3.11. This mirroring will only require that half the domain go through the meshing process, thus reducing computer memory usage during the meshing and polyhedral conversion process. The final mirroring step requires joining the reflection of a complete mesh through a plane in the existing mesh.

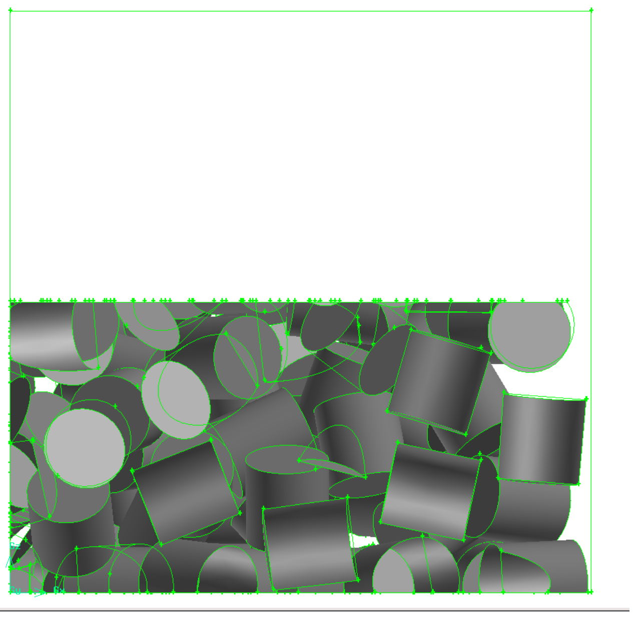


Figure 3.11: Example of a reduced domain prior to face and volume meshing.

3.3.2 Determining the Initial Face Mesh

One of the most important lessons learned in this project was that in order to create a high-quality arbitrary polyhedral mesh, a high-quality tetrahedral mesh is required. To achieve this, tetrahedron cells must posses low skew such that the line connecting two adjacent cell centroids passes through the face centroid of the common face between the adjacent cells. Highly skewed meshes will result in discretization errors that were briefly outlined in Chapter 2. As a rule of thumb for this project, a triangular element face mesh size was chosen to be about 1.5 percent of the particle diameter. To initialize, all faces (particles, inlet, outlet, symmetry plans, and tube) were selected and meshed with triangular elements. The resulting face mesh in GAMBIT was exported in .msh format and carried over to TGRID to generate a tetrahedral mesh, discussed in the next section.

3.3.3 Generating the Tetrahedral Volume Mesh

Though GAMBIT is capable of producing tetrahedral cells to fit extremely complex domains with an unstructured mesh, we need to generate our mesh cells in a particular manner in order to convert them to arbitrary polyhedral cells. GAMBIT uses an advancing front algorithm (See Figure 2.3 for a breakdown of meshing algorithms), starting from an existing triangulated face mesh and continually adding layers to fill a domain [95]. Unfortunately this is not sufficient for our purposes, since high quality arbitrary polyhedral meshes require Delaunay triangulated tetrahedral cells as a precursor.

A Delaunay triangulated tetrahedral mesh follows the "empty sphere" property so that any node must not be contained within the circumsphere of any tetrahedra cell in the mesh [96]. A circumsphere is further defined as the sphere in which its edge passes through all four vertices of the tetrahedron cell. This property can be illustrated quite easily in two-dimensions and is shown in Figure 3.12. Since all of the nodes reside on a circumcircle in Figure 3.12.a, the Delaunay criterion is maintained. In Figure 3.12.b, neighboring triangle nodes are inside another triangle's circumcircle, violating the Delaunay criterion. Delaunay violations can be removed by adjusting the connectivity as shown in Figure 3.12 (a) and (b), and should be performed on all face meshes prior to the volume meshing step.

Once a triangular mesh has been created on all faces, the volume mesh must be generated to fill the domain. In general, the cells near the particle surface are extremely fine in order to account for the boundary layer. To accomplish this, a geometric growth function from all boundaries is used with a growth rate of 1.2. It is typical in meshing these domains that anywhere from 9 million to as many as 24 million tetrahedral cells for our half domain are generated in this step. Subsequently, the entire mirrored domain could have on the order of 20 to 50 million tetrahedral cells. Once a volume mesh has been created, all Delaunay violations are removed through the process of cell splitting to create a Delaunay tetrahedral mesh[96]. As an additional

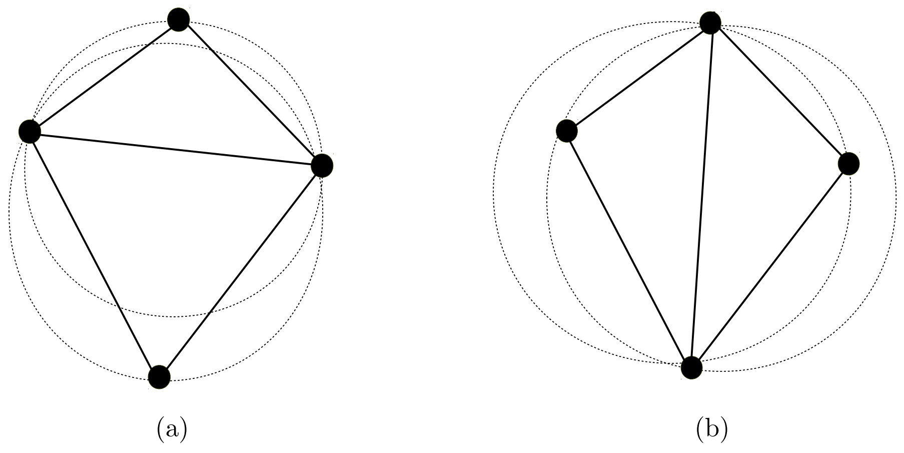


Figure 3.12: Example of a triangular element that is (a) maintains Delaunay criterion; and (b) violates Delaunay criterion.

step, the mesh can be smoothed and Delaunay violations re-checked and removed if necessary.

The final step in the tetrahedral mesh generation step is to assess the mesh quality and correct any remaining problems. Mesh quality is partially assessed using cell skewness, defined numerically using the volume deviation method (for triangular faces and tetrahedral cells) as:

$$\text{Cell Skew} = \frac{\text{optimal cell size} - \text{actual cell size}}{\text{optimal cell size}}.$$

In general, a histogram of mesh cell skewness is determined in order to judge the overall quality of a mesh. As a rule of thumb, mesh quality can be assessed using Table 3.1. If there is at least one cell with a skew greater than 0.95, re-mesh the entire domain. The process of building a mesh is an iterative process, continually adjusting

Table 3.1: Comparison of Relative Mesh Cell Quality to cell Skewness

Skewness	0 - 0.25	0.25 - 0.5	0.5 - 0.8	0.8 - 0.95	0.95 - 0.99	0.99 - 1.00
Cell Quality	Excellent	Good	Acceptable	Poor	Sliver	Degenerate

face cell size, surface mesh growth rate, removing Delaunay violations, smoothing a mesh, assessing mesh quality and re-meshing. Once a relatively fine, Delaunay mesh with low cell skew has been created, it is converted to an arbitrary polyhedral mesh.

3.3.4 Polyhedral Mesh Conversion

In order to reduce the cost of some of the calculations the tetrahedron meshes were converted to an arbitrary polyhedral mesh. For clarity, a typical cell in an pure polyhedron mesh has 10-14 faces [101] with the general shape shown in Figure 2.1. In OpenFOAM, the utility polyDualMesh is used to convert the tetrahedron cells to polyhedron while leaving the hexahedron cells unconverted. The starting mesh must be three-dimensional (possess a depth component) and be created using a Delauney tetrahedron mesh generation [96, 95] algorithm. In general, the algorithm for converting a tetrahedral mesh to an arbitrary polyhedral mesh requires several steps.

1. Decompose tetrahedron cell about each tet node at each cell. During the decomposition, the faces of the new polyhedron are defined using quadrilateral area elements surrounding the tet nodes.
2. Collect the quad faces of the polyhedron cell and close the cell around a center point to create a medium dual.
3. Selectively join the quad faces surrounding a polyhedron cell into a smaller number of faces.
4. Move the center point of the polyhedron cell to a new centroid and the new cell is defined at this new centroid.

A more detailed explanation of this method can be pursued in a presentation by Kelecy [70]. What is important to note is that tetrahedral cells are combined, faces are moved and combined to optimal positions, and a polyhedron cell is created with more faces around a cell center to yield an improved mesh. By having an "improved" mesh, equation discretization will yield better results and improve stability of the solution method.

3.3.5 Parallel Mesh Decomposition

In order to run an efficient parallel computation, all of the nodes (e.g. CPU Processors) must be performing relatively the same amount of work. To achieve a balanced load across the nodes, the mesh must be decomposed to minimize the faces of the processor-processor boundaries and evenly split the mesh into sub-meshes so that each node has relatively the same number of cells to compute. This is achieved using Scotch (http://www.labri.fr/perso/pelegrin/scotch/), a library for for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning. To our advantage, the Scotch library has been integrated into OpenFOAM such that it can be invoked by using the application `decomposePar` . For the packed bed domains in this project, Figure 3.13 shows a typical domain decomposition where each color represents a different processor. If a mesh is decomposed properly, the parallel execution will be more efficient and scale better to larger clusters.

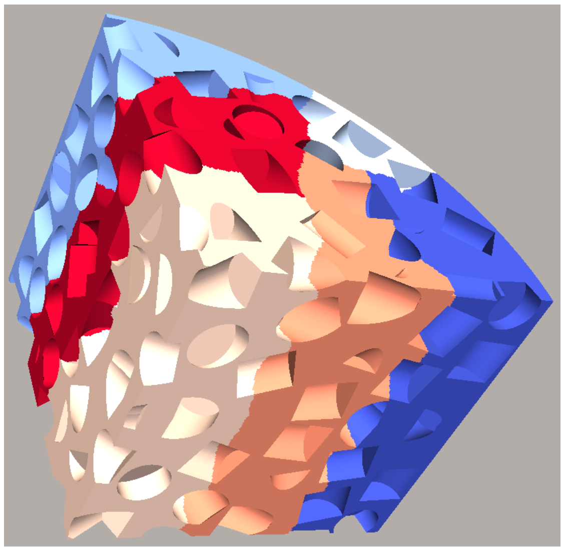


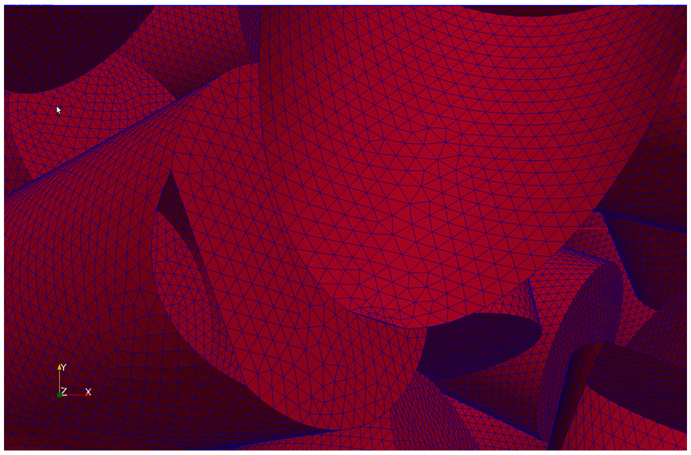
Figure 3.13: Typical parallel processor distribution on a packed-bed mesh

3.3.6 Results and Discussion

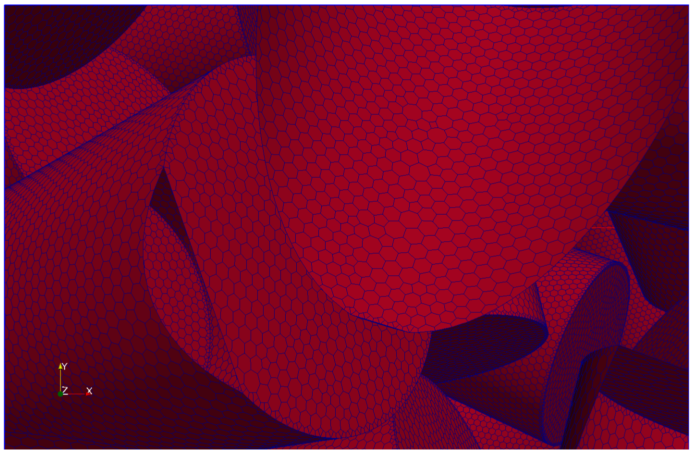
It was found that the conversion of the pure tetrahedral meshes 1) reduced the number of cells and faces in the mesh; 2) improved the mesh structure and quality; and 3) eventually enabled a faster calculation. Prior to the conversion of the Delaunay tetrahedral cells to arbitrary polyhedral cells, the entire mesh was composed of 32,506,136 tetrahedral cells and 65,750,700 triangular faces (i.e. 16,253,068 cells and 32,875,350 faces for each half before mirroring). After conversion, the entire mesh was composed of 6,163,984 cells and 41,268,924 faces, an 81 percent reduction in cell count and 37 percent reduction in face count. The composition of the arbitrary polyhedral mesh was 5.7 percent hexahedra (356,314 cells) and 94.3 percent (5,807,670 cells) complex polyhedra. In addition the conversion produced a higher quality mesh so that all OpenFOAM mesh quality (e.g. skewness, aspect ratio, orthogonality, etc.) and topology (e.g. connectivity, shape, etc.) checks passed. The improvements can be observed visually in Figure 3.14, showing the removal of tetrahedra grain boundaries over the particle surface. Lastly, because the reduced cell and face count inherently produces a smaller $A\mathbf{x} = \mathbf{b}$ system, it will always be less time per iteration in our linear system solver.

As an additional step to improve the speed of our calculation, we can reorder the sparse matrix so that the number of diagonal bands are reduced. In practice, the reordering increases memory access efficiency and generally increases the speed of the inner iterations (i.e. linear system solver iterations). Using the `renumberMesh` OpenFOAM utility, the number of bands in our mesh were reduced by 97.8 percent from 3,081,992 bands to 38,461.

For the parallel computation, Scotch was used to decompose the mesh over 6 processors with the results of the decomposition in Table 3.2. On average, each processor performed calculations on nearly 1 million cells, with a rule of thumb being 50k cells-per-processor a more desirable ratio. No study was performed to see the effect of decreasing the cells-per-processor (i.e. using more computational nodes). Also, the number of processor patches was 5 for each domain, meaning that each processor must communicate and exchange data with 5 other processors during the calculation. Though this is seemingly a low number and OpenFOAM is fairly well optimized for parallel calculations, it is still undesirable to have so many processor patches. Lastly, the SEAS cloud (`http://cloud.seas.wustl.edu/`) was tested as a viable option to using a workstation. The larger number of computational nodes was beneficial to submit many small jobs or larger jobs that had no upper time limit, however, it



(a)



(b)

Figure 3.14: Example of the surface mesh on a particle (a) full triangular faces prior to arbitrary polyhedral conversion; and (b) polyhedral faces that smoothed out remaining grain boundaries in created triangular mesh.

was found to perform at a lower computational efficiency than the current workstation. This was primarily due to inefficient node communication hardware, since many message passing intensive programs are not regularly run on the cloud (i.e. it was not designed for this type of computation). The domains decomposed into 10, 16, 32, and 64 processors were all slower than a single workstation decomposed into 8 sub-domains.

3.4 Closure

Geometry generation using a Monte-Carlo packing algorithm

A Monte-Carlo packing algorithm using a sorting method is presented as an effective method to produce loosely packed bed domains of cylindrical and trilobed particles. The particle-particle overlap determination is achieved using a method proposed by Blaak et Al.[18], based on approximating a cylinder with spheres, spherocylinders, and ellipsoids. The bulk porosities of approximately 0.65 are achieved by this method with relatively few packing cycles due largely to the use of a sorted list of particles. The sorting of the particles allowed for particles located closer to the bottom of the container to be packed before particles located higher in the bed. The increase in space created greater degrees of freedom for particles to achieve a more dense packing in less packing cycles.

The radial porosity distributions produced by the packing algorithm are qualitatively comparable to experimental work by Roblee et al.[110], showing similar wall effects observed experimentally. The main difference being that the experimental results by Roblee showed an extremely densely packed bed of particles near the wall and throughout the bed that is considerably more dense than those seen in CT and Digi-DEM reported in the literature[21]. This is due mainly to external forces (gravity and manual compression) during the physical packing process in the experimental work. The Monte-Carlo algorithm presented here uses only random motion and a lowering of the top boundary to reduce the degrees of freedom to form the packed bed of particles. For a more densely packed bed, there must be either more packing

Table 3.2: Mesh summary of Scotch decomposed mesh

Processor #	# of Cells	# of Processor Patches
0	1,028,789	5
1	1,025,616	5
2	1,020,067	5
3	1,017,494	5
4	1,039,664	5
5	1,032,354	5
Average	**1,027,330.66**	**5**

cycles or an algorithm incorporating a force field similar to ones used in molecular dynamics simulations. The Monte-Carlo packing algorithm proposed in this paper tracks the location and orientation of each face of the particles being packed. Knowing the location and orientation of each particle face facilitates the meshing process as there is no need for more preprocessing to extract the edges and faces from a pixilated image produced using the method by Gan et al.[47] and Caulkin et al.[21]. Ultimately, this image extraction process may introduce more approximations into the already extremely intricate structures seen in packed beds of particles.

The resulting computational meshes fully define the location of the particle faces within a domain of randomly packed cylinder based particles. In terms of interstitial CFD modeling, knowing the exact location of the particle faces allows for extremely fine boundary layer meshes to be used to increase resolution of near particle modeling of transport phenomena. Ultimately, meshes accurately describing the microstructure of packed beds enable more realistic CFD simulations, providing further insight into the fluid dynamics on the length scale of the interstitial spaces between particles. As a result, a deeper understanding of fluid flows within packed beds can be achieved and will ultimately improve catalyst shape optimization, current models describing reactors, and unit operations leveraging the intricate structure of packed beds.

Mesh Generation

The mesh generation portion of the project was a time-consuming task that involved numerous meshes (almost 50 gigabytes of data), with solutions being attempted on most of the created meshes. The method developed to create a high quality mesh can be summarized in the following steps:

1. Create the underlying domain in GAMBIT using the output journal file from the Mote-Carlo packing algorithm
2. Create an initial face mesh in GAMBIT and export the mesh as .msh format
3. Import the .msh file into TGRID, and remove any Delaunay violating triangular faces

4. Create the pure tetrahedral mesh using growth functions and remove any Delaunay violations. Smooth the volume mesh and assess the quality, re-meshing if necessary.

5. Convert the pure tetrahedral mesh into an arbitrary polyhedral mesh using ANSYS Fluent 13.

6. Import the mesh into OpenFOAM and then decompose the domain according to the number of parallel nodes to be used in the calculation

For the parallel computations, the mesh was converted to an arbitrary polyhedral mesh to reduce the cell count by 81 percent and the face count by 37 percent. Additionally, the renumbering of the mesh was used to increase memory access efficiency, by reducing the number of diagonal bands by almost 98 percent. Using the Scotch library was instrumental in decomposing the mesh into separate sub-domains with nearly equal cell and processor patch counts. It can be said with confidence, that the combination of care in creating the mesh and careful planning to decompose the mesh reduced the overall solution time significantly. If more projects are pursued that will leverage OpenFOAM, it is prudent that more investment be made to improve our local cluster or the SEAS cloud.

Chapter 4

Interstitial-Scale Momentum Transport Modeling

4.1 Introduction

The necessary background covering the finite volume method, Monte-Carlo packing algorithm, and mesh generation procedure was given in Chapters 2 and 3. In this chapter, the interstitial-scale momentum transport modeling of packed-beds is discussed, along with an analysis of the resulting laminar and turbulent velocity fields. Beginning with an introduction to the phenomena of turbulence, the methods for solving the Navier-Stokes Equations under laminar and turbulent conditions are covered in detail. More precisely, the Reynolds averaged Navier-Stokes (RANS) model is mentioned, with a special emphasis given to the low Reynolds number Lam-Bremhorst two-equation k-ϵ model as a method to close the Reynolds stress term ($\langle u'v' \rangle$) appearing in the RANS model. Finally, a detailed analysis of the three-dimensional flow field; the classical radial profiles within the bed; and a newly developed post-processing method to contract complex data into a one-dimensional perpendicular profile is given. Ultimately, the current chapter will provide the necessary discussion for the next chapter on interstitial-scale scalar transport in packed-beds.

4.2 Momentum Transport Modeling

Packed-bed systems are composed of complicated structures of spaces between randomly packed particles. From experience and dimensional analysis, it is known that the size of the random interstitial spaces, fluid viscosity, and inlet velocity are important parameters influencing the overall pressure drop in the bed and the flow regime of the fluid. For simplicity, the influence of these three parameters are encompassed into a particle Reynolds number that has been previously discussed by Gunjal et. al. [54], and defined as

$$Re_p = \frac{d_p U_0}{\nu}. \tag{4.1}$$

For this specific definition of Reynolds number, the characteristic length is the particle diameter (d_p), the characteristic velocity (U_0) is defined as the average interstitial velocity within the packed-bed, and the fluid kinematic viscosity for air is used throughout. according to this specific formulation, creeping laminar flow is assured for flows less than $Re_p = 0.1$. Gunjal further notes that based on the work of Jolls and Hanratty [66], the transition between laminar and turbulent flow occurs at $Re_p \approx (300 - 400)$. As a result, this project assumed laminar flow under $Re_p < 150$ and fully developed chaotic turbulent flow was present at $Re_p > 1000$. Such clear definitions of flow regimes provides a roadmap for choosing a specific method to solve the incompressible Navier-Stokes equation.

The incompressible Navier-Stokes is a model that is valid a fluid traveling much slower than the speed of sound and is assumed to sufficiently describe the flow within packed-bed systems. Furthermore, it was also assumed that the fluid is completely characterized by the three components of velocity ($\mathbf{U}_x$, $\mathbf{U}_y$, and $\mathbf{U}_z$) and the value of pressure at each moment in space and time [80]. Lastly, it is assumed that the fluid obeys the continuum hypothesis and can be described fully by the continuity equation

$$\nabla \cdot \mathbf{U} = 0 \tag{4.2}$$

and the Navier-Stokes Equations

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{U} + \mathbf{g}. \tag{4.3}$$

If the computational mesh is fine enough, Equations 4.2 and 4.3 will completely resolve the fluid flow in both laminar and turbulent flows via direct numerical simulation (DNS). For turbulent fluids in packed-beds, the mesh refinement requirements make DNS infeasible from a computational point of view with our current computing capability at Washington University. Using a modeling method that adequately characterizes a turbulent fluid flow at a low computational cost is a major thrust of CFD. Several methods exist to capture the effects of turbulence including Large Eddie Simulation (LES), Lattice Boltzmann Simulations (LBS), and the least computationally intensive Reynolds-Averaged Navier-Stokes (RANS) method. An explanation of the phenomena of turbulence, along with an introduction to using the RANS equations to determine the first moment of velocity $\langle \mathbf{U} \rangle$, are briefly covered in the next section.

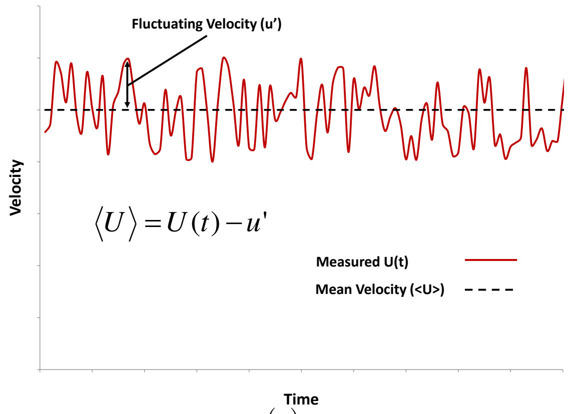
4.2.1 Turbulent Momentum Transport Theory

It is well known that there are two major flow regimes, laminar and turbulent, that are separated by a less distinct transitional flow regime. Laminar flow is characterized by fluid elements moving in regular paths with parallel path-lines, lateral mixing by molecular diffusion only, and little to no swirling or eddy motion. Laminar flow can be complex and remain non-turbulent while flowing through complicated domains. At very low flow rates, viscous forces are dominant compared to convective forces. Because viscous interactions dissipate energy input to the fluid system[8], if the local velocity is sufficiently high, convective forces can dominate and create a situation where repeated measurement of a velocity at a particular point under identical conditions will yield a different value [11]. Under these circumstances, if the random character of the fluid is sustained at the same conditions, the fluid is said to be turbulent. If the fluid is not sustained under these conditions, it is said to be in a transition state

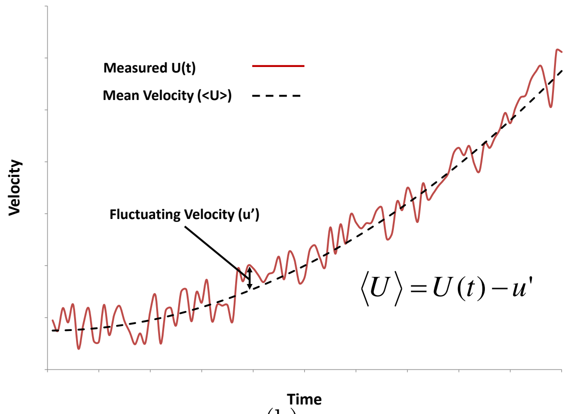
[8]This decay of random velocities is due to viscous dissipation of the fluctuations, a cornerstone of Kolmogorov's theory [87].

between laminar and turbulent flow in which regions of turbulent perturbation are dissipated back to complex laminar flow.

For a fully turbulent fluid, the values of velocity of each point in space are highly chaotic. If at constant average boundary inputs (e.g. constant flow rate), each component of the random velocity field fluctuates around a time-averaged mean velocity vector component ($\langle U_i \rangle$) similar to the representation in Figure 4.1.(a), it can be considered to be at "steady-state". Though turbulent flow is inherently chaotic and transient in nature, the observation of the fluctuating velocity about a mean enables the modeling of the first moment of the random velocity field at very long time-scales. If transient boundary inputs are present in the system, the interpretation of the fluctuating random velocity about a mean becomes an ensemble averaged random velocity[9] field moving in time as in Figure 4.1.(b). Rather than attempting to capture the full instantaneous random nature of the fully turbulent flow, the time and ensemble averaged flow field is modeled using the Reynolds averaged Navier-Stokes model.



(a)



(b)

Figure 4.1: Random point velocity fluctuations for (a) steady-state turbulent flow and (b) transient turbulent flow

In the Reynolds Averaged Navier-Stokes (RANS) equations, turbulent momentum conservation is described using the Navier-Stokes equations coupled with a perturbation in the velocity and pressure variables[17]. The approach uses a time and space

[9]The ergodic nature of the random fluctuations of turbulence requires time and ensemble averaging to coincide.

dependent velocity variable $\mathbf{U}(\mathbf{x}, t)$ decomposed into a time or ensemble averaged velocity $\langle \mathbf{U} \rangle$ and a fluctuating velocity $\mathbf{u}'$ component. Termed as Reynolds decomposition, the random velocity field is formulated such that

$$\mathbf{U}(\mathbf{x}, t) = \langle \mathbf{U}(\mathbf{x}, t) \rangle + \mathbf{u}'(\mathbf{x}, t). \tag{4.4}$$

Similarly, the pressure can be represented in terms of the Reynolds decomposition of the original variable $p(\mathbf{x}, t)$ so that

$$p(\mathbf{x}, t) = \langle p(\mathbf{x}, t) \rangle + p'(\mathbf{x}, t). \tag{4.5}$$

By substituting 4.4 and 4.5 into the Navier-Stokes equations (Equation 4.3) and then Reynolds Averaging each differential term (See Appendix A.3), the continuity equation becomes

$$\nabla \cdot (\langle \mathbf{U} \rangle + \mathbf{u}') = 0, \tag{4.6}$$

and momentum conservation described by Navier-Stokes becomes

$$\frac{\partial \langle U_i \rangle}{\partial t} + \langle U_j \rangle \frac{\partial \langle U_i \rangle}{\partial x_j} = \nu \nabla^2 \langle U_i \rangle - \frac{1}{\rho} \frac{\partial}{\partial x_j} \left\langle u_i' u_j' \right\rangle - \frac{1}{\rho} \frac{\partial \langle p \rangle}{\partial x_j}. \tag{4.7}$$

In 4.6 and 4.7, $\langle U \rangle$ and u' are solenoidal vector fields[105, 6], ν is the molecular kinematic viscosity, and ρ the fluid density. Equation 4.7 is often referred to as the unsteady Reynolds averaged Navier-Stokes equation or URANS. For steady-state turbulent flows, the time derivative in Equation 4.7 is zero and the remaining equation is referred to as the Reynolds averaged Navier-Stokes or RANS equation. An important observation is that Equation 4.7 presents a Reynolds stress term $\rho \left\langle u_i' u_j' \right\rangle$ as a cross correlation of u_i' and u_j', leaving more unknowns than equations. This is referred to as the turbulence closure problem.

To deal with the issue of closure, the problem can be approached in several different ways, three of which are presented in this discussion. The first method is to approximate the unclosed $\left\langle u_i' u_j' \right\rangle$ term with the Boussinesq eddy viscosity hypothesis[19], given as[104]

$$-\rho\left\langle u_i' u_j'\right\rangle = \mu_t\left(\frac{\partial\left\langle U_i\right\rangle}{\partial x_j} + \frac{\partial\left\langle U_j\right\rangle}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\left(\mu_t\frac{\partial\left\langle U_k\right\rangle}{\partial x_k} + \rho k\right), \tag{4.8}$$

where k is the turbulent kinetic energy equal to $u_i' u_i'/2$, δ_{ij} is the Kronecker delta[17], and μ_t is the turbulent viscosity scalar determined by a turbulent viscosity model. Over the last century, numerous turbulent viscosity models have been devised including $k-\epsilon$, $k-\omega$, Spalart-Allmaras, and RNG $k-\epsilon$ [133]. These models, along with numerous other formulations, are covered in detail in many books on fluid dynamics and computational continuum mechanics[104, 44].

The second method for determining the unclosed stress $\rho\left\langle u_i' u_j'\right\rangle$, involves modeling the components of the Reynolds stress tensor with transport equations. The model derivation for the Reynolds stresses transport equations is thoroughly covered in many books on computational fluid dynamics[104, 133] and is removed from the current discussion. The final equation[45] of the transport of Reynolds stresses is expressed as

$$\frac{\partial\left\langle u_i' u_j'\right\rangle}{\partial t} + \left\langle U_k\right\rangle\frac{\partial\left\langle u_i' u_j'\right\rangle}{\partial x_k} + \frac{\partial\left\langle u_i' u_j' u_k'\right\rangle}{\partial x_k} = P_{ij} + \Pi_{ij} + \nu\nabla\left\langle u_i' u_j'\right\rangle - \epsilon_{ij}, \tag{4.9}$$

with the production term

$$P_{ij} = -\left\langle u_i' u_k'\right\rangle\frac{\partial\left\langle U_j\right\rangle}{\partial x_k} + \left\langle u_j' u_k'\right\rangle\frac{\partial\left\langle U_i\right\rangle}{\partial x_k}, \tag{4.10}$$

the velocity-pressure-gradient term

$$\Pi_{ij} = -\frac{1}{\rho}\left\langle u_i'\frac{\partial p'}{\partial x_j} + u_j'\frac{\partial p'}{\partial x_i}\right\rangle, \tag{4.11}$$

and the turbulent dissipation term

$$\epsilon_{ij} = 2\nu\left\langle\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\right\rangle. \tag{4.12}$$

The triple-correlation term $\left\langle u_i' u_j' u_k' \right\rangle$ is closed with an assumption similar to the Boussinesq eddy viscosity hypothesis.

An additional approach used to close the Reynolds stress term are algebraic Reynolds stress models (ASMs). These types of models assume a nonlinear constitutive model to relate the Reynolds stresses and the rate of mean strain rather than additional transport equations for each of the Reynolds stresses[104]. These relationships are held to physical and mathematical constraints including Galilean invariance and "realizability" to ensure physically meaningful results. Lastly, the basis of ASMs require that adjustments be made to adequately predict turbulent flow and generally do not perform as well as a full Reynolds-stress model.

Given sufficient detail into the phenomena of turbulence and a method to model the first moment of the random velocity field, further details into how to apply the theory in order to solve the RANS equations must be introduced. In the next section the methodology used to model the complex fluid flow within packed-beds is provided for steady-state RANS models using a two equation $k - \epsilon$ variant. The algorithm for decoupling the components of the velocity field and pressure are briefly addressed and a discussion of the specific boundary conditions over the domain is given for clarity.

4.2.2 Momentum Modeling Methodology

Pressure Velocity Coupling and the RANS Equation

Due to the nonlinearities in the Navier-stokes equations, instabilities due to turbulence, and the complex geometry of the packed-bed, an implicit strategy was used to solve the set of partial differential equations. Specifically, a segregated approach in which each velocity and pressure component was solved separately, with coupling between equations treated using the "current" variable value. This is not a new concept, as it was very thoroughly discussed by Patankar [99, 98] more than 30 years ago and more recently by Jasak [61]. Through this method of decoupling, an accurate solution to the nonlinear coupled equations with fast convergence and a high degree of stability can be achieved

For steady problems, the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm is used to solve the steady-state Navier-Stokes equation and RANS model presented in the previous section. The SIMPLE algorithm uses a segregated solver for all of the components (U_x, U_y, U_z, and P), while solving and correcting the velocity in pressure-solving sub-iterations. While the details of this algorithm are important for this discussion, it is left for the motivated reader to follow in the work by Jasak [61]. As a more cursory explanation, the algorithm begins with:

1. Set each field U_x, U_y, U_z, P and the appropriate turbulence model parameters (e.g. k and ϵ) with the proper boundary conditions that are outlined later in this section.

2. Stepping into the outer-iteration (pseudo-time steps for steady-state calculations), collect the terms in the Navier-Stokes equation containing a velocity component. For the convection term

$$\nabla \bullet (\rho \mathbf{U}\mathbf{U}),$$

   $\rho\mathbf{U}$ is treated explicitly and is based on the previous iteration (or initial guess). Because we are using finite volume discretization, this implicit treatment of $\rho\mathbf{U}$ is utilized as the face-flux term F_f in the surface integral for the advection term defined by Equation 2.4. In the OpenFOAM code, the aforementioned F_f term is a surface scalar field `phi` that is not to be confused with the ϕ_f in Equation 2.4. Lastly, the Laplacian term is defined implicitly using Equation 2.21 and need not be further linearized since it is already linear.

3. Using the current value of the pressure field, the gradient of pressure is calculated. At this point, each velocity component is under-relaxed (Equation 2.28) and solved implicitly using the current value of the pressure gradient. Often this step is referred to as the "momentum-predictor" step.

4. At this point, we have an initial guess for the velocity field from our momentum-predictor step. The pressure Poisson equation is solved and the fluxes at the face are corrected from the pressure field. The pressure Poisson equation and new corrected fluxes are determined in such a way to ensure continuity is satisfied through the process of Rhie-Chow interpolation [61, 67]. In an effort to

determine a better approximation of the pressure field, the pressure equation can be solved again.

5. Once a pressure field has been calculated and corrected in the pressure-corrector loop, the pressure field is relaxed and then used to correct the velocity field for use in the next outer-iteration.

6. At this point in the solution algorithm, the turbulence model is solved using the current values of velocity and pressure. In the OpenFOAM code, this is represented in the term `turbulence->correct();` after the pressure-velocity SIMPLE corrector loop. Convergence is checked and this marks the end of an outer-iteration.

7. The loop over out-iterations is continued until the desired convergence is reached, with appropriate data written at user specified data intervals.

The details of the solver are left for those interested in looking directly at the code (e.g. simpleFOAM), however the overall steps in the algorithm have been presented. What remains is an explanation (and justification) of the specific turbulence model used in this study. For RANS models, there are many turbulent viscosity closure models. For reasons outlined in the next section, the Lam-Bremhorst model was chosen to capture the near wall (particle and tube) turbulent boundary layers.

The Lam-Bremhorst k-ϵ Model

As noted by Wilcox, turbulence is a continuum phenomena in which the length scale of the smallest turbulent eddy (i.e. the Kolmogorov length) is orders of magnitude above molecular scales [133]. Furthermore, due to the random nature of the fluid motion, we are limited in our pursuit of an exact representation[10] to seeking a statistical representation of the flow using the RANS model. The key to closing the RANS model is to choose an appropriate turbulence model, and in the case of packed-beds one that is formulated for relatively low Reynolds number flows.

[10]Though we can use direct numerical simulation (DNS) to get an "exact" flow field for velocity, it is neither practical nor possible in most research laboratories due to lack of computational resources.

The choice of low-Reynolds number turbulence models is vast, yet one that is trustworthy, easy to use, and readily available is deemed the best. In OpenFOAM there are several low-Reynolds number turbulent viscosity models including the Lam-Bremhorst, Launder-Sharma, Q-Zeta, Lien-Cubic, Lien-Leschziner and Spalart-Allmaras. Although it was previously noted in Table 1.1 that the Spalart-Allmaras model was used by this project. It was found that faster convergence could be met with a $k-\epsilon$ model.

For the standard $k-\epsilon$ model, near wall interactions are handled via wall-functions based on the "law of the wall" so that the near wall velocity profiles can be approximated with functions based on experimental results. For wall bounded flows, variants of the standard $k-\epsilon$ are formulated such that no wall functions are used and that the transport equation "knows" where the closest wall is in relation to the current position being calculated. As a result, these variants match much closer (both in the bulk and near the wall) to DNS results compared to the standard $k-\epsilon$ for even simple wall bounded Couette flows [114]. Because these low Reynolds number models can accurately predict the near-wall dissipation, the Lam-Bremhorst (LB) $k-\epsilon$ model [77] was chosen for this study.

The LB model is particularly desirable because of its performance with relatively course near wall meshes, a primary concern for our existing complex meshes with high cell count. Specifically, the LB model is designed to work well with $y^+ \approx [1, 12]$, where the dimensionless wall distance y^+ is defined as [133]

$$y+ = \frac{y\sqrt{\nu||\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}\mathbf{x}} \cdot \mathbf{n}||}}{\nu}, \tag{4.13}$$

with y and $\mathbf{n}$ defined as the perpendicular distance to the nearest wall and nearest wall surface normal vector respectively. Dimensionless wall distance merely serves as a method to indicate how much of the boundary layer is being captured by the cell closest to the wall. A higher y^+ indicates a courser mesh that is not capturing the transition of turbulent to laminar flow near the wall. The LB model is accurate at even moderate y^+ values and is valid for y^+ up to 12. The specific description of the LB model can be represented as a generalized form of the $k-\epsilon$ model, presented next.

For incompressible flow, a generalized form [133] of the k-ϵ model can be written as

$$\frac{\partial k}{\partial t} + \nabla \bullet (\mathbf{U}k) = \nu_t \nabla^2 \mathbf{U} - \epsilon + \nabla \bullet \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \nabla k \right] \tag{4.14}$$

$$\frac{\partial \tilde{\epsilon}}{\partial t} + \nabla \bullet (\mathbf{U}\tilde{\epsilon}) = C_{\epsilon 1} f_1 \frac{\tilde{\epsilon}}{k} \nu_t \nabla^2 \mathbf{U} - C_{\epsilon 2} f_2 \frac{\tilde{\epsilon}^2}{k} + E + \nabla \bullet \left[\left(\nu + \frac{\nu_t}{\sigma_\epsilon} \right) \nabla \tilde{\epsilon} \right] \tag{4.15}$$

where the turbulent kinetic energy dissipation ϵ is related to $\tilde{\epsilon}$ by

$$\tilde{\epsilon} = \epsilon - \epsilon_0. \tag{4.16}$$

The reference value (ϵ_0) is taken at y = 0 and for the Lam-Bremhorst model is equal to zero. The turbulent viscosity is defined as

$$\nu_t = C_\mu f_\mu \frac{k^2}{\tilde{\epsilon}} \tag{4.17}$$

Further definition of the LB model requires the definition of constants and dampening functions such that

$$f_\mu = \left(1 + \frac{20.5}{Re_T} \right) \left(1 - e^{-0.0165 R_y} \right)^2 \tag{4.18}$$

$$f_1 = 1 + \left(\frac{0.05}{f_\mu} \right)^3 \tag{4.19}$$

$$f_2 = 1 - e^{-Re_T^2} \tag{4.20}$$

$$\epsilon_0 = 0 \tag{4.21}$$

$$E = 0 \tag{4.22}$$

$$C_{\epsilon 1} = 1.44,\ C_{\epsilon 2} = 1.92,\ C_\mu = 0.09,\ \sigma_k = 1.0,\ \sigma_\epsilon = 1.3, \tag{4.23}$$

with

$$Re_T = \frac{k^2}{\tilde{\epsilon}\nu},\ R_y = \frac{k^{1/2}y}{\nu},\ y^+ = \frac{y\sqrt{\frac{\tau_w}{\rho}}}{\nu}. \tag{4.24}$$

Given an explanation of the Lam-Bremhorst model; the background in the methodology to decouple velocity, pressure, and turbulence model; and the theoretical background in the the phenomena of turbulence, the last point of discussion before presenting the results is given next. One of the most influential portions of any model is the choice of boundary conditions. They must be realistic and provide stability for the implemented solution algorithm. The next section will discuss the boundary conditions for velocity, pressure, turbulent kinetic energy (k), and turbulent kinetic energy dissipation (ϵ) used in this project.

Boundary Conditions

In general, a sequence of boundary conditions are used to iteratively condition a set of simulations into a more realistic rendition. In that manner, the boundary conditions for the velocity, pressure, and LB model are briefly addressed.

- **The Mean Velocity Field:** For asymptotic agreement, a no-slip zero-velocity was used at the wall boundaries (i.e. particles and the outer tube surface). No wall-functions were used in this study and the near-wall mesh was constructed such that the boundary layer was resolved to a sufficient degree. At the outlet, to increase stability a no-inflow boundary condition was set. This results in the assumption that if the velocity vector at the boundary is pointing outward with respect the the boundary normal vector, the boundary condition is set as a zero-gradient. If the velocity vector is pointing inward with respect to the boundary normal vector, the boundary condition is set to a fixed value of zero to indicate that there is no mass inflow. This was necessary for numerical stability in flows that show a large amount of swirl at the exit. In OpenFOAM, this is defined using

```
outlet
```

```
        {
            type                inletOutlet;
            inletValue          uniform (0 0 0);
            value               uniform (0 0 0);
        }
```

In the case of the inletOutlet boundary condition in OpenFOAM, the `uniform (0 0 0)` entry is merely an initialization and will change to the corresponding value based on the zeroGradient condition of outflow. Lastly, the inlet condition is the most complicated and requires special consideration. For most all other research in the field of interstitial flow modeling in packed beds, a periodic mesh is used, thus eliminating pressure driven flow since the pressure at the inlet and outlet are identical. Because pressure driven flow is so integral to the flow within packed-beds, a method of mapping the exit condition to the inlet was used. To maintain a mass flow rate through the bed of particles, the distribution was scaled such that a constant average velocity was maintained throughout the simulation. In OpenFOAM (version 2.1.x) this mapping of values from one boundary (patch) to another is defined by

```
    inlet
        {
            type                mapped;
            value               uniform (0 0 1.3);
            interpolationScheme cell;
            setAverage          true;
            average             (0 0 1.3);
        }
```

Where the cell-centered values of the cells at the boundary are mapped to the inlet boundary and set for example to a vector average of 1.3 $\frac{m}{s}$ in the z-direction and zero in the x- and y-directions. The corresponding boundary by which the inlet "knows" is abstracted and controlled in the global definition of the boundary types in the constant/polyMesh/boundary file.

At this point, it is important to note that there is strong coupling between the entrance and exit boundaries that may result in a negative feedback loop. If there is a strong chance of swirl (i.e. creating a velocity vector opposite to the boundary normal), then the inlet outlet patch will assign a velocity component of zero. The resulting "mapped" boundary condition will then map this zero velocity to the inlet boundary and scale the remaining velocity vectors to the predetermined average value. This will ensure that the flow rate through the packed bed will always be constant and converge, but has the potential to amplify the upward directional vectors, while dampening the swirling flow at the exit. It is presented here as a word of caution for those repeating the work.

- **The Mean Pressure Field:** Pressure is a surface force, much like shear, however only acting on the fluid element in the normal direction. Pressure and shear are closely related and oftentimes instabilities, unrealistic behavior, and unphysical values in the turbulent parameters manifest themselves as instabilities in the pressure field (and subsequent velocity field). Because pressure is so closely tied to shear-stress and velocity we must under-relax the values heavily and choose extremely stable boundary conditions that case smaller perturbations in the flow field.

  In most incompressible flow simulations in which a velocity is prescribed at the inlet and a zero-gradient in velocity at the outlet, the inlet boundary condition of pressure is set to be a zero-gradient boundary, meaning that the pressure is determined from the resulting velocity field value in order to maintain continuity at the boundary cells near the inlet. This is standard practice among all finite volume codes and is generally accepted across all fields of engineering utilizing a similar method of decoupling pressure and velocity. In addition, the pressure condition at the particle surface and tube wall was set as a zero-gradient during the process of solving the pressure Poisson equation.

  In order to maintain stability and satisfy the continuity equation, a fixed value boundary condition for pressure is applied at the outlet and usually set to a gauge pressure (absolute pressure minus atmospheric pressure) of zero. Because we are defining our pressure field in gauge pressure space, negative pressures merely mean we are determining a value less than atmospheric pressure and not below absolute zero pressure. In some cases of extremely stiff systems, we may opt for a more subtle "pseudo" fixed value boundary condition in which we gradually change the pressure value at the outlet to converge to a prescribed fixed mean value. This pseudo-fixed value boundary condition produced a more stable –especially eliminating constantly rising inlet pressure– solution progression towards eventual convergence.

- **The Mean Turbulent Kinetic Energy Field (k) and turbulent dissipation rate (ϵ):** The Lam-Bremhorst model is designed to predict the effects of dissipation in the viscous sublayer and therefore does not require a wall function like the standard $k - \epsilon$ model. The prediction of the phenomena within the near-wall region is completely left to the model. Because of this, the wall boundary conditions for both k and ϵ are set to a zero-gradient. From a physical perspective, this boundary condition is meaningful as we only require that the production of k and ϵ be zero in the laminar sublayer. Just above the laminar sublayer, we expect the an interface of laminar and turbulent flow; with large shear that produces "turbulence". The generated k and ϵ are then diffused into the laminar subregion and dissipated, a analysis that agrees with Kolmogorov's hypothesis and the cascade of turbulent energy [133]. For the inlet values of k and ϵ, several assumptions must be made. To determine k, the turbulent intensity
$$I = \frac{||u'||}{||\mathbf{U}||},$$
was assumed to be a fairly low 5 %. The inlet value of k could then be determined using
$$k = \frac{3}{2} I ||\mathbf{U}||^2 \qquad \left[\frac{m^2}{s^2}\right].$$
For estimating inlet ϵ, a turbulent length scale (l) was assumed to be 10% of the particle diameter. Subsequently, ϵ was determined using the definition
$$\epsilon = \frac{C_\mu^{0.75} k^{0.5}}{l} \qquad \left[\frac{m^2}{s^3}\right],$$
where the turbulent model constant $C_\mu = 0.09$. Though these are initial guesses for k and ϵ, the values of both quickly equilibrated and changed to a more realistic value ones the fluid entered the packed-bed.

At this point in the overall discussion, the domain on which calculations have been performed on was introduced (Chapter 3); the theory of turbulence along with the modeling methodology for simulating turbulent flows; and the details into the models used to approximate physical phenomena within the packed bed have been presented. Specific details concerning the convergence history, relaxation values, equation discretization schemes, linear system solvers have been left out of the discussion thus far. In the next section, the results of the simulations on randomly packed-beds of cylinders is given with a thorough discussion over three-dimensional data, radial profiles, and a newly developed post-processing viewing perpendicular profiles is provided.

4.3 Results and Discussion

In this section, the complex three-dimensional data from the velocity field, the classical radial profiles in the packed-bed, and a newly developed perpendicular profile will be discussed. Results for both laminar and turbulent flows in packed-beds of 3 cm diameter and length cylinders in a 40 cm tube were determined. Particle Reynolds numbers of 10, 100, 1500, 2500, 3000, and 5000 were simulated until steady-state solutions were reached. A steady-state solution was considered to be converged when the each variable residual measurement was constant, the mass flow rate through the domain was constant, and several chosen probe point values within the system were constant. The convection term in the RANS model was discretized with the 2nd order linear upwind scheme with a least-squares gradient approximation that was limited with a cell based limiter. An overall convergence criteria of 1e-6 for all variables was used, along with relaxations values of 0.7 for velocity, 0.3 for pressure and 0.4 for k and ϵ was set and is considered ideal for this type of simulation. Using these parameters, pressure drop calculations were determined as an initial point of discussion and are presented in the next section.

4.3.1 Pressure-Drop Calculations

Estimating pressure drop in a packed bed is often performed using Ergun's equation (Equation 4.25), where the classical values for the constants E_1 and E_2 are 150 and

7/4 respectively [17].

$$\underbrace{\left(\frac{(P_0 - P_l)}{L}\right)\left(\frac{\rho d_p}{G_0^2}\right)\left(\frac{\epsilon_{bulk}^3}{1-\epsilon_{bulk}}\right)}_{\text{Y}} = E_1 \underbrace{\left(\frac{\mu(1-\epsilon_{bulk})}{D_{tube}G_0}\right)}_{\text{X}} + E_2 \tag{4.25}$$

Based on the particle diameter (d_p) and shape, bulk porosity of the bed (ϵ_b), superficial fluid velocity ($\dot{V}/A_{tube} = G_0$), fluid viscosity, and tube diameter the pressure drop across the bed can be calculated accurately. However, the fitting constants E_1 and E_2 vary from system to system. It is therefore useful to determine the Ergun constants for the calculation of pressure drop at various superficial fluid velocities.

Using the linearized form of Ergun's Equation 4.25, pressure drop data as a function of packed-bed Reynolds number

$$Re_{PB} = \frac{D_{tube}G_0}{\nu(1-\epsilon_{bulk})}$$

is plotted in Figure 4.2, showing an $R^2 \approx 1$. The fit of Ergun's model is highly dependent on boundary conditions, overall convergence, and near wall mesh (i.e. local y^+ values).

As you can see in Figure 4.2, the Spalart-Allmaras model is capable of producing results that will fit Ergun's equation perfectly. However, if the Lam-Bremhorst model is used the values will not match the Ergun Equation. Though values of pressure-drop that did not fit Ergun's model, qualitatively similar results were seen between other turbulence models. Additionally, it is known that the Ergun equation in reality does not fit very well to flows in which the particle diameter to tube diameter ratio is less than 25. Knowing this, Ergun's Equation should not be used as a primary source of validation for interstitial-scale CFD modeling efforts on small systems though this is often the case in the literature. Ideally, instantaneous and time-averaged three-dimensional data on the interstitial scale should be used for complete validation. However, the length-scales of the voidage, the physical geometry of the packed-bed, and fidelity required for such validations make it a difficult task. The next subsection will discuss the three-dimensional velocity data and associated derived variables from the velocity field and RANS model.

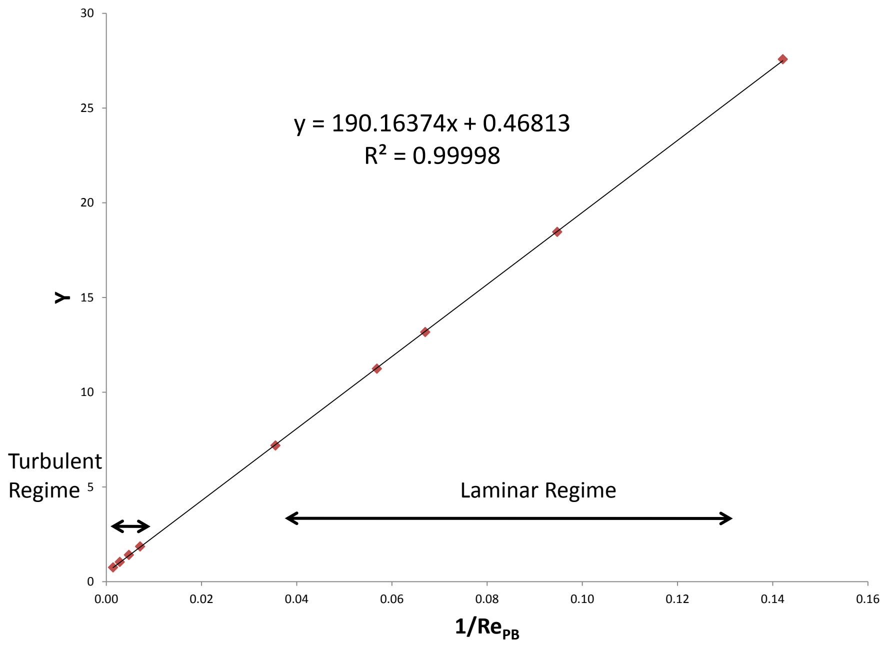


Figure 4.2: Typical pressure drop data from the simulations that is fitted to Ergun's equation (Equation 4.25) using the fv^3 version of the Spalart-Allmaras turbulent closure model

4.3.2 Three-Dimensional Data

For complex three-dimensional data, it often illustrative to present the data in its raw image form for qualitative inspection in order to draw some conclusions about the nature of the flow. For this study, all three-dimensional data is given as a slice of the domain along the (1,-1,0) or (0,0,1) miller index planes. These planes are shown in Figure 4.3 as a red line for clarity.

For plug flow reactors it is well-known that deviations from plug flow are common and strongly depend on Reynolds number, the particle shape and type, radial porosity distributions, and fluid properties of the system. In Figure 4.4 (a) and (b), the low particle Reynolds flows show laminar flow with a relatively uniform velocity distribution. For higher particle Reynolds numbers in the turbulent regime, the flow exhibits a great amount of wall flow that is greater than the inner portions of the packed bed.

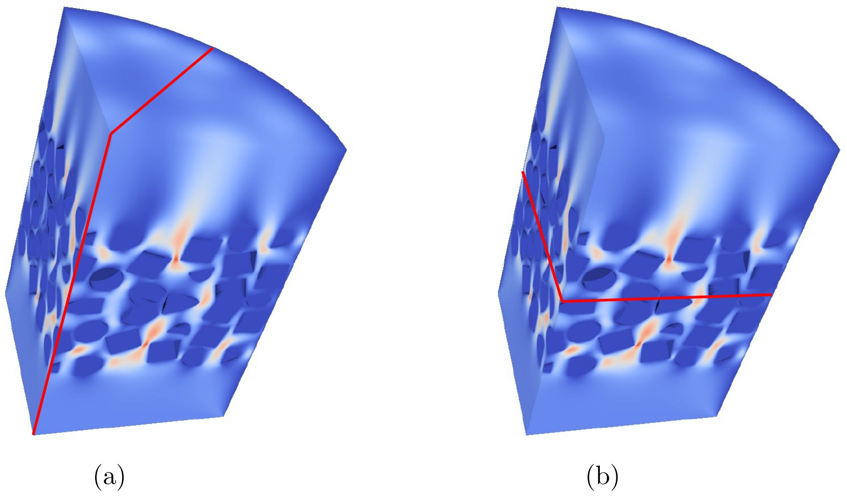
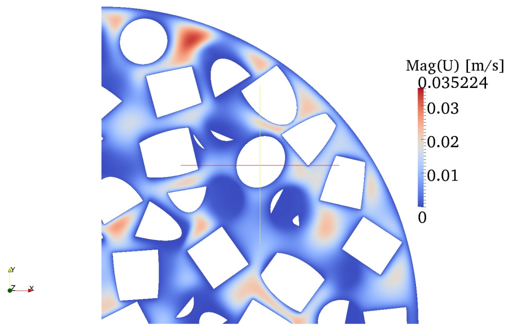


Figure 4.3: **Presentation of three-dimensional data along the (a) (1,-1,0) and (b) (0,0,1) plane through the bed using Miller index notation.**

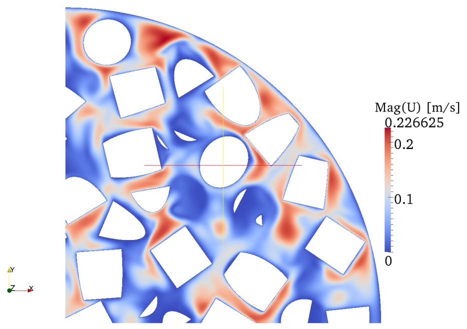
Though this result is not surprising, it is indeed very indicative of how influential the wall effects are on small particle to tube ratio systems.

For an improved perspective of the channeling at the wall, it is constructive to look at the (1,-1,0) plane through the center of the packed bed. Upon initial inspection, one will notice that the bed has been mirrored. This was discussed in Chapter 3, as a method to increase the mesh cell density in order to capture the boundary layers around the particles. With a similar analysis to the previous discussion on the (0,0,1) plane, the turbulent flows show a dramatic increase in the near-wall velocity (i.e. channeling) due to the porosity being the largest at the wall.

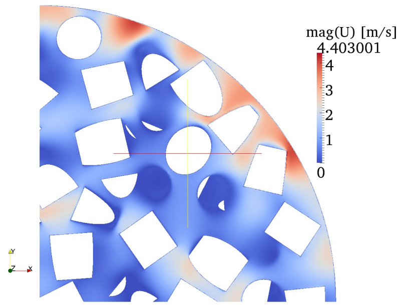
A closer inspection yields a more relevant result, which involves shedding and swirling of flow past the randomly packed cylinders. To illustrate this, a comparison of the velocity and vorticity magnitude near a specific particle is shown in Figure 4.6 for both laminar and turbulent flows. For laminar flows, the fluid moves smoothly around the particles and is evident by the slow moving fluid wrapping around the lower corners of the particle. The vorticity of the between the particle is low due to the non-overlapping streamlines of the laminar flow, and is highest at the interface of the bulk flow and slow moving fluid near the particle surface. This is seen as a thin red layer near the particle surface in the bottom left corner of Figure 4.6 and exhibits relatively



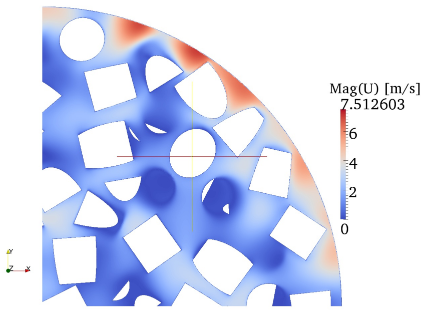
(a)



(b)



(c)



(d)

Figure 4.4: Velocity magnitude slices along (0,0,1) plane through the packed bed for (a) laminar flow at Re$_p$=10; (b) laminar flow at Re$_p$=100; (c) fully turbulent flow at Re$_p$=1500; and (d) fully turbulent flow at Re$_p$=2500

homogeneous mixing near the along the length of the particle surface. The last important observation is the presence of very slow moving fluid behind the particles that extends well behind the particle in question. For the case of the turbulent flow in the right column, the gradients in velocity as one travels away from the particle surface normal are less extreme at large distances from the particle surface. All of the transition occurs near the particle surface, i.e. large wall shear stress over the particle. Accompanying the turbulent flow is an increase in vorticity magnitude at the sharp edges of the particle and behind the particle. This increase in rotation in-turn dissipates the kinetic energy of the flow. The abrupt change in direction of the fluid around the particle is a key source of drag and pressure drop in packed beds, requiring a more thoughtful design for further improvements in pressure drop reduction.

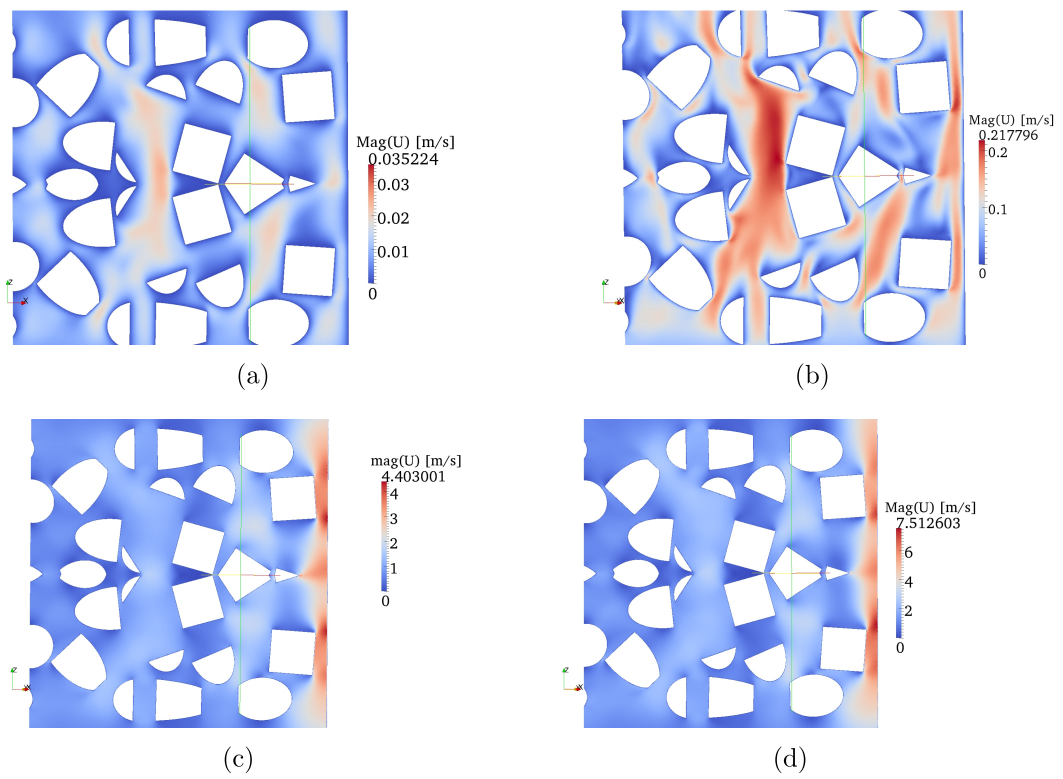


Figure 4.5: Velocity magnitude slices along (1,-1,0) plane through the packed bed for (a) laminar flow at Re$_p$=10; (b) laminar flow at Re$_p$=100; (c) fully turbulent flow at Re$_p$=1500; and (d) fully turbulent flow at Re$_p$=2500. The entrance and exit regions of the beds have been removed for improved visualization.

If more emphasis is put on energy dissipation, a look into the turbulent kinetic energy (k) and turbulent kinetic energy dissipation (ϵ) shows that a relative maximum of k and ϵ coincide with the rotation of the fluid shown in Figure 4.7. If a broader snapshot was taken and translated to the (0,0,1) plane in the bed, we would see a picture of highest k and ϵ in the outer region of the bed at the areas of highest relative velocity. Owing its existence to the complex interactions of the fluid shedding over the particle, it is difficult to derive any meaning directly from inspection beyond what has been noted. Further contraction of the data will yield a more fruitful discussion of near particle behavior, and is provided in the next section covering radial profiles.

In the last portion of the discussion of three-dimensional data, it is beneficial to adopt a more subjective measure to "visualize" fluid vortices. For this purpose, we can use

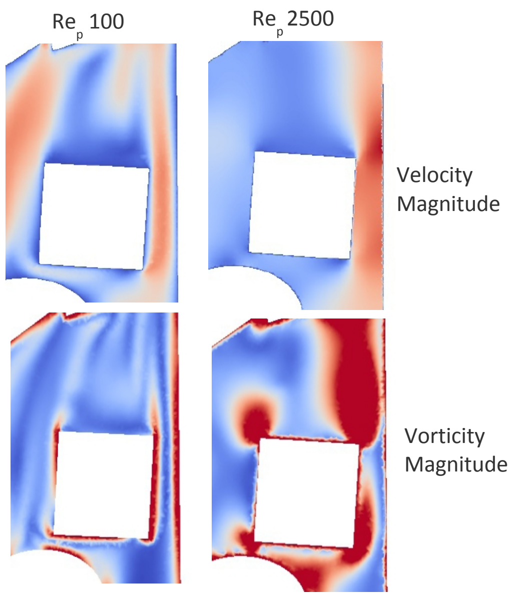


Figure 4.6: Comparison of velocity magnitude (top row) and vorticity magnitude (bottom row) for laminar (left column) and turbulent (right column) flows around a particular particle in the bed.

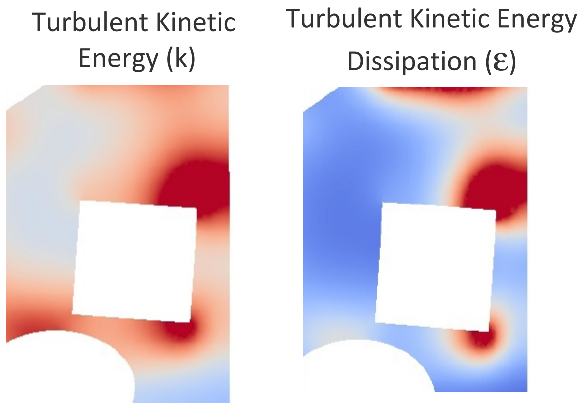


Figure 4.7: Surface plots of turbulent kinetic energy (left) and turbulent kinetic energy dissipation (right) near a particle surface.

the Q-criterion [57] that can be defined as

$$Q = \frac{1}{2}\left(trace(\nabla\mathbf{U})^2 - trace(\nabla\mathbf{U}\cdot\nabla\mathbf{U})\right).$$

Used as a method to obtain a qualitative look at vortex formation, Q-criteria is a subjective assessment of velocity gradients to differentiate pure shearing motions from actual swirling motion of a vortex (something that vorticity magnitude cannot

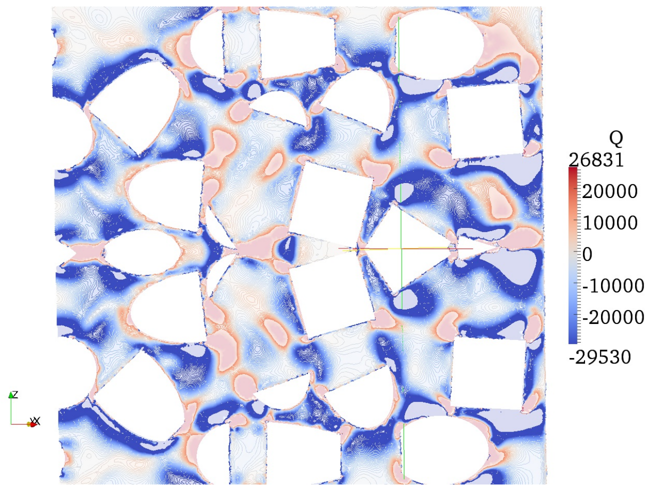


Figure 4.8: Surface and contour plot of Q-criterion used to visualize the presence of vortexes around the packed particles.

provide)[73]. Looking at Figure 4.8, we can paint a very complex portrait of our flow in which flow from one particle strongly influences flow near another particle. The two colors (red and blue) represent positive and negative Q-criterion. Positive (Red) represents a situations in which actual swirl is dominant over shearing, revealing an actual vortex in the system. A vortex shedding off of one particle influences the direction of rotation of another vortex off of another particle in the form of a "doublet". These fluid structures disrupt the formation of streamlines between the particles and increase the convective exchange, but disrupt the formation of a boundary layer. In general, the shedding of fluid and rotation is maximum at sudden changes in the particle surface normal direction. The smooth faces with unidirectional flow over top promote orderly flow, while abrupt changes (such as a fluid approaching a particle) cause convective acceleration, an increase in drag, and eventual shedding of this energy in the form of a swirl. Although this fact may seem trivial, it is one of the key mechanisms of energy loss in the automotive industry and drives the concern for lift and drag in external flow design and simulations. Though this discussion is cursory, there are many more coherent vortex detection strategies [73] that could be used for validation and investigation of packed-bed systems that are not presented here.

The complex three-dimensional results presented here were qualitatively evaluated, giving a sense of the complex process of acceleration, vortex shedding, and dissipation of energy around particles. On a global scale the summation of the individual effects are seen in the pressure-drop (drag forces) and also lumped together in radial distributions of velocity, k, and ϵ. What follows in the next section is a detailed evaluation of the lumping of these individual effects with respect to the global radial direction in the packed-bed.

4.3.3 Distributions in the Radial Direction

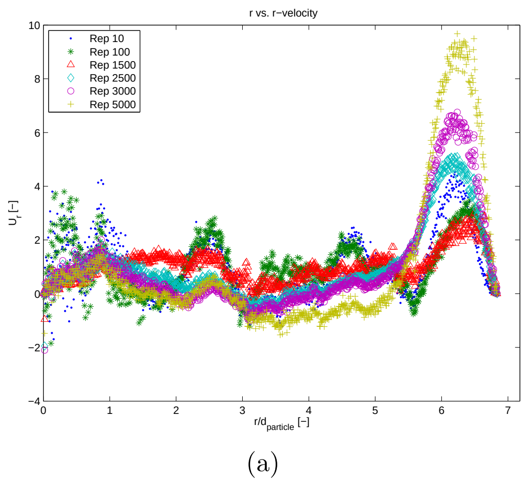
As we saw at the end of the previous section, it is constructive to contract complex data into information of lower tensor dimensionality, eg. vectors $\rightarrow$ vector magnitude or stress components $\rightarrow$ Q-criterion. The analysis of fields in three-dimensional space of higher rank tensors (e.g. vectors and stress tensors) is simplified by determining a representative rank 0 tensor field (scalar field). In a similar spirit of simplification, we can contract data in three spatial dimensions into lower spatial dimensions through averaging through planes or about an axis. This allows for analysis of global interactions by lumping information, with some loss of detail.

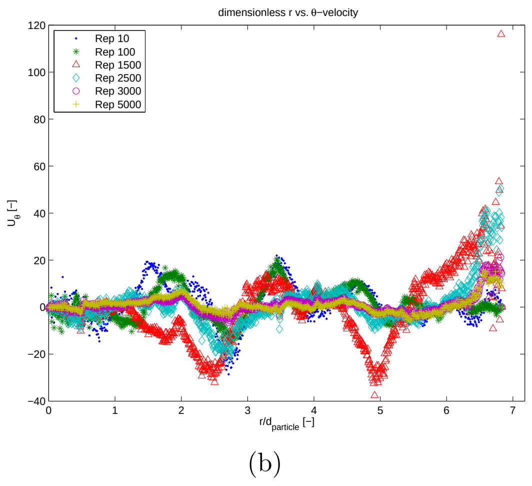
The most familiar form of this spatial averaging is the formulation of global radial data profiles. The data in the packed bed is converted to cylindrical coordinates with the principle axis of the bed aligned with the z-axis of our coordinate system. The data (in r,θ,z format) is then averaged from the base of the packed bed to the height H, and further averaged around the θ-axis. The resulting information is now in one spatial dimension along the r-direction, outward from the centerline along the principle axis.

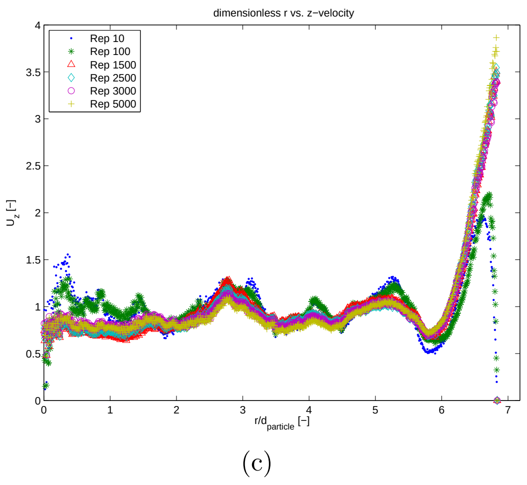
The first and most relevant radial distribution to discuss is the velocity data. For each velocity component (U_r,U_θ, and U_z), the data is non-dimensionalized through scaling with the average value at each Reynolds number [11].

- ***Radial Velocity:*** The radial velocity component (U_r) within the interstitial spaces (Figure 4.9(a)) in the packed-bed had a nearly equal trend of inward

[11]Because the average magnitude of the laminar flow solutions were orders of magnitude lower than the turbulent solutions, each series was non-dimensionalized with average value of that series rather than the laminar values.







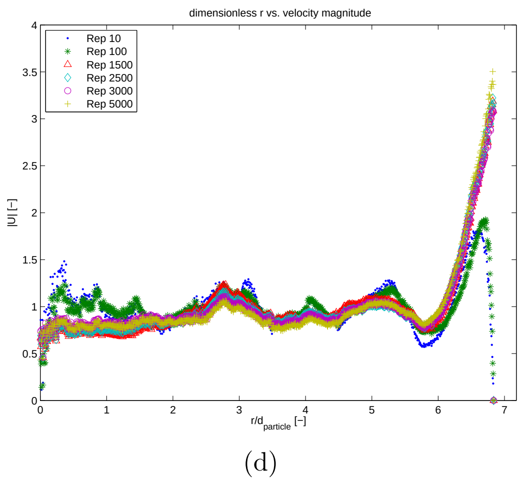


Figure 4.9: Dimensionless (a) radial velocity component $U_r/\bar{U}_r$, (b) azumuthal velocity component $U_\theta/\bar{U}_\theta$, (c) axial velocity component $U_z/\bar{U}_z$, and (d) velocity magnitude $||\mathbf{U}||/||\bar{\mathbf{U}}||$ vs. dimensionless radial distance from the center (r/$d_{particle}$) for laminar and turbulent flows.

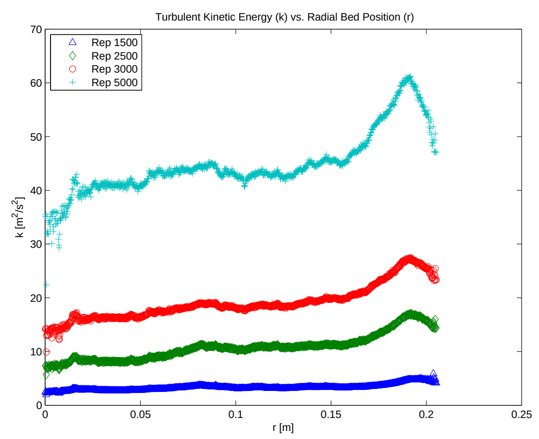
and outward radial velocity towards the center of the bed, with the fluid being pushed up against the tube near the wall of the reactor. Due to the random nature of the particle packed up against the smooth wall, the fluid is pushed outward to the fast moving lower pressure fluid near the wall itself. The presence of a radial pressure profile was not seen on these scales, but may have an effect on the radial velocity. At higher Reynolds number, the radial velocity near the tube wall was higher, indicating the possibility of higher transport from the center of the packed-bed.

- ***Azumuthal Velocity:*** The azumuthal velocity component (U_θ) within the interstitial spaces (Figure 4.9(b)) in the packed-bed had a trend of nearly equal positive and negative velocities, indicating no overall helical motion of the bulk in this particular system.

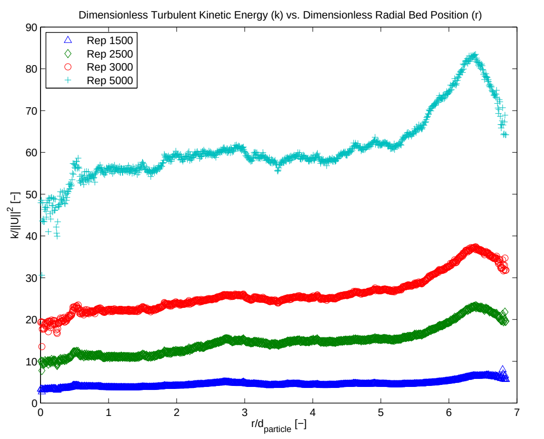
- ***Axial Velocity and Velocity Magnitude:*** The axial velocity component (U_z) and velocity magnitude ($||\mathbf{U}||$) shown in Figures 4.9(c) and 4.9(d) show that axial flow dominates. Comparing the non-dimensional data of Figures 4.9(a) through (d), it can be noted that the axial velocity is an order of magnitude larger than the radial and azumuthal velocity magnitudes. Though this is not a surprising fact, it highlights a key dysfunction in packed beds, poor radial scalar transport (e.g. heat and mass transport). A key metric to improve radial transport would be to design a particle that increases radial flow, as convective transport is much faster than conduction.

Secondary to velocity profiles, it is useful to visualize radial profiles of model and derived quantities such as turbulent kinetic energy, turbulent kinetic energy dissipation, and u'. Turbulent kinetic energy represents the root-mean-square of the fluctuating components ($\mathbf{u}'$) of the Reynolds decomposed velocity ($\langle \mathbf{U} \rangle + \mathbf{u}'$), i.e. the kinetic energy per unit mass of the turbulent eddies [87]. Figures 4.10(a) through 4.10(d) are the radial distributions for turbulent kinetic energy and turbulent kinetic energy dissipation. It should be noted that the turbulent kinetic energy (k) vs. radial position plot in Figure 4.10(b) has been non-dimensionalized using the magnitude of the velocity at the lowest turbulent Reynolds number (i.e. $Re_p = 1500$). Furthermore the dimensionless turbulent kinetic energy dissipation vs. dimensionless radial position has been scaled with the magnitude of the turbulent kinetic energy dissipation at $Re_p = 1500$.

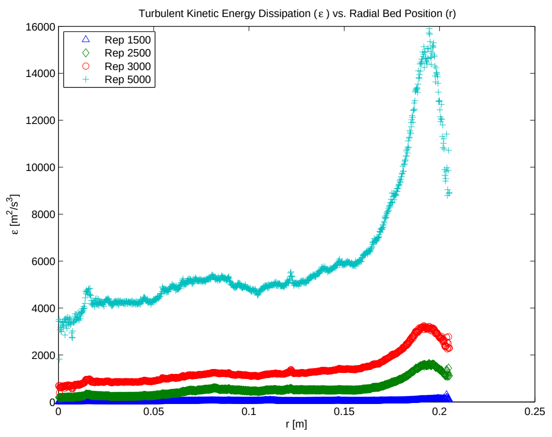
The first note that should be made is that the magnitude of the k and ϵ scale similarly with the previously discussed radial velocity profiles, i.e. higher particle Reynolds number creates higher turbulence in area where porosity is highest. In an area of the bed the is seemingly the most desirable for the fluid to travel, the fluid contacting the particles converts the energy put into the packed-bed into turbulence and is ultimately dissipated and lost. Through this process of creating eddies and dissipating energy, heat and mass transport are enhanced the most in this region in the bed. This also



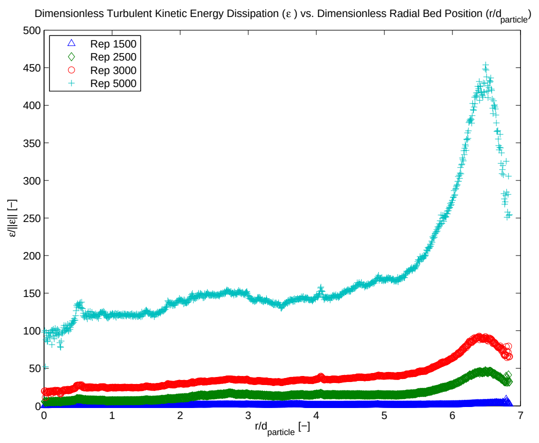
(a)



(b)



(c)



(d)

Figure 4.10: Dimensionless (a) radial velocity component $U_r/\bar{U}_r$, (b) azumuthal velocity component $U_\theta/\bar{U}_\theta$, (c) axial velocity component $U_z/\bar{U}_z$, and (d) velocity magnitude $||\mathbf{U}||/||\bar{\mathbf{U}}||$ vs. dimensionless radial distance from the center (r/$d_{particle}$) for laminar and turbulent flows.

implies, through Kolmogorov theory, that the smallest turbulent eddies in the system will present in the region within 1.5 particle diameters of the wall. The mechanism for the generation and dissipation of the turbulence is most evident by the fact that

1. Fluid is accelerated near the wall of the tube due to a pressure driving force and higher levels of porosity.

2. The fluid approaches a particle face that is perpendicular the principle flow direction, requiring convective acceleration around the particle to satisfy no-slip and continuity near the particle surface.

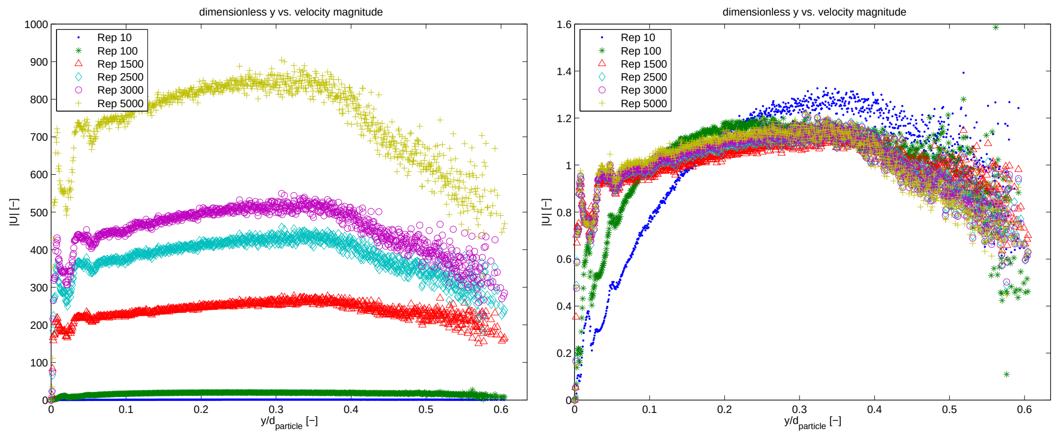
3. The energy of this acceleration is ultimately dissipated in the shedding vortexes off of the adjacent wall particles themselves.

Knowing this mechanism, one may be inclined to design a particle that would either 1) create less shedding vortexes or 2) redirect the flow towards the inner portion of the bed so that the enhancement of the eddy formation can benefit inner portions fo the bed. Though it is known what is occurring on a relatively global scale in the bed, the generation of and dissipation of turbulence is happening on a more local scale close to all of the walls in the bed. Instead of thinking on a global scale, thinking on a scale local to every particle might provide further insight into the near particle fluid behavior at every point in the bed. This is the subject of the next subsection, covering the perpendicular profiles.

4.3.4 Perpendicular Profiles

In the previous section, one-dimensional distributions were formulated with respect to a global coordinate system originating along the centerline of the principle axis of the bed in the outward radial direction. Presented here as a novel method to analyze flow data in packed beds, let us perceive a coordinate system that is again one-dimensional but originating locally from every no-slip (i.e. wall) boundary face in the direction normal to each of these boundary faces. This concept of a one-dimensional model in the perpendicular direction from a wall though novel in the analysis of interstitial-scale CFD has been widely accepted as a method to describe thin-film theory and mass transfer coefficients[36]. Since field values at every point in the packed-bed are associated with exactly **one** closest perpendicular wall face, a distribution can be created.

If the magnitude of the velocity field at every point in the bed with a perpendicular wall distance y from the nearest no-slip boundary is averaged, we will see the plots shown in Figure 4.11. In Figure 4.11(a), each of the velocity values have been scaled



(a) (b)

Figure 4.11: Dimensionless velocity magnitude scaled with (a) the average velocity of the lowest laminar Reynolds number of $\mathrm{Re}_p = 10$ and (b) the average velocity at each Reynolds number, related to dimensionless perpendicular distance from the nearest wall (y) scaled with particle diameter.

with the average velocity magnitude at the lowest Reynolds number studied (i.e. Rep 10). All of the perpendicular velocity profiles are asymptotically identical at the wall, with a no-slip boundary condition of a zero velocity. As the particle Reynolds number increases, the perpendicular gradient of velocity $\left(\frac{\mathrm{d}\mathbf{U}}{\mathrm{d}y}\right)$ drastically increases, agreeing with the notion of increasing wall-shear-stress with increasing Reynolds number. Within 0.05 y/d$_{particle}$ on the abscissa, we see oscillations of the ordinate at all turbulent Reynolds numbers. These oscillations are due to the mixture of near wall flow environments of high speed flow aligned with the principle flow-direction (i.e. positive z-axis) and the low speed "dead-zones" behind particles. This agrees with our previous description of the three-dimensional flow field in previous sections of near cylindrical particle environments being a mixture of high and low speed flow. At intermediate perpendicular distances ($0.05 \leq y/d_{particle} \leq 0.35$), a linear trend is seen with a consistent maximum near $y/d_{particle} \approx 0.325$. At the furthest perpendicular distances, there is considerable scatter at all Reynolds numbers. This scatter accents the overall complex flow field of high speed rivulets, swirling flow shedding off of particles, and low speed stagnant zones; all averaged together to construct seemingly uncorrelated random random values. In Figure 4.11(b), each of the series have been scaled with the average velocities of the series. Again, similar trends are seen

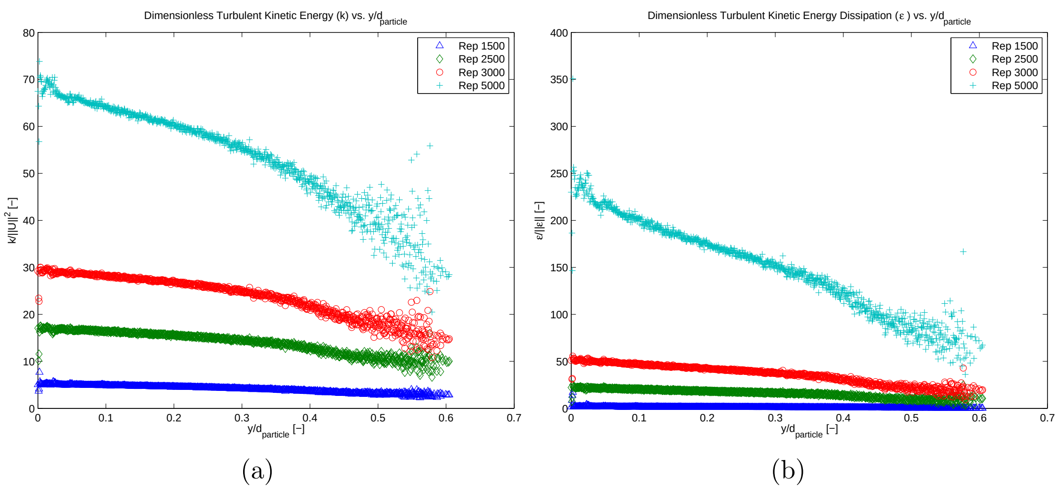


Figure 4.12: Dimensionless (a) perpendicular turbulent kinetic energy profiles scaled with the lowest turbulent velocity magnitude and (b) perpendicular turbulent kinetic energy dissipation rate scaled with the lowest turbulent ϵ.

to Figure 4.11 of high perpendicular gradient of velocity for turbulent flows, while the laminar flows have a much more gradual and parabolic profile. The near wall oscillations are consistent among all Reynolds numbers, but more exaggerated for turbulent flows than laminar flows. The lesser oscillations in laminar flow are consistent with our previous discussion of the three-dimensional data since the streamlines around particles are smooth and relatively similar flow field around the particles at low speeds. Because the effects of the no-slip condition are transmitted further from the particle surfaces through viscous effects, a more gradual velocity profile is seen at intermediate $y/d_{particle}$ ranges for laminar flows. Finally, at larger perpendicular distances the amount of scatter increases.

Further support of this novel one-dimensional approach to extract and conceptualize the data from the complex three-dimensional fields, involves the turbulent kinetic energy and dissipation. Constructed in an identical manner as the perpendicular velocity magnitude profiles, the perpendicular k and ϵ profiles are shown in Figure 4.12. The most notable characteristic of Figures 4.12(a) and (b) is the maximum of both k and ϵ just above the wall surface. This is consistent with our knowledge of turbulent boundary layers in jets [78], in that the maximum production of turbulent kinetic energy (and dissipation) is just above the laminar sublayer. At this point in the turbulent boundary layer, the gradients in velocity are large and the production of turbulence is greatest. Just below this peak, the flow is laminar and therefor no k or ϵ are produced; only diffusing through the laminar region of the boundary layer. Knowing this point of peak k and ϵ production can provide valuable information as to the average characteristic diffusion length and possible thickness of the laminar sublayer in the packed-bed.

The perpendicular profiles seen in this research enhance our picture of the representative boundary layer in packed beds. Thin film theory in packed beds assumes a smooth velocity profile approaches a well-developed representative boundary layer thickness such that

$$k_{mass} = \frac{\mathcal{D}}{\delta_m},$$

relating a mass transfer coefficient (k_{mass}) with the molecular diffusivity ($\mathcal{D}$) and a characteristic mass transfer length (δ_m). With the current results, we may revise this picture to incorporate the global averaging of mixed high and low speed flows around the surface of the particles. Additionally, different particles will possibly give different results of perpendicular profiles, allowing for a point of optimization or yet another (more accurate) empirical relationship for mass transfer coefficient.

4.4 Closure

This chapter brought together the theoretical background presented in earlier chapters and provided a cursory discussion of turbulent theory and the methodology of interstitial-scale momentum transport modeling for steady-state laminar and turbulent flows. The major finding from the research is that much knowledge can be gleaned by a qualitative analysis of the three-dimensional flow fields, radial profiles, and perpendicular profiles. More specifically, the shape of the particle perturbs the flow near the surface and downstream to create both positive and negative flow characteristics that should be more thoroughly understood and applied to improved particle design. What follows are several take home messages.

- **RANS Models of Momentum Transport:** RANS-based models provide an important description of the steady-state flow-field within the packed which is otherwise extremely complex and highly depended on boundary conditions, turbulent closure relations (i.e. turbulent viscosity model), mesh structure, equation discretization, and convergence criteria. What has been lacking in the literature for interstitial-scale modeling is a definitive method for modeling the transport phenomena in packed-beds.

- **Three-Dimensional Data Analysis:** Clear differences in near particle flow phenomena were shown for several cases of laminar and turbulent flows. For laminar flow, the fluid moves around the particle creating very little swirl near the sharp edges of the particle. The laminar flow exhibited very uniform flow through the radial direction in the bed, however had many regions of recirculation in-between and downstream from particles. By using the Q-criteria proposed by Hunt et. al [57], the differences in near particle flow environment can be further exaggerated to show additional flow complexities and vortex formations.

- **Contraction of Data:** Complex three-dimensional data was contracted for two primary reasons (1) To improved conceptualization and understanding and (2) to reflect complex information to lower dimension models (e.g. dispersion models, Euler-Euler models, etc.). For the radial profiles of velocity, it was seen that the flow magnitudes in the center of the bed were relatively flat with a drastic peak near the wall. This peak at the wall of course coincided with the rise in porosity from wall effects. In addition the radial profiles of turbulent kinetic energy (k) and turbulent kinetic energy dissipation rate (ϵ) showed that there was a much larger presence of turbulence within the first 1.5 particle diameters near the wall. This rise in turbulence and dissipation would indicate the increase in mixing and energy loss near the wall due to the complex approach and swirling the fluid experiences.

- **Perpendicular Profiles:** Perpendicular profiles were introduced as a novel method to post-process the complex velocity, k, and ϵ fields. The profiles agreed asymptotically with the boundary conditions as well as accepted descriptions of turbulent boundary layers in general. It was seen that the turbulent kinetic energy and dissipation experienced distinct peaks just away from the walls and a sharp decline near the wall, inidicating the diffusion and dissipation of turbulence in a laminar sublayer. Overall, the perpendicular profiles can be viewed as a representative boundary layer of the packed-bed, that could be used to refine our viewpoint of thin-film theory in packed-beds and revise one-dimensional models (e.g. mass transfer and thin-film theory).

With a thorough presentation of momentum transport in packed beds, the next step is to pursue a discussion in scalar transport in packed beds. Linking the momentum transport results to scalar transport will identify key points in catalyst design improvements and provide a more detailed picture phenomena in packed beds. This is the subject of the next chapter, covering scalar transport in packed-beds for both laminar and turbulent flow fields.

Chapter 5

Interstitial-Scale Scalar Transport Modeling

5.1 Introduction

In Chapter 4, interstitial-scale momentum transport modeling was performed on a computational mesh generated from the methods discussed in Chapter 3. The interstitial-scale flow field was found to be a complex composite of phenomena in which fluid elements experienced convective acceleration around particles, followed by shedding of vortices; the formation of developed boundary layers was disrupted by the complex streamlines and shedding of vortices; and a large number of low-speed "dead" zones behind and in-between particles. All of these effects are inherently important in scalar transport due to the interconnectivity of the momentum and scalar transport equations.

In this chapter, the theory of turbulence presented in Chapter 4 will be extended in a discussion of turbulent mixing that will lead to an improved understanding of the intricacies of scalar transport in packed-beds. The RANS methodology will be applied to the scalar transport equation, and will require additional effort to close the scalar-flux term arising from our mathematical formulation leveraging Reynolds decomposition. All of the theoretical background will then be used to evaluate transient behavior that is simulated in the packed-bed.

As a method of interrogating the packed-bed, simulated step-tracer experiments in both laminar and turbulent flows were performed. In both flow regimes, three-dimensional transient data was collected along with the cup-mixing concentration monitored at the exit of the bed. The flow characteristics captured in the momentum transport modeling is directly linked to the three-dimensional data and age-distribution. Lastly, time-step and turbulent Schmidt dependence is addressed with closing thoughts on the scalar transport simulations. Prior to the discussion of simulation results, a brief background in turbulent mixing and modeling scalar transport in turbulent flows is provided.

5.2 Scalar Transport Modeling

5.2.1 Scalars and Turbulent Mixing

A scalar is defined as a rank 0 tensor representing a simple physical quantity (e.g. volume fraction, mass, temperature, or species concentration) in a system. Subsequently, transport equations are used to describe the conservation of a scalar within a system to represent inflow, outflow, generation, and accumulation of a scalar quantity. If scalar transport does not interact with concurrent transport processes in the system, it is appropriately named a passive scalar. Conversely, if the scalar affects other simultaneous transport processes, it is referred to as an active scalar (e.g. thermal interactions with momentum through a buoyancy term in the momentum conservation equation). Scalar transport (passive or active) is present throughout many scientific fields, from basic science in the study of transport phenomena to industrial engineering processes. In the case of a scalar in a flowing fluid, turbulence is often encountered and requires additional consideration.

Although our understanding of turbulent flow has grown tremendously over the last century, fundamental knowledge of the mechanism of conserved scalar transport in a turbulent flow is still a growing field of research. Recent reviews by Dimotakis[39], Warhaft[131], and Tominaga and Stathopoulos[126] provide valuable background covering the experimental, theoretical, and modeling work concerning scalar transport during the 20th century. Dimotakis reviewed turbulent mixing, with a thorough discussion into what has been categorized as level-1, level-2, and level-3 mixing where,

- **Level-1** mixing involves passive scalars, such that the combination of a fluid and tracer or fluids of similar properties result in no effect on overall flow dynamics.
- **Level-2** mixing is indicated by the interaction of two fluids causing a change in flow dynamics (eg. Rayleigh-Taylor instability flows).
- **Level-3** mixing is categorized by mixing that produces changes in overall fluid intensive properties (density or composition), resulting in a change in flow dynamics (eg. buoyancy driven flow).

Dimotakis notes that studies on level-2 and level-3 mixing are very much open research topics, while level-1 mixing research has been limited to canonical flows (eg. pipe flow, free shear layers, and jets) and relations to empirical data. The conclusions drawn by Dimotakis end on the hope for experimental studies to produce more detailed data needed to develop large eddy simulation (LES) subgrid scale (SGS) models based on observations of scalar mixing rather than on low-order statistics. Warhaft focused on the turbulent passive scalar, touching deeply on passive scalar anisotropy and the direct connection of turbulent length scales rather than the cascading mechanisms of turbulent interactions. Mainly giving an experimental perspective, cornerstones of turbulence theory are addressed in an effort to understand intermittency. Warhaft's discussion, however fundamentally important in understanding turbulence, is beyond the general remarks of this review and left to the motivated reader. Lastly, the work by Tominaga and Stathopoulos offers a discussion of the Reynolds averaged approach to modeling passive scalars, with an emphasis on gas-phase urban and building diffusion problems. The overall conclusion from Tominaga and Stathopoulos is that there is a need for optimal global Sc_t numbers based on dominant effects instead of the generally accepted values of turbulent Schmidt (Sc_t) number in the range of 0.7-0.9. The implications of a turbulent Schmidt number are provided later in this discussion. The cited reviews provide an appropriate introduction into the topic of passive scalar transport in turbulent flows, but leave room for further improvement on the subject in terms of Reynolds averaged modeling approaches.

5.2.2 Turbulent Scalar Transport Theory

Inasmuch the RANS equation requires closure to account for the effect of chaotic turbulent fluctuations, so does the transport of a passive scalar in a turbulent flow field[45, 105]. The conservation equation describing the scalar transport of species α is

$$\frac{\partial \phi_\alpha}{\partial t} + \nabla \cdot (U\phi_\alpha) = \nabla \cdot D_\alpha \nabla \phi_\alpha + S_\alpha(\phi), \tag{5.1}$$

where the $S_\alpha(\phi)$ is the chemical source term (i.e. rate of production) for species α. Using Reynolds decomposition[17], a scalar quantity ϕ_α is decomposed into an ensemble averaged value $\langle \phi_\alpha \rangle$ and a fluctuating component ϕ'_α as

$$\phi_\alpha(\mathbf{x}, t) = \langle \phi_\alpha(\mathbf{x}, t) \rangle + \phi'_\alpha(\mathbf{x}, t). \tag{5.2}$$

Substituting 5.2 into the scalar transport equation for species α in 5.1, we are left with

$$\frac{\partial \langle \phi_\alpha \rangle}{\partial t} + \nabla \cdot \langle U\phi_\alpha \rangle = \nabla \cdot D_\alpha \nabla \langle \phi_\alpha \rangle + \langle S_\alpha(\phi) \rangle, \tag{5.3}$$

where the second term on the left hand side of (5.3) requires further decomposition into

$$\begin{aligned} \nabla \cdot \langle U\phi_\alpha \rangle &= \nabla \cdot \langle (\langle U \rangle + u')(\langle \phi_\alpha \rangle + \phi'_\alpha) \rangle \\ &= \nabla \cdot (\langle U \rangle \langle \phi_\alpha \rangle + \langle u'\phi'_\alpha \rangle). \end{aligned} \tag{5.4}$$

Back substitution of 5.4 into 5.3 results in the unsteady Reynolds averaged passive scalar equation

$$\frac{\partial \langle \phi_\alpha \rangle}{\partial t} + \nabla \cdot (\langle U \rangle \langle \phi_\alpha \rangle) = \nabla \cdot D_\alpha \nabla \langle \phi_\alpha \rangle - \nabla \cdot \langle u'\phi'_\alpha \rangle + \langle S_\alpha(\phi) \rangle. \tag{5.5}$$

As it was seen in the RANS equation, the unclosed $\langle u'\phi'_\alpha \rangle$ scalar-flux term in (5.5) involves fluctuating components that must be reconciled. In the case of production (i.e. non-zero chemical source $S_\alpha(\phi)$), additional numerical treatment is required to close the ensemble averaged chemical source term $\langle S_\alpha(\phi)\rangle$. Further discussion of the chemical source closure can be found in works by Pope[105] and Fox[45]. The closure of the $\langle u'\phi'_\alpha\rangle$ term in 5.5 is discussed in the next section.

The Closure of The Scalar-Flux term $\langle u'\phi'_\alpha\rangle$

The closure of the $\langle u'\phi'_\alpha\rangle$ term in Equation 5.5 is generally accomplished with either a gradient-diffusion hypothesis (GDH) model similar to the eddy viscosity model (4.8), an algebraic moment (AM) model, or a scalar-flux transport model (SFM). Each of these approaches differ in formulation, complexity of transport equations, and has strengths and weaknesses that will be covered in this discussion. The full derivations and applications of each of these methods are available in the cited literature.

- **Gradient Diffusion Models**
  By far the simplest method to account for $\langle u'_i\phi'_\alpha\rangle$ is to use a gradient diffusion hypothesis (GDH) in which

$$\langle u'\phi'_\alpha\rangle = -D_t\nabla\langle\phi_\alpha\rangle = -\frac{\nu_t}{Sc_t}\nabla\langle\phi_\alpha\rangle\,, \tag{5.6}$$

  with Sc_t being the turbulent Schmidt (or turbulent Prandtl) number and D_t being the turbulent mass diffusivity. The gradient diffusion hypothesis assumes isotropic turbulence for simplicity. As a result, the GDH is known to inaccurately predict turbulent effects in cases where the scalar flux is not aligned with the mean scalar gradient (i.e. highly anisotropic flows). The benefit of the GDH closure is that there are no additional transport equations, making it relatively simple to implement numerically.

- **Algebraic Models**
  A more rigorous approach than the GDH involves determining an anisotropic turbulent diffusivity tensor D^t, first introduced by Batchelor[9] as

$$\langle u'\phi'_\alpha\rangle = -D^t\nabla\langle\phi_\alpha\rangle\,, \tag{5.7}$$

with the difficulty being in directly determining the components of D^t_{ij}. Younis[135] noted that the simplest rational algebraic model (AM) was developed by Daly and Harlow[35], setting D^t directly proportional to the Reynolds Stresses, with

$$\langle u'\phi'_\alpha \rangle = -C_\theta \frac{k}{\epsilon} \langle u'v' \rangle \nabla \langle \phi_\alpha \rangle \, , \tag{5.8}$$

where C_θ is set as a positive constant. This formulation overcomes one of the shortcomings of GDH due to misalignment of scalar-flux and mean scalar gradient but, as Younis[135] comments, 5.8 predicts the wrong magnitude of the scalar-flux in the direction normal to the mean scalar gradient. The conclusion by Younis was drawn through comparison of the ratio of streamwise to cross-stream heat fluxes in fully developed channel flow with heated walls, noting the slight differences of the Daly-Harlow model (5.8) compared to DNS results by Kim[71]. A similar model discussed by Fox[45], reveals a slight variation where

$$\langle u'\phi'_\alpha \rangle = -\frac{k}{Sc_t \epsilon} \langle u'v' \rangle \nabla \langle \phi_\alpha \rangle \, , \tag{5.9}$$

which is used in conjunction with the $k - \epsilon$ or Reynolds-stress models. 5.9 correspondingly overcomes the flaw of scalar-flux and mean gradient misalignment of the GDH, due to its use of the anisotropic $\langle u'_i u'_j \rangle$ term. Both 5.8 and 5.9 draw criticism[135, 45] for their lack of accuracy in predicting the scalar-flux; yet they both imply a strong influence of Reynolds-stress on simple scalar-flux models; a more agreeable description of physical phenomena in turbulent flows than those relying on the simplifying assumptions of turbulent isotropy. The theme of Reynolds-stress influence on scalar-flux is recurrent in commentaries by Churchill[25], related directly to Pr_t, and the discussions in classic literature by Levich[80] on diffusion rates in turbulent boundary layers. Not surprisingly, more extensive models used to close the scalar-flux yield more detailed descriptions of turbulent effects on scalar transport. Hence, one may more accurately resolve scalar-flux by utilizing additional transport equations, which is discussed in the next section.

- **Scalar-Flux Transport Models**
  Similar to the Reynolds-stress equations, in a scalar-flux model (SFM) approach

the $\langle u'\phi'_\alpha \rangle$ is treated directly[135, 45] with the transport equation

$$\frac{\partial \langle u'\phi'_\alpha \rangle}{\partial t} + \langle \mathbf{U} \rangle \cdot \nabla \langle u'\phi'_\alpha \rangle = \nabla \bullet \left(J - \langle u'v'\phi'_\alpha \rangle - \frac{1}{\rho} \langle p'\phi'_\alpha \rangle \delta_{ij} \right) + P + R - \epsilon_\alpha, \tag{5.10}$$

where J is a component capturing the influence of molecular diffusion that is often neglected, since molecular diffusion will be small compared to turbulent diffusion effects in higher Reynolds number flows; P is the closed scalar-flux production term; R is the unclosed pressure-scalar-gradient term; and ϵ_α is the scalar flux dissipation term often neglected if assuming small-scale isotropy. The remaining $\langle u'v'\phi'_\alpha \rangle$ and $\langle p'\phi'_\alpha \rangle$ flux terms are generally closed by relations similar to the gradient-diffusion hypothesis[45]. The triple correlation term $\langle u'v'\phi'_\alpha \rangle$ in 5.10 should be of much lower magnitude compared to the double correlation term $\langle u'\phi'_\alpha \rangle$ in 5.5, making the use of GDH to describe $\langle u'v'\phi'_\alpha \rangle$ an acceptable approximation of less dominant phenomena in the SFM. The SFM is one of the most detailed Reynolds averaged models for scalar-flux transport, and is the most computationally intensive method compared to the GDH and AM presented in this discussion.

Reynolds averaged passive scalar modeling requires the use of a scalar-flux model to close the $\langle u'\phi'_\alpha \rangle$ term in Equation 5.5. Generally, this is accomplished with either a gradient-diffusion hypothesis (GDH) model similar to the eddy viscosity model (4.8), an algebraic moment (AM) model, or a scalar-flux transport model (SFM). Though it is desirable to use the most complex, and possibly the most accurate, it was found during this project that more complex models can give out of the ordinary results. For this reason, the GDH was used exclusively for all transient step-tracer simulations. The methodology for solving the passive scalar transport equation is discussed in the next section.

5.2.3 Scalar Transport Modeling Methodology

In transient modeling of scalar transport in turbulent flows, Equation 5.5 captures the ensemble averaged scalar concentration. Modeling reacting systems in turbulent flows presents additional difficulty, due to the added necessity of closing the $\langle S_\alpha(\phi)\rangle$ in Equation 5.5. For simplicity, these studies are not concerned with any reaction in the bulk fluid, since valuable information about mixing can be derived from a simpler approach of modeling passive scalars.

The method of simulating passive scalars requires that there is only one-way coupling between velocity and scalar transport was permitted. One-way coupling is accomplished by determining a steady-state velocity field from a RANS solving technique, followed by the transient passive scalar transport Equation defined as

$$\frac{\partial \langle C\rangle}{\partial t} + \nabla \cdot (\langle U\rangle \langle C\rangle) = \nabla \cdot D \nabla \langle C\rangle + \frac{\nu_t}{Sc_t}\nabla^2 \langle C\rangle . \tag{5.11}$$

Because the velocity is known, Equation 5.11 is a linear 2^{nd} order partial differential equation. At the inlet of the packed-bed, a constant tracer concentration is set, while the exit and walls are defined as a zero-gradient. At this point, the system is defined fully and an implicit formulation of the problem into a linear system of equations can be achieved and solved with methods described in Chapter 2. With the necessary background in turbulence and scalar transport modeling, the results of the step-tracer studies are discussed in the next section.

5.3 Results and Discussion

In this section, the complex three-dimensional data describing the step-tracer flow through the packed bed will be discussed. Results for both laminar and turbulent flows in packed-beds of 3 cm diameter and length cylinders in a 40 cm tube were determined. The cup-mixing concentration was collected at the exit of the packed-bed until the concentration was approximately 99.9 percent of C_{step}. A detailed discussion of the tracer flowing through the packed bed, age-distribution curves, and dependence of turbulent Schmidt and Courant number are provided in the following sections.

5.3.1 Transient Passive Scalar Transport

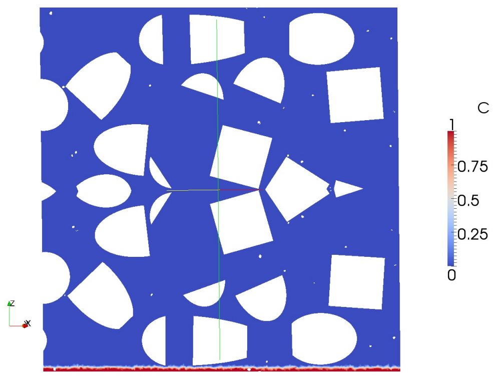
Tracer studies are an integral portion of the analysis of contacting patter within chemical reactors. To perform the studies on the interstitial scale, a cross-sectionally uniform step-input tracer of concentration 1 mol/m^3 was released just below the bed of particle at time zero. At a plane just downstream from the bed, the cup-mixing concentration, defined as

$$\tilde{C} = \frac{\int_{\mathrm{S}} C\,(\mathbf{n}\cdot\mathbf{U})\,\mathrm{d}A}{\int_{\mathrm{S}} (\mathbf{n}\cdot\mathbf{U})\,\mathrm{d}A} \approx \frac{\sum_i C_i\,(\mathbf{n}_i\cdot\mathbf{U}_i)\,A_i}{\sum_i (\mathbf{n}_i\cdot\mathbf{U}_i)\,A_i}, \tag{5.12}$$

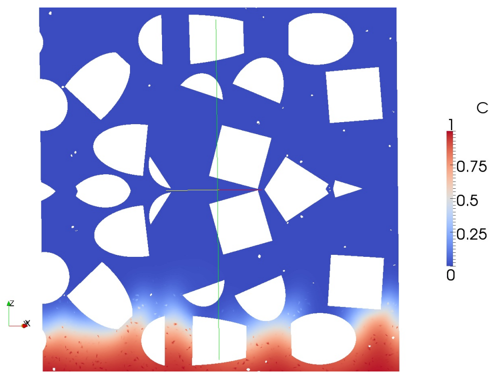
was collected over surface S. The cup-mixing concentration is fundamentally different than the average concentration, since cup-mixing concentration represents the momentum-flux weighted average of concentration.

Before the tracer breakthrough curves are discussed, it is useful to first discuss transport of the passive scalar through the interstitial spaces. For laminar flows (i.e. $Re_p < 250$), the presence of noticeable species diffusion is seen through interface smearing as the tracer flows through interstitial spaces. In Figures 5.1.(a)-(f), it can be seen that there is considerable diffusion and spreading of the tracer front as it travels through the bed. This spreading can be viewed as a residence time effect since as the tracer front move slowly through the bed, there is more time allowed for diffusion to spread the tracer-fluid interface. Furthermore, higher speed convective channels through the bed deliver an abundance of tracer that must diffuse perpendicularly to the slower moving zones behind and in-between particles. Though this is expected, this accentuates the necessity of improvement in particle shape such that convection delivers a scalar to all particle surfaces.

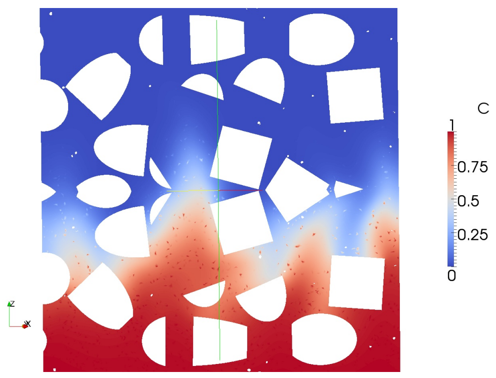
For turbulent flow, the behavior of the tracer was noticeably different and is seen in Figures 5.2.(a)-(f). The tracer moves swiftly through channels of higher convection with very little spread of the fluid front. The fluids arriving at the exit at different ages is primarily due to tortuosity rather than molecular diffusion (residence time effects). Also, at these higher Re_p the flow is much more susceptible to channeling and bypassing due to jets in the flow through the bed; a fact that is much more evident in the E-curves presented in the next section.



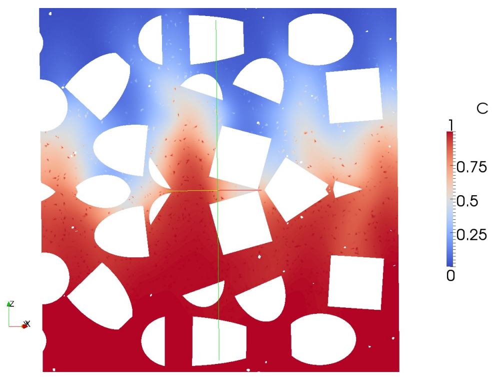
(a) t = 0.00 sec



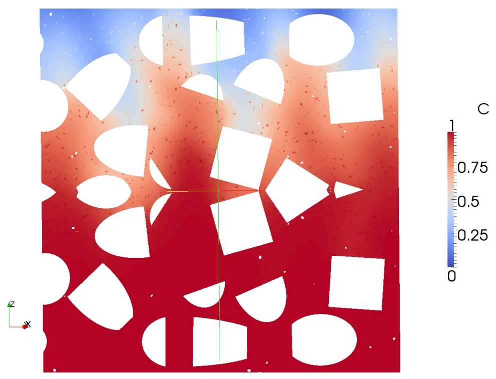
(b) t = 0.25 sec



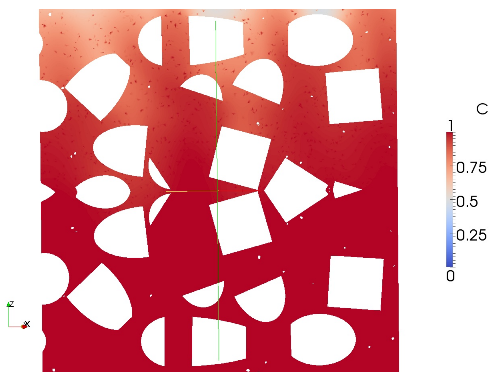
(c) t = 9.00 sec



(d) t = 17.25 sec



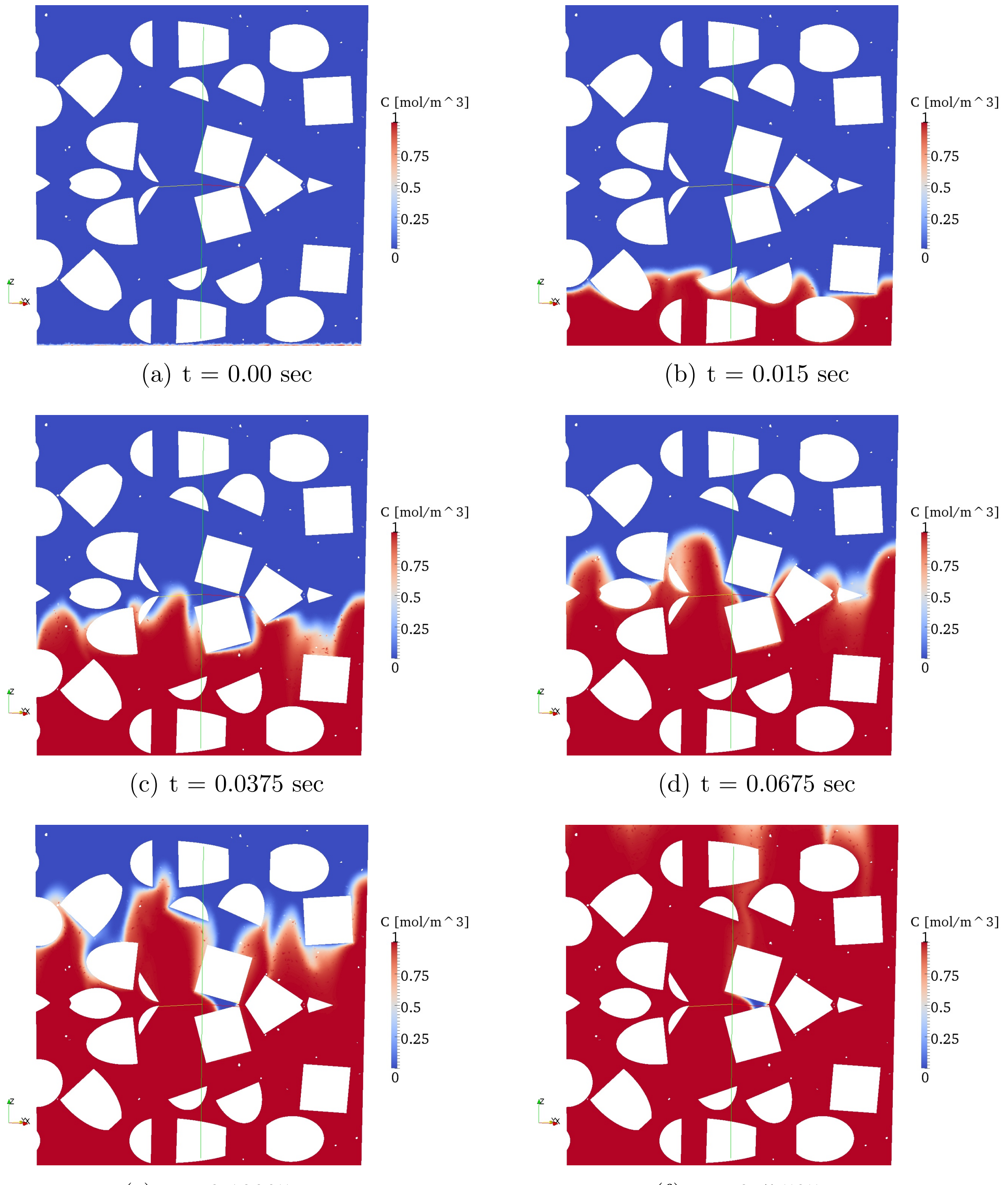
(e) t = 23.25 sec



(f) t = 32.25 sec

Figure 5.1: Laminar ($Re_p = 10$, $Sc = 0.79$, $C0 = 0.8$, and $Bo = 7.9$) step-tracer surface plot through the (1,-1,0 plane) cutting plain with $\bar{t} = 26.8\,[\textbf{sec}]$ and $\sigma^2 = 78.01\,[\textbf{sec}^2]$.

In the snapshots of the step-tracer through the packed-bed, there is a drastic difference between laminar and turbulent flow regimes. For turbulent flows, the spreading of



(a) t = 0.00 sec (b) t = 0.015 sec

(c) t = 0.0375 sec (d) t = 0.0675 sec

(e) t = 0.10005 sec (f) t = 0.47505 sec

Figure 5.2: Turbulent ($Re_p = 1500$, $Sc = 0.79$, $Sc_t = 0.7$, $Co = 0.8$, and $Bo = 1188$) step-tracer surface plot through the (1,-1,0 plane) cutting plain with $\bar{t} = 5.25 \cdot 10^{-2}$ [sec] and $\sigma^2 = 4.74 \cdot 10^{-5}$ [sec^2].

the tracer heavily dependent on the complex structure (tortuosity) within the bed rather than the diffusion of species.

5.3.2 F and E Curve Analysis

Knowing the degree of non-ideality is the key to reflecting interstitial-scale simulation results down to lower order models (i.e. compartmental, dispersion, or even Eulerian-Eulerian CFD models). One principle method to understand deviation from ideal flow, in this case plug-flow, is the pulse or step-tracer experiment. As mentioned before, the cup-mixing concentration (Equation 5.12) was determined for the plane just outside the packed particles, producing a C-curve. For simplicity, the C_{step} was chosen to be 1, making the C-curve equal to the F-Curve, since it is desired to calculate the age distribution (E_t-curve) of the fluid. The F and E-curves are related via

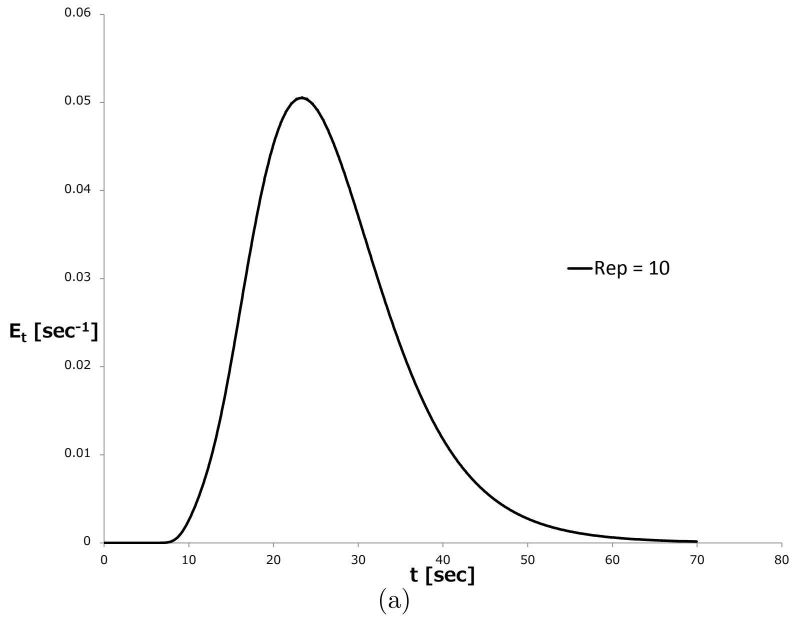
$$E_t = \frac{\mathrm{d}F}{\mathrm{d}t} \approx \frac{F_i - F_{i-1}}{t_i - t_{i-1}} \tag{5.13}$$

Figure 5.3, is a comparison of the age-distribution and dimensionless age-distribution for a laminar and turbulent flow in the packed bed. Several comments can be made about the two figures. The step-tracer in a laminar flow if shown in Figure 5.3.(a), and exhibits a large amount of spread in comparison to the ideal dirac-delta that is seen in a perfect step-tracer experiment. As noted in previous discussions, the spread of this curve is due to overall bed tortuosity, molecular diffusion, slow moving fluid in-between particle, and post particle "dead" zones. it is expected that a different particle shape reducing the slow moving fluid in-between particle and post particle "dead" zones would reduce the spread of the curve. It may be undesirable to eliminate the tortuosity due to the benefit of mixing in the bed. In Figure 5.3.(b), we can see the dimensionless E-curves for laminar and turbulent flows defined by the variables

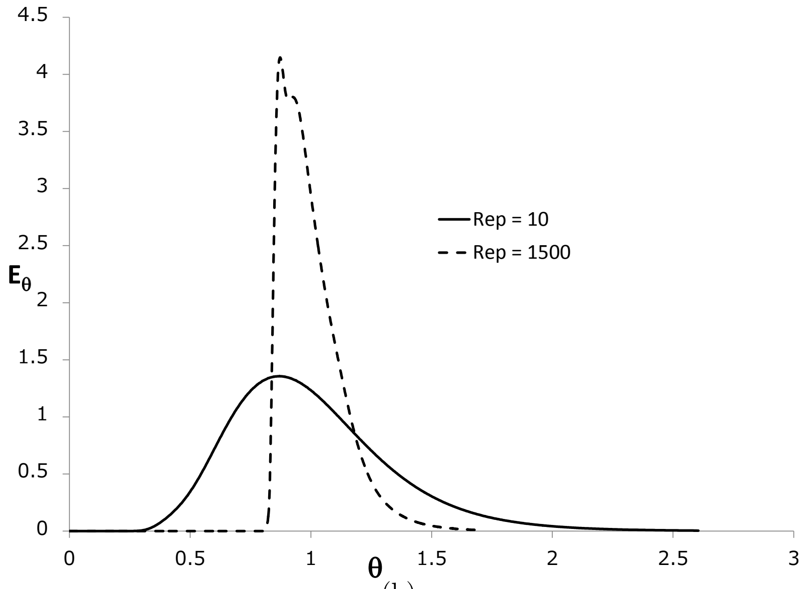
$$E_\theta = \bar{t}E_t \tag{5.14}$$

$$\theta = \frac{t}{\bar{t}}. \tag{5.15}$$

For the turbulent flow we can immediately see that there is much less variance in the curve and that there is a sharp initial front in the $Re_p - 1500$ curve. This sharp jump is due primarily to channel or bypassing that is near the wall. This was seen in our previous discussion concerning the t-dependent surface plots in which it



(a)



(b)

Figure 5.3: Presentation of (a) an age-distribution curve for a laminar flow with $Re_p = 10$ and (b) a comparison of dimensionless age-distribution (E_θ) versus dimensionless time ($\bar{t}$) between a laminar flow and fully turbulent step-tracer.

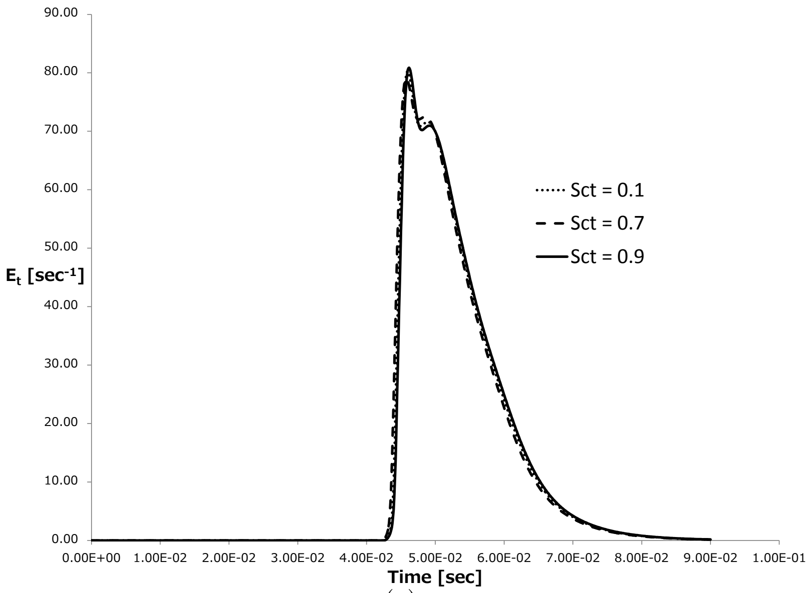
was noticed that the flow was much more susceptible channeling phenomena. Not surprising, this is seen in even higher Reynolds number flows. The channeling seen in these small geometries may be due to the physical setup of the domain and has not been mentioned by other researchers. Since most research project in interstitial-scale modeling do not present tracer experiments, it is doubtful that they do not exhibit similar effects. If so, any added wall heat transfer may be an aberration of channeling rather than the particle shape alone. This fact brings into question the validity of drawing conclusions on such small domains.

The last point of analysis in this portion of the project is an investigation of the dependence of turbulent Schmidt and Courant number on the results. Firstly, Figure 5.4(a) shows the effect of turbulent Schmidt number on a step-tracer experiment. It was found that there is no significant difference between the resulting E-curves. What was of importance is that at lower turbulent Schmidt numbers, more inner iterations are necessary to converge the solution at each time-step. The slight differences between the curves in Figure 5.4(a) are most likely due to the convergence and stopping criteria rather than differences in the model.

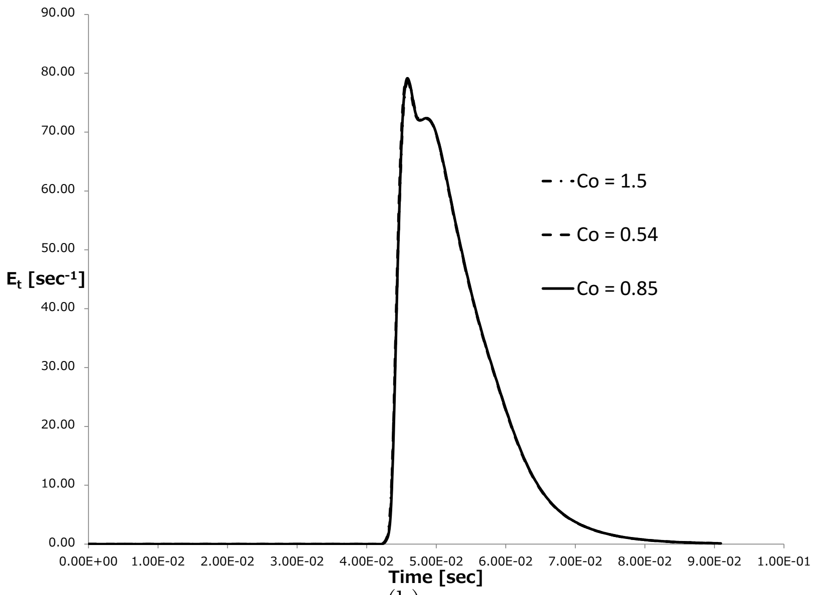
Another unanswered question is the dependence of transient scalar transport calculations on time-step size i.e. Courant number. The cell Courant number is very generically defined as

$$Co_{cell} = \frac{||\mathbf{U}||\delta t}{\delta x}, \tag{5.16}$$

where δt and δx refer to the time-step size and spatial discretization width. The Courant number is used as a time-step size control method since there are practical and stability limitations (for explicit schemes) with time-step size that are system dependent. Equation 5.16 is adequate when a finite difference formulation is used, however is not sufficient for a finite-volume formulation. Because the finite-volume method is primarily concerned with cell-face values and face-fluxes, a formulation based on the face-flux of momentum is used and is discussed in appendix B.2. What is important to note is that based on an alternate formulation, Courant number limitations of 1 are not necessary for implicit finite-volume formulations of the problem. Lastly, Courant numbers as high as 1.5 were used and did not presented stability issues; did not cause unbounded and unphysical results; and produced no difference in the age-distribution curves presented in Figure 5.4.(b).



(a)



(b)

Figure 5.4: E curves for flows with $Re_p = 1500$ comparing (a) the effect of turbulent Schmidt and (b) the effect of Courant number.

5.3.3 Effect of Scalar-Flux Model

As discussed in previous sections, the closure of the $\langle u'\phi'_\alpha \rangle$ term in Equation 5.5 is generally accomplished with either a gradient-diffusion hypothesis (GDH) model similar to the eddy viscosity model (4.8), an algebraic moment (AM) model, or a scalar-flux transport model (SFM). Though one goal of the project was to explore the various algebraic models and scalar-flux transport models, it was found that the algebraic model presented by Fox (Equation 5.9) presented unphysical results. Moreover, a Reynolds stress model would be necessary to further leverage more detailed scalar-flux models. Though it is an interesting topic that is worthy of study, it was decided that this pursuit is left as a continuing project and is outlined as a future project based on this work.

5.4 Closure

This chapter brought together the theoretical background presented in earlier chapters and provided a cursory discussion of turbulent scalar transport theory and the methodology of interstitial-scale scalar transport modeling for transient laminar and turbulent flows. The overall finding is that the overall behavior of the tracer in the packed-bed bed is a combination of the effects of local flow environments (convective acceleration around particle, vortex shedding, dead-zones, low speed zones near contact points, etc.), and the tortuous path that the fluid must travel between the bed entrance and exit. With an improved knowledge of the near particle flow characteristics and how they affect mass transport, additional improvements in particle shape optimization can be achieved. What follows are several take home messages.

- **Reynolds Averaged passive Scalar Equation:** The Reynolds averaged passive scalar equation is capable of providing the ensemble averaged concentration fields for turbulent flows. For Reynolds averaged approaches, scalar-flux closure generally achieved through the gradient-diffusion hypothesis. In this project, the more complex algebraic and scalar-flux transport models gave unphysical results and requires more research and validation arrive at a final conclusion of their utility.

- **Turbulent Schmidt Number and the GDH:** For the work that was performed, there was no sensitivity of the model to turbulent Schmidt within a reasonable range of values for $Sc_t = [0.1, 1]$. What was found is that for lower

Sc_t the number of inner iterations (i.e. iterations to solve the $A\mathbf{x} = \mathbf{b}$ system) over each time-step (outer-iteration). For values above 0.7 there was no significant change in inner-iteration count.

- **Three-Dimensional Data Analysis:** Clear differences in near particle flow phenomena were shown for several cases of laminar and turbulent flows. For laminar flow, the fluid moves around the particle creating very little swirl near the sharp edges of the particle. The laminar flow exhibited very uniform flow through the radial direction in the bed, however had many regions of recirculation in-between and downstream from particles. This characteristic translated to higher dispersion in the age-distribution curve seen during the step-tracer experiments. Most importantly, molecular diffusion to the surface of the particles was a primary mode of transport of species rather than convection. For turbulent flows, the wave front through the bed was defined and convection dominated. The dead-zones behind the particles were not evident in the age-distribution curves, but were seen as an area of design improvement. One such design improvement has been suggested by Nijemeisland [92], requiring holes in the lengthwise direction of the particle to sweep away the dead-zone at the rear of each particle. What has not been addressed is the creation of intense vortexes off of the front of the particles, disrupting the formation of boundary layers over the particle and adjacent particles. By disrupting the formation of the boundary layers over the particles, the characteristic length that a species must diffuse is increased, further limiting already transport limited reaction rates.

- **Residence-Time vs. Tortuosity Effects:** For laminar flows, the tracer front has more time in between changes in direction that might cause mixing. This additional time allows the tracer front to smear, giving way to a more “diffused” fluid-tracer interface. In turbulent flows, the deviation from the flat fluid-tracer front was due to the tortuosity of the complex structure. The direct relationship of tortuosity and the pressure drop in the bed was also alluded to in the previous chapter as well. What can be said is that the convective acceleration (i.e. the change in direction fo the near particle flow fields) around the particle lends itself as the main cause of drag and pressure drop in the bed. Reducing this convective acceleration, while maintaining the complex structure of the bed to move reactants to every face of the particles is key to the improved utilization of packed-bed. For mass transport, these abrupt changes in direction of the fluid adds to the radial and azimuthal spreading of the tracer. A better design be entail a balance between pressure drop and spreading of reactants evenly throughout the bed.

- **Particle Design Improvements:** As mentioned previously, one of the improvements suggested by Nijemeisland [92] was to insert holes along the lengthwise axis of the particle. One of the benefits of of this design would be to "sweep" away the post particle dead-zones. One point that could be made is that by introducing a volume that is not directly in contact with the higher-speed flow outside the particle, the designer runs the risk of creating internal dead-zones. One suggestion from this project is to elongate the particle shape to a form more consistent with a grain of rice. This elongation will reduce the drag of fluid approaching the face of the particle and could reduce post-particle dead-zones. Modeling such a particle will require a reworking of the packing algorithm. It also should be noted that in order to produce an optimized particle shape, the shape must be altered (based on the discussion at hand), repacked, re-meshed, and subjected to fluid simulation. Key objective functions should be identified to enable a more systematic approach to optimization.

One of the key issues with the study of interstitial-scale flow phenomena is the lack of computing power. With more computational speed, one could capture more complicated and transient behavior (e.g. Kelvin-Helmholtz instability) with large-eddie simulation (LES) or hybrid LES and RANS models. One such method to increase the computational power of a cluster as a minimal cost per flop is to leverage the new computing paradigm of graphics processing units or GPUs. The subject of using GPUs to speed up the process of CFD through integrating sparse linear system solvers in OpenFOAM is the subject of the penultimate chapter of this dissertation.

Chapter 6

Implementing Sparse Linear System Solvers Based on CUDA in OpenFOAM

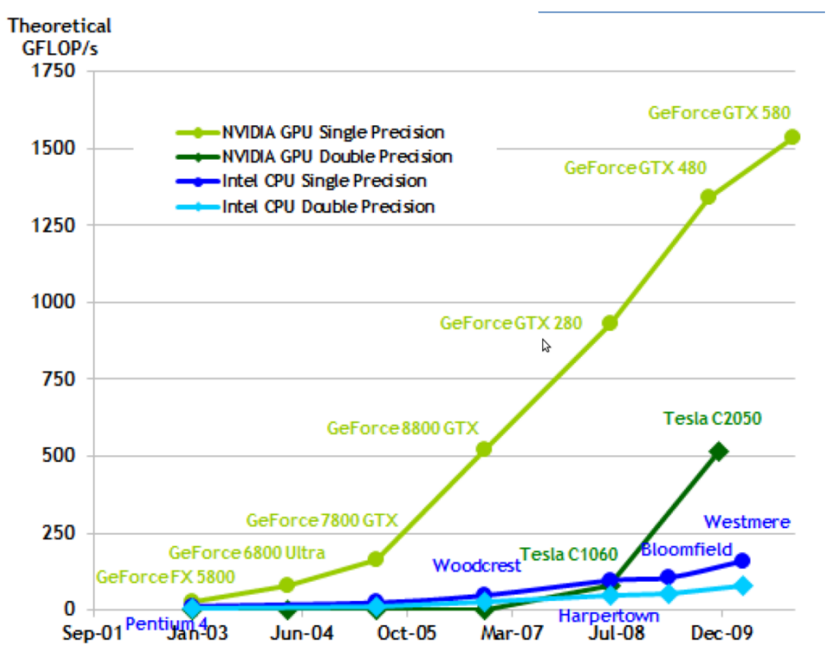
6.1 Introduction

Chapter 2 showed that an implicit finite volume formulation of transport equations produces systems of linear algebraic equations. The solution of these sparse systems is the most time-consuming portion of the CFD solution methodology. This has led to the goal of accelerating a widely used CFD code through the use of graphics processing units to parallelize Krylov subspace linear system solvers [117, 113, 129]. More specifically, an unreleased library for solving sparse linear systems in OpenFOAM [29] using CUDA will be further built upon, optimized, and released to the open source CFD community under the name Cufflink. For the remainder of this chapter, the GPU technology will be introduced, sparse linear algebra and iterative solvers will be discussed, and preliminary results confirming the acceleration of OpenFOAM code will be given.

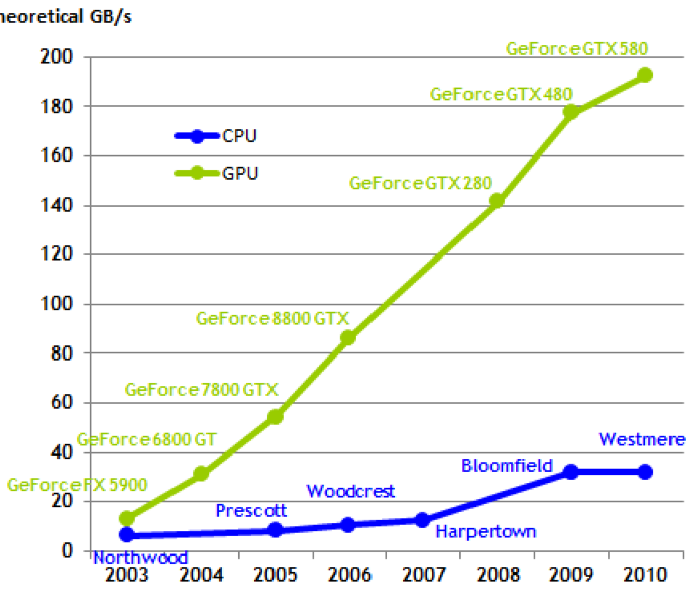
6.1.1 The Graphics Processing Unit: A Shift in Computing Paradigm

In the last decade, Graphics Processing Units or GPUs have become more prevalent in high performance computing due to their performance gains in memory bandwidth and computational speed. Seen in figure 6.1(a) below, the comparison of Intel based CPUs and Nvidia based GPUs for single and double precision calculations show a disruptive jump in performance, in some cases more than an order of magnitude over the CPU. Additionally, the GPU is a fast developing technology with drastically improving memory bandwidth with each generation of GPUs being brought to market. With such growing performance characteristics, the GPU has become a highly competitive and synergistic technology with CPU and is persistent throughout the scientific computing community.

The success of a new computing paradigm requires greater performance over CPUs, a usable programming platform, and reduced communication bottlenecks. The computing speed gains that GPUs possess are directly attributed to the many core, many thread, computing structure of the hardware itself. They are built to execute compute intensive and concurrent actions for data processing rather than flow control and caching [32]. The many core parallel computing approach requires a computing language conducive to simplified multi-thread programming. With this goal in mind Nvidia created the Compute Unified Device Architecture (CUDA C/C++) programming language specifically for Nvidia brand GPU devices. CUDA C/C++ is similar in appearance to that of C and C++ with additional structures for improved control and implementation of parallel algorithms. In addition, many libraries for standard templates, numerical algorithms, image processing, and signals analysis provide further simplified use of GPUs in scientific computing. Lastly, for large problems that cannot be solved using a single GPU, fast communication between multiple GPUs is paramount. Recent advances using Nvidia GPUDirect have produced as much as 30 percent improvement in GPU-GPU communication speed across networked nodes [33]. The advances in communication speed improve with each generation of Nvidia GPUs, as well as each new development in networking hardware. The speed of the parallel execution, fast on-board GPU memory, and improvements in communication speed make the GPU an ideal platform for CFD acceleration. The next section will



(a)



(b)

Figure 6.1: Intel CPU vs. Nvidia GPU [32] (a) GFlops per second performance comparison for single and double precision (b) Memory bandwidth comparison

provide a cursory discussion of sparse linear algebra, the implemented methods of the newly developed library, and outline the computational workflow required for the library to be tied into OpenFOAM.

6.2 Theory and Implementation

It was shown in Chapter 2, that a linear system of equations arises during the implicit finite volume discretization of linear (or linearized) transport equations. The coefficient matrix of these systems are always sparse for partial differential equations, and increase in size for increasing mesh cell and face count [44]. To improve algorithm efficiency and reduce memory requirements, only non-zero entries are stored in specialized storage formats including lower-diagonal-upper (LDU)[12], compressed sparse row (CSR), coordinate matrix (COO), and diagonal (DIA) [112]. Depending on the storage scheme, algebraic operations will be formulated slightly different in order to exploit the sparsity pattern and enable the fastest possible computation.

Once the linear system has been defined through the implicit discretization procedure, a solution must be found in a fast, efficient, and computationally inexpensive manner. Solving the system of equations using direct methods such as Gauss elimination or directly calculating the inverse of the coefficient matrix are computationally prohibitive on even moderately large meshes. To reduce the expense associated with solving such systems, iterative methods that guess a solution, check if a solution has been found, and redirect the next guess are commonly used. If the cost per iteration is small and the number of iterations are low, then iterative methods will be faster and more efficient than a direct method.

In general, the class of iterative solvers used in this work are based on Krylov subspace methods for both symmetric and asymmetric positive definite coefficient matrices. For clarity, a Krylov subspace method uses the residual vector ($\mathbf{r}_i = \mathbf{b} - A\mathbf{x}_i$) to select a search direction for determining the next guess for the solution. Because Krylov methods determine residual vectors orthogonal to the previous search directions, Krylov methods are guaranteed to produce linearly independent search direction unless the

[12]The LDU format is native to OpenFOAM and is used exclusively in the OpenFOAM library.

residual is zero, i.e. the solution is found [116]. For the sake of this brief discussion, a more detailed discussion is left to Saad [112] and Shewchuk [116]. For symmetric positive (SPD) matrices, methods based on the preconditioned conjugate gradient (PCG) method were used, while for asymmetric matrices a preconditioned biconjugate gradient (PBiCG) method was used. Both of these methods are known to be stable, suffer a low degree round-off error, and can be extremely fast if a preconditioner is used to increase convergence rate and lower iteration count.

The PCG method is method used only for a symmetric coefficient matrix (e.g. pressure field), while the PBiCG is used for asymmetric coefficient matrices (e.g. velocity field, scalar transport field, etc.). For this discussion, only the PCG method is presented to illustrate common linear algebraic operations that will requires special consideration later in the chapter. In Algorithm 2, the bold letters are vectors, the capitalized variables are matrices, and the unbolded lower-case characters are scalars. It can be seen that for each inner iteration of our solver loop, there are two vector dot products, two matrix-vector multiplications, and three vector-vector additions. In addition, there is one vector magnitude calculation when determining if the current solution is accurate enough.

Algorithm 2: Parallel Preconditioned Conjugate Gradient

```
1  Compute: $\gamma = \text{normFactor}(A, \mathbf{x}, \mathbf{b})$;
2  Initialize: $\mathbf{r} = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{z} = M^{-1}\mathbf{r}$, $rz = (\mathbf{r} \cdot \mathbf{z})$, $\mathbf{p} = \mathbf{z}$, and $r_{\gamma,0} = r_\gamma = ||\mathbf{r}||/\gamma$;
3  while $(r_\gamma > tol)$ & $(r_\gamma / r_{\gamma,0} \geq relTol)$ & $(count \leq maxIters)$ do
4      $\mathbf{y} \leftarrow A\mathbf{p}$ ;
5      $\alpha \leftarrow rz/(\mathbf{y} \cdot \mathbf{p})$ ;
6      $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$ ;
7      $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{y}$ ;
8      $\mathbf{z} \leftarrow M^{-1}\mathbf{r}$ ;
9      $rz_{old} \leftarrow rz$ ;
10     $rz \leftarrow (\mathbf{r} \cdot \mathbf{z})$ ;
11     $\beta \leftarrow rz/rz_{old}$ ;
12     $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$ ;
13     $r_\gamma \leftarrow ||\mathbf{r}||/\gamma$;
14     count++ ;
15 end
```

Another important note is that the residual $r_\gamma \leftarrow ||\mathbf{r}||/\gamma$ is scaled by a normalization factor that is unique to OpenFOAM. For completeness, this normalization factor is determined using the procedure

Algorithm 3: OpenFOAM Normalization Factor Computation

```
1 $\mathbf{y} \leftarrow A\mathbf{x}$ ;
2 $x_{ref} \leftarrow \langle \mathbf{x} \rangle$ ;
3 $\mathbf{x}' \leftarrow$ vector of length $n$, all entries equal to $x_{ref}$ ;
4 $\mathbf{y}' \leftarrow A\mathbf{x}'$ ;
5 $\mathbf{y}'' \leftarrow \mathbf{y} - \mathbf{y}'$ ;
6 $\mathbf{b}'' \leftarrow \mathbf{b} - \mathbf{y}'$ ;
7 $\gamma = ||\mathbf{y}''|| + ||\mathbf{b}''||$;
```

This normalization procedure is necessary to make complete one-to-one comparison to the CPU based solvers in OpenFOAM with the GPU based solvers in this library[13]. Now that we have introduced the idea of sparse linear algebra and Krylov subspace methods, the next section will discuss the implemented linear algebraic operations in order to produce a multi-GPU capable library to solve large systems sparse linear algebraic equations.

6.2.1 Implemented Algebraic Operations

Although both CUSP and THRUST provide general transformation (e.g. vector magnitude and Euclidean norm) and operations for level 1 (vector-vector) and 2 (matrix-vector) Basic Linear Algebra Subprograms (BLAS), the multi-GPU implementation of the vector dot product, vector magnitude, matrix-vector multiplication (for domain decomposed geometries), and vector average were not available [14]. The operations created for this work are derivatives of the CUSP operations and an Open Message Passing Interface (MPI) implementation. For this presentation, only the multi-gpu matrix-vector multiplication is discussed, while the Euclidean norm

[13]Often times, this normalization procedure is left out of GPU based linear system solvers and often misrepresents true results.

[14]This is as of Summer 2011, and may be different at the time of this publication

(`gpuSumMag(vector)`), vector dot product (`gpuSumProd(vector,vector)`), and vector average (`gpuAverage(vector)`) are presented in detail[15] in Appendix B.1.1.

Domain Decomposed Multi-GPU Matrix-Vector Multiplication

One of the strengths of Krylov subspace methods is the fact that it eliminates matrix-matrix multiplication [127], making the matrix-vector operation the most time consuming portion in the solution process. In recent years, as the GPU technology has progressed, highly efficient and highly parallel matvec multiplication algorithms have paved the way for faster scientific computing [48]. Because of the importance of the matvec operation in the multi-GPU implementation of Krylov methods, it deserves special discussion about how it ties together with the coarse grained parallelism through domain decomposition that OpenFOAM does so efficiently.

When a physical domain is decomposed using OpenFOAM (i.e. simple, scotch, or metis decomposition from Section 3.3.5), the mesh is broken up so that a section is assigned to a computational node or processor. Each computational node (e.g. a cpu-core of separate GPU) can be on the same machine or spread across a network. The idea of domain decomposition was first encountered in Section 3.3.5, where Figure 6.2 is a decomposed packed-bed mesh with the different colors representing different processors associated with that portion of the mesh. On a simpler mesh, each of the sub-meshes are numbered by processor, where each domain knows the location of their neighbor domains, and they are coupled through a processor interface shown in Figure .

Overall, our goal is to solve the associated $A\mathbf{x} = \mathbf{b}$ system created by our implicit finite volume method, but in a faster way though coarse-grained parallelism. Each of these sub-meshes in Figure 6.3 have an associated coefficient matrix we will label A_{ii} and an interface matrix formulated from the processor interface coefficients we will call A_{ij}. Overall, the global matrix is composed of all of the sub-matrices and interface matrices simply called A. Moreover, each sub-mesh has an associated $\mathbf{x}$ and $\mathbf{b}$ such that we can visualize the global problem similar to Figure 6.4.

[15]The actual code for these transformations and level 1 BLAS implementations is located in the *cufflink-library/lduMatrix/solvers/CFL_Headers/globalOps.H* file.

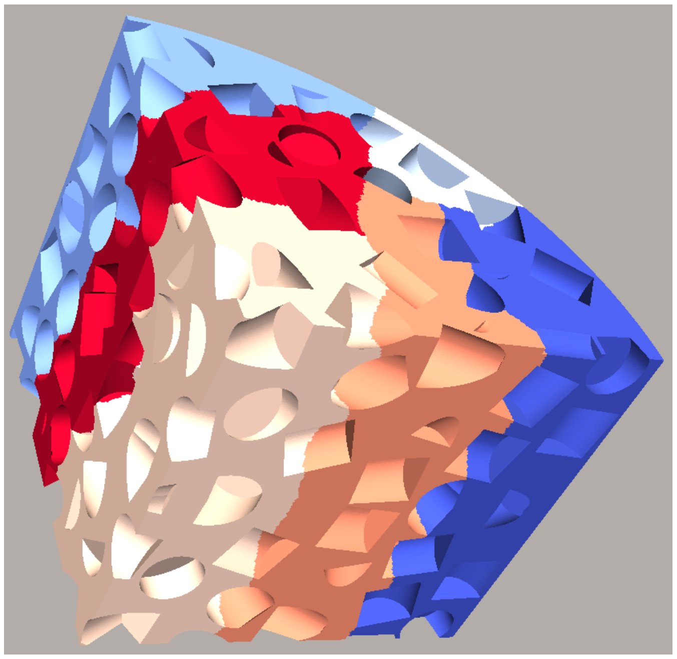


Figure 6.2: Packed-Bed mesh decomposition using Scotch

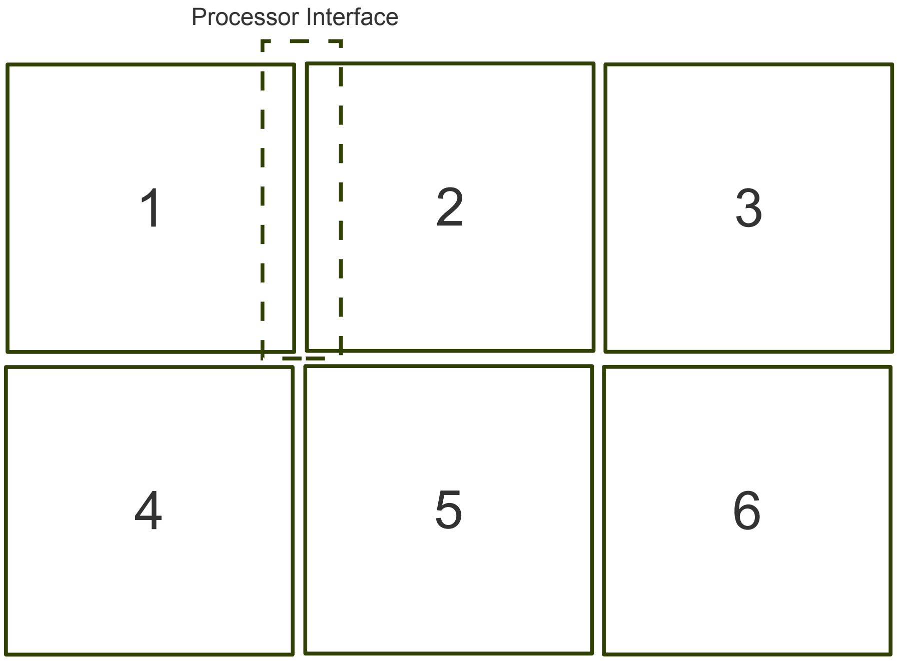


Figure 6.3: Simple domain decomposition across six nodes, each containing processor interfaces between sub-meshes

What is important to note here is that each processor interface has two sub-matrices and the interface matrices between two nodes are the transpose of each other i.e. $A_{ij} = A_{ji}^T$.

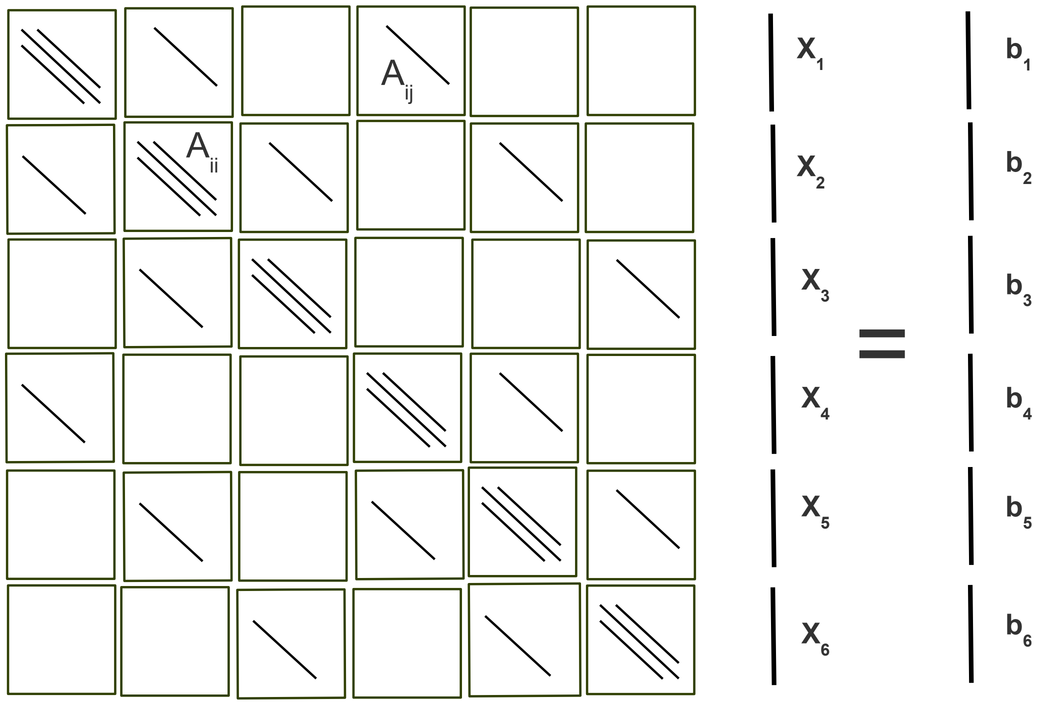


Figure 6.4: A global representation of our decomposed $A\mathbf{x} = \mathbf{b}$ system with coefficient (A_{ii}) and interface (A_{ij}) matrices for each sub-mesh

Now that we have defined out decomposed system, the remaining task of defining the matvec multiplication is relatively straightforward, yet the implementation is rather difficult to optimize. The product vector ($\mathbf{b}_i$) for each sub-domain is now dependent on the coefficient matrix (A_{ii}) and vector ($\mathbf{x}_i$) of that sub-domain plus the influence of the interface matrices (A_{ij}) and neighboring sub-mesh vector values ($\mathbf{x}_j$). This can be defined by

$$A_{ii}\mathbf{x}_i + \sum_{j=0\neq i}^{N} A_{ij}\mathbf{x}_j = \mathbf{b}_i \tag{6.1}$$

where this repeated over N sub-domains. The actual matvec multiplication and summation of the vectors is handled in CUSP. Furthermore, from a programming perspective, this means that each node must have knowledge the interface matrices and vector values of the neighboring sub-meshes to compute a product vector. It is this information transfer that is the subject of constant improvement and optimization and is not presented in the present discussion. Lastly, the reader must recognize that **any** operation within an iterative method involving a matvec multiplication in a decomposed domain must go through this algorithm. The next section will discuss the iterative solution process using the presented matvec multiplication that is now referred to as the `gMatVec(matrix,vector)` function.

6.2.2 Parallel Preconditioned Conjugate Gradient Method with Normalized Residual

Thus far, the concepts of the iterative solver, the multi-GPU BLAS operations, and the multi-GPU matvec operation have been covered. In order to actually solve the system of equations derived from the decomposed mesh across multiple nodes, a slightly different PCG method should be used. Algorithm 4 outlines the multi-GPU implementation of the PCG algorithm in pseudo-code. When compared to the serial version of the PCG in algorithm 2, all of the vector-vector dot product, matvec, and vector magnitude operations have been replaced with the mulit-GPU version of the functions. Vector-vector addition does not require GPU=GPU communication and was not changed.

Algorithm 4: Parallel Preconditioned Conjugate Gradient

```
1  Compute: $\gamma = \text{normFactor}(A, \mathbf{x}, \mathbf{b})$;
2  Initialize: $\mathbf{r} = \mathbf{b} - gMatVec(A, \mathbf{x}_0)$, $\mathbf{z} = gMatVec(M^{-1}, \mathbf{r})$,
   $rz = gpuSumProd(\mathbf{r}, \mathbf{z})$, $\mathbf{p} = \mathbf{z}$, and $r_{\gamma,0} = r_\gamma = gpuSumMag(\mathbf{r})/\gamma$;
3  while $(r_\gamma > tol)$ & $(r_\gamma / r_{\gamma,0} \geq relTol)$ & $(count \leq maxIters)$ do
4      $\mathbf{y} \leftarrow gMatVec(A, \mathbf{p})$ ;
5      $\alpha \leftarrow rz / gpuSumProd(\mathbf{y}, \mathbf{p})$ ;
6      $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{p}$ ;
7      $\mathbf{r} \leftarrow \mathbf{r} - \alpha \mathbf{y}$ ;
8      $\mathbf{z} \leftarrow gMatVec(M^{-1}, \mathbf{r})$ ;
9      $rz_{old} \leftarrow rz$ ;
10     $rz \leftarrow gpuSumProd(\mathbf{r}, \mathbf{z})$ ;
11     $\beta \leftarrow rz / rz_{old}$ ;
12     $\mathbf{p} \leftarrow \mathbf{z} + \beta \mathbf{p}$ ;
13     $r_\gamma \leftarrow gpuSumMag(\mathbf{r})/\gamma$;
14     count++ ;
15 end
```

As a result of the new algorithm for the decomposed domain, the normalization factor, γ, is now calculated using

Algorithm 5: Normalization Factor Computation

```
1 $\mathbf{y} \leftarrow gMatVec(A, \mathbf{x})$ ;
2 $x_{ref} \leftarrow gpuAverage(\mathbf{x})$ ;
3 $\mathbf{x}' \leftarrow$ vector of length $n$, all entries equal to $x_{ref}$ ;
4 $\mathbf{y}' \leftarrow gMatVec(A, \mathbf{x}')$ ;
5 $\mathbf{y}'' \leftarrow \mathbf{y} - \mathbf{y}'$ ;
6 $\mathbf{b}'' \leftarrow \mathbf{b} - \mathbf{y}'$ ;
7 $\gamma = gpuSumMag(\mathbf{y}'') + gpuSumMag(\mathbf{b}'')$;
```

In order to optimize the performance of the multi-GPU algorithms, efficient communication between GPUs, more thoughtful bundling of data structures (e.g. sparse matrix storage techniques), and optimized node communication hardware must be considered and are in need of further research. In the next section, the subject of how Cufflink is coupled with OpenFOAM is outlined.

6.2.3 Computational Workflow of Cufflink Solvers

Solving the system of linear equations on the GPU involves multiple levels of design. In addition to an efficient algorithm to solve the system (e.g. preconditioned conjugate gradient method), the transfer and manipulation of the data itself must be performed with efficiency in mind. The basic process of moving data from OpenFOAM to Cufflink and a subsequent solve in the linear system solvers involves three major phases that are illustrated in Figure 6.5 and outlined in the following descriptions.

I. **OpenFOAM:** In OpenFOAM, all of the mesh handling and equation discretization are performed using the CPU and stored in the computer memory. Within the top-level solver (e.g. icoFOAM), the method `solve(...)` is called and triggers the LDU format coefficient matrix (A) along with the solution vector ($\mathbf{x}$), source ($\mathbf{b}$), and solver convergence criteria to be passed to a container class holding the linear system. The LDU matrix only stores the diagonal, upper triangular, and lower triangular elements of the matrix with the row and column indices of the upper triangular elements for a total of five vectors. Using the CPU, the LDU matrix is converted to a partial COO matrix by copying the lower, upper, and diagonal elements along with only the row and column indices of the upper triangular elements.

II. **Thrust and CUSP:** Using Thrust and CUSP, the incomplete COO matrix in the computer memory is then transferred to the GPU card memory along with the solver convergence criteria. The remainder of the COO matrix is filled with the row and column indices of the lower triangular elements along with the row and column indices of the diagonal elements. At this point, the COO matrix is reordered in an effort to optimize memory access and can be converted to other formats such as CSR or DIA.

III. **Cufflink:** In this final sequence of steps, the linear system is passed to a specific solver along with the convergence criteria. Once convergence has been reached,

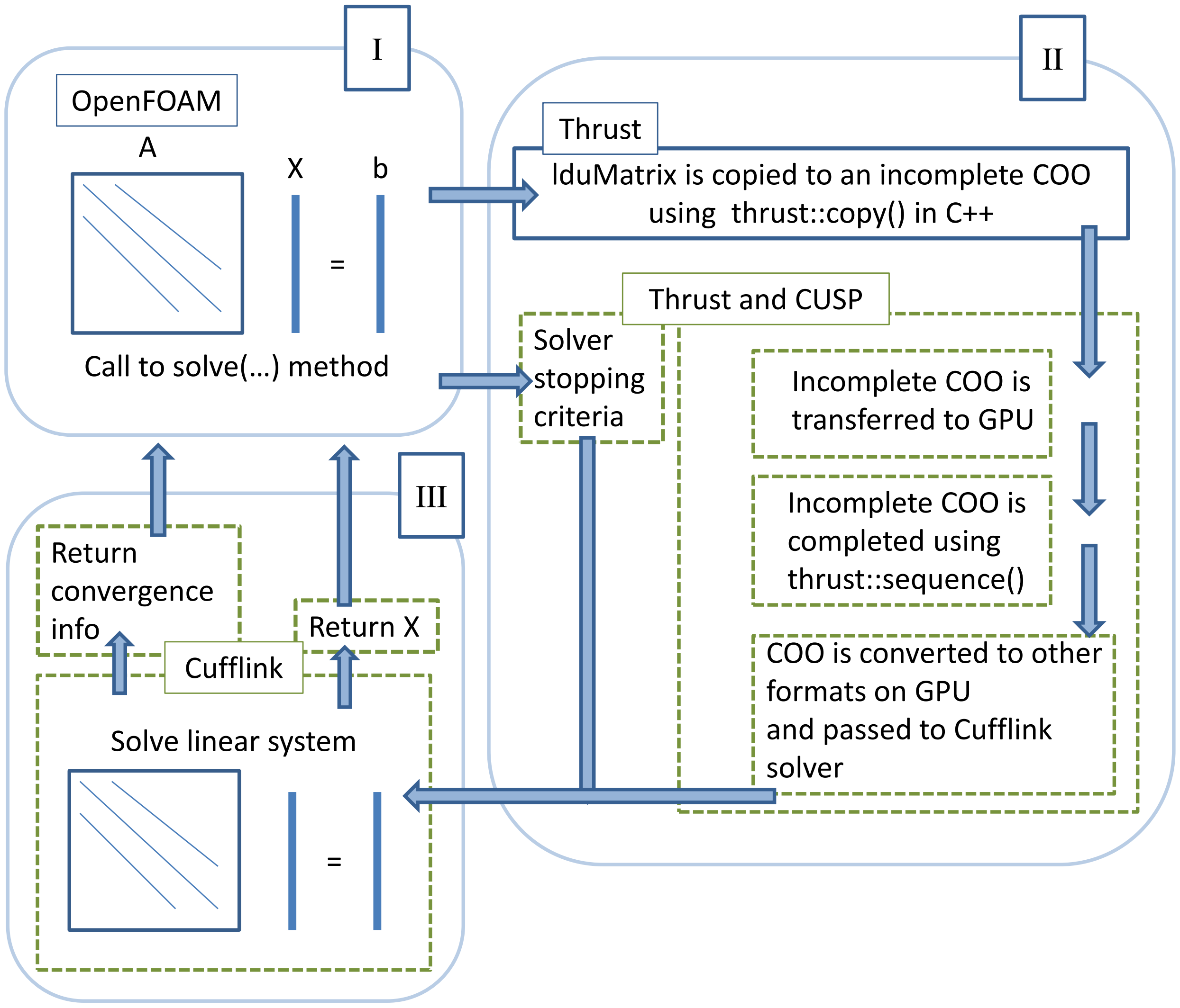

I
OpenFOAM
A
X
b
=
Call to solve(...) method
II
Thrust
lduMatrix is copied to an incomplete COO using thrust::copy() in C++
Thrust and CUSP
Solver stopping criteria
Incomplete COO is transferred to GPU
Incomplete COO is completed using thrust::sequence()
COO is converted to other formats on GPU and passed to Cufflink solver
III
Return convergence info
Return X
Cufflink
Solve linear system
=

Figure 6.5: Data flow diagram outlining the movement and conversion of data from OpenFOAM to the Cufflink solvers. The process is broken up into three portions where (I.) takes place in OpenFOAM C++ code, (II.) leverages Thrust and CUSP to convert and move the information to the GPU, and (III.) the system is solved on the GPU using Cufflink with the solution transferred back to OpenFOAM.

or the solver stops, the solution vector ($\mathbf{x}$) and information about the solutions process (e.g. iteration count, residual, etc.) are passed back to OpenFOAM and the outer-iteration continues to solve other variables in the top-level of the fluid solver.

Though much of the details of the code have been abstracted in this discussion, the overall flow of information is captured by Figure 6.5. Once the data has been moved to the GPU, the calculation often requires less time that the data transfer itself. The next section will present several sets of results for different case studies, along with a discussion pertaining to further studies using the Cufflink library.

6.3 Results and Discussion

Determining the performance of the GPU-based solvers was based on overall clock-times of the top-level solver in OpenFOAM. All of the latency due to data transfer, preconditioner formulation, and the solver itself have been lumped together. As a feasibility study of the technology, a simple case solving the temperature profile in a flat plate was chosen.

6.3.1 Steady-State Scalar Transport

As a proof-of-concept for speed-up achieved by allowing the GPU to solve the linear system created in OpenFOAM, a simple case solving Laplace's equation $\nabla^2 T = 0$ on a square plate was chosen. The reason for such a simple case was to select a problem that is inner-iteration dominant[16], i.e. the overall solve time was controlled by the linear system solver efficiency. For this simple case, all timing was taken before and after the `solve(...)` method was called in the top level solver.

Previous studies have shown GPU solvers to be superior to CPU based solvers in OpenFOAM when comparing the most advanced preconditioned conjugate gradient GPU solver with the unpreconditioned conjugate gradient solver on the CPU in OpenFOAM [29]. Specifically, the multigrid solvers in Cufflink were shown to be more than one hundred times faster than the unpreconditioned conjugate Gradient in OpenFOAM. A more constructive comparison is between the best CPU-based solver

[16] An example of a system that is outer-iteration dominant is the steady-state solution of the Navier-Stokes equation.

in OpenFOAM and the best GPU-based solver in Cufflink. Figure 6.6 shows a comparison of the geometric algebraic multigrid (GAMG) solver in OpenFOAM over six processors with the Cufflink solvers using one and two GPUs.

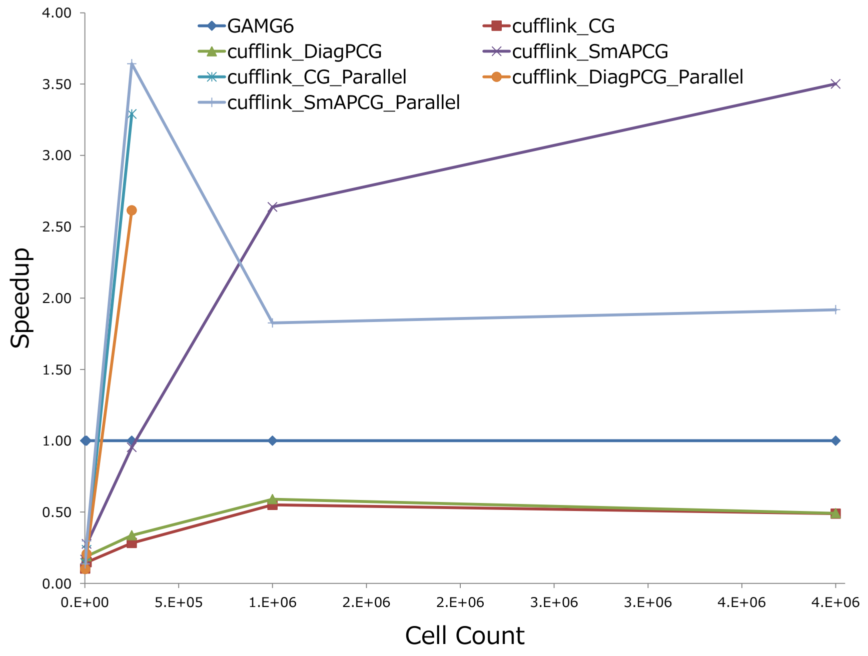


Figure 6.6: Speed-up comparison of geometric algebraic multigrid solver in OpenFOAM over 6 processors (GAMG6) with the several Cufflink solvers using one and two GPUs, where CG = conjugate gradient, DPCG = diagonal preconditioned conjugate gradient, SmAPCG = smoothed aggregation multigrid preconditioned conjugate gradient, _Parallel indicates two GPUs, and cufflink indicates a GPU solver.

On the x-axis, a mesh with size cell count was created over the square plate and the times before and after the `solve(...)` methods were collected. The difference of these two times is the overall solve time. On the y-axis, speed-up was determined by dividing the overall solve time using the GAMG solver on six processors, with the overall solve times of each Cufflink solver at each cell count. Overall, the parallel GAMG solvers are faster than the simpler diagonal preconditioned conjugate gradient and unpreconditioned conjugate gradient solvers in Cufflink. For larger systems, the smoothed aggregation multigrid preconditioned solvers in Cufflink outperformed the GAMG solver in OpenFOAM. The parallel solvers showed significant speed-up, but lost performance when communication between the GPU cards because the bottleneck. GPUs that communicate across a network will show even more loss of performance due to communication issues. Though these results seem to be very simple, they are a significant leap forward that needs more attention that will be discussed next.

6.3.2 Current Developments

Currently, a research group at the University of Adelaide is using Cufflink to perform large eddy simulations applied to ship hydrodynamics. As a preliminary result, they have shown as much as fifteen times speed-up using multiple GPUs on a high performance cluster in their lab. This result, although an initial result is shows that the proposed multi-GPU algorithm is efficient enough to see speed-up in fluid simulations. What has yet to be performed is to optimize communication between the GPUs, where this is a hardware and software based solution that will not be discussed in the current work. This work will be discussed in subsequent publications after this dissertation due to time constraints.

6.4 Closure

In this chapter, an implementation of sparse linear system solvers based on CUSP and Thrust where coupled with OpenFOAM. This new library, called Cufflink was shown to accelerate the OpenFOAM code compared to the geometric algebraic multigrid solver in OpenFOAM. There are several conclusions that can be made from this work.

- Cufflink is the first library to have multi-GPU linear system solvers that use the domain decomposition in OpenFOAM. No other major CFD software has this capability and this is the first project of its kind.
- Because there is a bottleneck of data transfer from the computer memory to the GPU memory, these transfers of data should be minimized. This implies that CFD solvers in which many outer-iterations are performed will be less efficient than CFD solvers that mostly consist of inner-iterations. For example,

it is very common in steady-state Navier-Stokes solvers that decouple pressure and velocity to only converge the velocity fields by a few percent, relax the solution, and then solve pressure, relying on many outer-iterations for overall convergences. In transient solvers, each time step requires many inner iterations in order to produce a conservative answer for that time step and are therefore inner-iteration dominant.

- More test cases need to be run on optimized hardware. The current setup in our lab is not capable of producing adequate results for publication due to hardware limitations. As a result, collaboration between CREL, the University of Adelaide, and Nvidia should continue after this project.

Chapter 7

Summary of Contributions and Future Work

The overall objective of this work was to investigate heat, mass, and momentum transport on the length-scale of the particles within a randomly packed beds of particles and decipher an appropriate method to generate the random domain and capture various physical phenomena. The principle objectives of this thesis were to:

1. Develop a packing algorithm to randomly place cylindrical particles in a tubular domain while knowing the exact location, orientation, and dimensions of each particle.
2. Develop a strategy to generate computational meshes of these randomly packed particles that minimizes numerical errors.
3. Using computational fluid dynamics, simulate particle-scale momentum transport using the Navier-Stokes equations (laminar flow) and Reynolds-averaged Navier-Stokes methodology (turbulent flow) in OpenFOAM.
4. Using computational fluid dynamics, simulate mass transport both laminar and turbulent flow fields to investigate the sensitivity of turbulent mixing in the system to the choice turbulent scalar-flux approximation methods.
5. Develop heterogeneous computing methodologies using graphics processing units to accelerate simulations and decrease overall simulation time.

With each of these overall goals in mind, there are several conclusion that can be drawn and take home messages from the research.

7.1 Constructing Packed Beds of Cylindrical Particles

- **Geometry generation using a Monte-Carlo packing algorithm** A Monte-Carlo packing algorithm using a sorting method is presented as an effective method to produce loosely packed bed domains of cylindrical and trilobed particles. The particle-particle overlap determination is achieved using a method proposed by Blaak et Al.[18], based on approximating a cylinder with spheres, spherocylinders, and ellipsoids. The bulk porosities of approximately 0.65 are achieved by this method with relatively few packing cycles due largely to the use of a sorted list of particles. The sorting of the particles allowed for particles located closer to the bottom of the container to be packed before particles located higher in the bed. The increase in space created greater degrees of freedom for particles to achieve a more dense packing in less packing cycles.

  The radial porosity distributions produced by the packing algorithm are qualitatively comparable to experimental work by Roblee et al.[110], showing similar wall effects observed experimentally. The main difference being that the experimental results by Roblee showed an extremely densely packed bed of particles near the wall and throughout the bed that is considerably more dense than those seen in CT and DigiDEM reported in the literature[21]. This due mainly to external forces (gravity and manual compression) during the physical packing process in the experimental work. The Monte-Carlo algorithm presented here uses only random motion and a lowering of the top boundary to reduce the degrees of freedom to form the packed bed of particles. For a more densely packed bed, there must be either more packing cycles or an algorithm incorporating a force field similar to ones used in molecular dynamics simulations. The Monte-Carlo packing algorithm proposed in this paper tracks the location and orientation of each face of the particles being packed. Knowing the location and orientation of each particle face facilitates the meshing process as there is no need for more preprocessing to extract the edges and faces from a pixilated image produced using the method by Gan et al.[47] and Caulkin et al.[21]. Ultimately, this image extraction process may introduce more approximations into the already extremely intricate structures seen in packed beds of particles.

The resulting computational meshes fully define the location of the particle faces within a domain of randomly packed cylinder based particles. In terms of interstitial CFD modeling, knowing the exact location of the particle faces allows for extremely fine boundary layer meshes to be used to increase resolution of near particle modeling of transport phenomena. Ultimately, meshes accurately describing the microstructure of packed beds enable more realistic CFD simulations, providing further insight into the fluid dynamics on the length scale of the interstitial spaces between particles. As a result, a deeper understanding of fluid flows within packed beds can be achieved and will ultimately improve catalyst shape optimization, current models describing reactors, and unit operations leveraging the intricate structure of packed beds.

- **Mesh Generation** The mesh generation portion of the project was a time-consuming task that involved numerous meshes (almost 50 gigabytes of data), with solutions being attempted on most of the created meshes. The method developed to create a high quality mesh can be summarized in the following steps:

    1. Create the underlying domain in GAMBIT using the output journal file from the Mote-Carlo packing algorithm
    2. Create an initial face mesh in GAMBIT and export the mesh as .msh format
    3. Import the .msh file into TGRID, and remove any Delaunay violating triangular faces
    4. Create the pure tetrahedral mesh using growth functions and remove any Delaunay violations. Smooth the volume mesh and assess the quality, re-meshing if necessary.
    5. Convert the pure tetrahedral mesh into an arbitrary polyhedral mesh using ANSYS Fluent 13.
    6. Import the mesh into OpenFOAM and then decompose the domain according to the number of parallel nodes to be used in the calculation

For the parallel computations, the mesh was converted to an arbitrary polyhedral mesh to reduce the cell count by 81 percent and the face count by 37 percent. Additionally, the renumbering of was used to increase memory access efficiency, by reducing the number of diagonal bands by almost 98 percent. Using the Scotch library was instrumental in decomposing the mesh into separate sub-domains with nearly equal cell and processor patch counts. It can be said with confidence, that the combination of care in creating the mesh and careful planning to decompose the mesh reduced the overall solution time significantly. If more projects are pursued that will leverage OpenFOAM, it is prudent that more investment be made to improve our local cluster or the SEAS cloud.

7.2 Interstitial-Scale Momentum Transport Modeling

- **RANS Models of Momentum Transport:** RANS-based models can provide an adequate description of the steady-state flow-field within packed-beds. This description is extremely complex and highly dependent on boundary conditions, turbulent closure relations (i.e. turbulent viscosity model), mesh structure, equation discretization, and convergence criteria.

- **Three-Dimensional Data Analysis:** Clear differences in near particle flow phenomena were shown for several cases of laminar and turbulent flows. For laminar flow, the fluid moves around the particle creating very little swirl near the sharp edges of the particle. The laminar flow exhibited very uniform flow through the radial direction in the bed, however had many regions of recirculation in-between and downstream from particles. By using the Q-criteria proposed by Hunt et. al [57], the differences in near particle flow environment can be further exaggerated to show additional flow complexities and vortex formations.

- **Contraction of Data:** Complex three-dimensional data was contracted for two primary reasons (1) To improved conceptualization and understanding and (2) to reflect complex information to lower dimension models (e.g. dispersion models, Euler-Euler models, etc.). For the radial profiles of velocity, it was seen that the flow magnitudes in the center of the bed were relatively flat with a drastic peak near the wall. This peak at the wall of course coincided with the rise in porosity from wall effects. In addition the radial profiles of turbulent kinetic energy (k) and turbulent kinetic energy dissipation rate (ϵ) showed that there was a much larger presence of turbulence within the first 1.5 particle diameters near the wall. This rise in turbulence and dissipation would indicate the increase in mixing and energy loss near the wall due to the complex approach and swirling the fluid experiences.

- **Perpendicular Profiles:** Perpendicular profiles were introduced as a novel method to post-process the complex velocity, k, and ϵ fields. The profiles agreed asymptotically with the boundary conditions as well as accepted descriptions of turbulent boundary layers in general. It was seen that the turbulent kinetic energy and dissipation experienced distinct peaks just away from the walls and a sharp decline near the wall, indicating the diffusion and dissipation of turbulence in a laminar sublayer. Overall, the perpendicular profiles can be viewed as a representative boundary layer of the packed-bed, that could be used to refine our viewpoint of thin-film theory in packed-beds and revise one-dimensional models (e.g. mass transfer and thin-film theory).

7.3 Interstitial-Scale Scalar Transport Modeling

- **Reynolds Averaged passive Scalar Equation:** The Reynolds averaged passive scalar equation is capable of providing the ensemble averaged concentration fields for turbulent flows. For Reynolds averaged approaches, scalar-flux closure generally achieved through the gradient-diffusion hypothesis. In this project, the more complex algebraic and scalar-flux transport models gave unphysical results and requires more research and validation arrive at a final conclusion of their utility.

- **Turbulent Schmidt Number and the GDH:** For the work that was performed, there was no sensitivity of the model to turbulent Schmidt within a reasonable range of values for $Sc_t = [0.1, 1]$. What was found is that for lower

Sc_t the number of inner iterations (i.e. iterations to solve the $A\mathbf{x} = \mathbf{b}$ system) over each time-step (outer-iteration). For values above 0.7 there was no significant change in inner-iteration count.

- **Three-Dimensional Data Analysis:** Clear differences in near particle flow phenomena were shown for several cases of laminar and turbulent flows. For laminar flow, the fluid moves around the particle creating very little swirl near the sharp edges of the particle. The laminar flow exhibited very uniform flow through the radial direction in the bed, however had many regions of recirculation in-between and downstream from particles. This characteristic translated to higher dispersion in the age-distribution curve seen during the step-tracer experiments. Most importantly, molecular diffusion to the surface of the particles was a primary mode of transport of species rather than convection. For turbulent flows, the wave front through the bed was defined and convection dominated. The dead-zones behind the particles were not evident in the age-distribution curves, but were seen as an area of design improvement. One such design improvement has been suggested by Nijemeisland [92], requiring holes in the lengthwise direction of the particle to sweep away the dead-zone at the rear of each particle. What has not been addressed is the creation of intense vortexes off of the front of the particles, disrupting the formation of boundary layers over the particle and adjacent particles. By disrupting the formation of the boundary layers over the particles, the characteristic length that a species must diffuse is increased, further limiting already transport limited reaction rates.

- **Residence-Time vs. Tortuosity Effects:** For laminar flows, the tracer front has more time in between changes in direction that might cause mixing. This additional time allows the tracer front to smear, giving way to a more "diffused" fluid-tracer interface. In turbulent flows, the deviation from the flat fluid-tracer front was due to the tortuosity of the complex structure. The direct relationship of tortuosity and the pressure drop in the bed was also alluded to in the previous chapter as well. What can be said is that the convective acceleration (i.e. the change in direction fo the near particle flow fields) around the particle lends itself as the main cause of drag and pressure drop in the bed. Reducing this convective acceleration, while maintaining the complex structure of the bed to move reactants to every face of the particles is key to the improved utilization of packed-bed. For mass transport, these abrupt changes in direction of the fluid adds to the radial and azimuthal spreading of the tracer. A better design be entail a balance between pressure drop and spreading of reactants evenly throughout the bed.

- **Particle Design Improvements:** As mentioned previously, one of the improvements suggested by Nijemeisland [92] was to insert holes along the lengthwise axis of the particle. One of the benefits of of this design would be to "sweep" away the post particle dead-zones. One point that could be made is that by introducing a volume that is not directly in contact with the higher-speed flow outside the particle, the designer runs the risk of creating internal dead-zones. One suggestion from this project is to elongate the particle shape to a form more consistent with a grain of rice. This elongation will reduce the drag of fluid approaching the face of the particle and could reduce post-particle dead-zones. Modeling such a particle will require a reworking of the packing algorithm. It also should be noted that in order to produce an optimized particle shape, the shape must be altered (based on the discussion at hand), repacked, re-meshed, and subjected to fluid simulation. Key objective functions should be identified to enable a more systematic approach to optimization.

7.4 Implementing Sparse Linear System Solvers Based on CUDA in OpenFOAM

- Cufflink is the first library to have multi-GPU linear system solvers that use the domain decomposition in OpenFOAM. No other major CFD software has this capability and this is the first project of its kind.

- Because there is a bottleneck of data transfer from the computer memory to the GPU memory, these transfers of data should be minimized. This implies that CFD solvers in which many outer-iterations are performed will be less efficient than CFD solvers that mostly consist of inner-iterations. For example, it is very common in steady-state Navier-Stokes solvers that decouple pressure and velocity to only converge the velocity fields by a few percent, relax the solution, and then solve pressure, relying on many outer-iterations for overall convergences. In transient solvers, each time step requires many inner iterations in order to produce a conservative answer for that time step and are therefore inner-iteration dominant.

- More test cases need to be run on optimized hardware. The current setup in our lab is not capable of producing adequate results for publication due to hardware limitations. As a result, collaboration between CREL, the University of Adelaide, and Nvidia should continue after this project.

7.5 Overall Comments

In order to continually improve processes in packed beds, particle shape design must be part of an overall process optimization strategy. Shown in this work was the role of tortuosity and packing structure directly on dispersion and pressure drop in packed-bed systems. Through a deeper understanding of both of these phenomena, energy costs can be reduced and particle surface or catalyst utilization will be improved. Ultimately, the success of interstitial-scale modeling depends on computational capabilities and deciding an objective function to optimize within a specific system set by industry. This work presented several suggestions of improving computational speed including using arbitrary polyhedral meshes and hardware acceleration using graphics processing units. With this research, there are still several unanswered questions including that must be addressed in future projects. These questions will be posed a the defense and added in a revision of this document.

Appendix A

Miscellaneous Mathematical Matter

A.1 General Form of Gauss's Flux Theorem

Gauss's theorem is most commonly associated with the divergence operator, as a method to convert the divergence of a vector function $\mathbf{F}$ over a control volume T into a surface integral over the piecewise smooth bounding surface S through:

$$\int\int\int_T \nabla \bullet \mathbf{F} \mathrm{dV} = \int\int_S \mathbf{F} \bullet \mathbf{n}\, \mathrm{dA}. \tag{A.1}$$

$\mathbf{n}$ is the outward normal vector of surface S on element dA [74]. As a more general form of Gauss's Theorem, the inner, outer, and cross product operators (represented as the multiplicative operator $\star$) can be converted from a volume integral to a bounding surface integral using:

$$\int_{V_P} \nabla \star (\mathbf{F}) \mathrm{dV} = \oint_{S_P} \mathrm{d}\mathbf{S} \star \mathbf{F}. \tag{A.2}$$

In this case, d$\mathbf{S}$ is the differential area element, normal to the surface S_P i.e. $\mathbf{n}\,$dA.

A.2 Owner-Neighbor Relationship in FVM

For completeness, Equation 2.5 can be further broken down into the summation of owner and neighbor values at each discrete control volume such that:

$$\sum_{f} F\phi_f = \sum_{\text{owner}} F\phi_f - \sum_{\text{neighbor}} F\phi_f, \tag{A.3}$$

where the owner and neighbor faces for each cell are chosen such that each face has exactly one owner and one neighbor.

A.3 Reynolds Averaging

The details of Reynolds averaging are often left out in basic texts on the subject. The Reynolds averaging of the general case, $\langle \mathbf{u}' \rangle$, and $\langle \mathbf{u}' \langle \mathbf{U} \rangle \rangle$ terms are given briefly.

Appendix B

Programming Related Matter

B.1 The Cufflink Library

Cuda For FOAM Link (cufflink) is an opensource library (http://code.google.com/p/cufflink-library) for linking numerical methods based on Nvidia's Compute Unified Device Architecture (CUDA$^{\text{TM}}$) C/C++ programming language and OpenFOAM®. Currently, the library utilizes the sparse linear solvers of Cusp and methods from Thrust to solve the linear Ax = b system derived from OpenFOAM's lduMatrix class and return the solution vector. Cufflink is designed to utilize the course-grained parallelism of OpenFOAM® (via domain decomposition) to allow multi-GPU parallelism at the level of the linear system solver.

Please note that cufflink is not approved or endorsed by Silicon Graphics International Corp., the owner of the OpenFOAM® trademark and producer of OpenFOAM® software.

B.1.1 Multi-GPU BLAS Level 1 Operations

This appendix is meant to further define the BLAS level 1 functions in the CUFFLINK library.

Reference to the gpuSumProd(vector,vector) function

The vector dot product across multiple GPUs is simply the sum of the local vector dot products on each of the GPU nodes. In this algorithm, the index i refers to the local vectors stored in the memory of each GPU node from 0 to N nodes.

Algorithm 6: Multi-GPU Vector Dot Product

```
1 Compute: $\mathbf{a} \cdot \mathbf{b}$;
2 $S_i = a_{1,i}b_{1,i} + a_{2,i}b_{2,i} + ... + a_{n,i}b_{n,i}$;
3 $\alpha = \sum_{i=0}^{N} S_i$ ;
4 broadcast $\alpha$ to all nodes;
```

It is important to note here that the dot product of the two vectors (line 2) is a preexisting function defined in CUSP (`cusp::blas::dotc(a, b)`), optimized for the GPU, and not rewritten. Additionally, the values of S_i on line 3 are summed and accumulated on the master node (usually node zero) with the MPI function `MPI_Reduce(...)`. The collected result is then sent out to all of the nodes for later use with the command `MPI_Bcast(...)`. Though seemingly simple, this was not implemented in the CUSP library a the time of use.

Reference to the gpuSumMag(vector) function

The calculation of the vector magnitude across multiple GPUs is simply the sum of the Euclidean norm ($M = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$) of the local vector on each of the GPU nodes. In this algorithm, the index i is the local vector stored in the memory of each GPU node from 0 to N nodes.

Algorithm 7: Multi-GPU Euclidean Vector Norm

```
1 Compute: $||\mathbf{a}||$;
2 $M_i = \sqrt{a_{1,i}^2 + a_{2,i}^2 + ... + a_{n,i}^2}$;
3 $\beta = \sum_{i=0}^{N} M_i$ ;
4 broadcast $\beta$ to all nodes;
```

It is important to note here that the Euclidean norm of the local vector (line 2) is a preexisting function defined in CUSP (`cusp::blas::nrm2(a)`), optimized for the GPU, and not rewritten. Additionally, the values of the local norm (M_i) on line 3 are summed and accumulated on the master node (usually node zero) with the MPI function `MPI_Reduce(...)`. The collected result is then sent out to all of the nodes for later use with the command `MPI_Bcast(...)`.

Reference to the gpuAverage(vector) function

The calculation of the average of a vector across multiple GPUs is simply the sum of the local vector components on each of the GPU nodes collected and then divided by the total number of entries in of all the local vectors. In this algorithm, the index i is the local vector stored in the memory of each GPU node from 0 to N nodes, where each vector has n_i components.

Algorithm 8: Multi-GPU Vector Average

```
1 Compute: $\bar{\mathbf{a}}$;
2 $M_i = a_{1,i} + a_{2,i} + ... + a_{n,i}$;
3 $A = \sum_{i=0}^{N} M_i$ ;
4 $T = \sum_{i=0}^{N} n_i$ ;
5 $\bar{\mathbf{a}} = A/T$ ;
6 broadcast $\bar{\mathbf{a}}$ to all nodes;
```

Line 2 in the algorithm is a preexisting routine in the TRHUST library, called with `thrust::reduce(a.begin(),a.end())`. Line 3 is the reduction using MPI (`MPI_Reduce(...)`) and the final result on line 5, producing $\bar{\mathbf{a}}$, is simple arithmetic. The collected result is then sent out to all of the nodes for later use with the command `MPI_Bcast(...)`.

B.2 Courant Number Definitions

The cell Courant number is very generically defined as

$$Co_{cell} = \frac{||\mathbf{U}||\delta t}{\delta x}, \tag{B.1}$$

where δt and δx refer to the time-step size and spatial discretization width. The Courant number is used as a time-step size control method since there are practical and stability limitations (for explicit schemes) with time-step size that are system dependent. Equation 5.16 is adequate when a finite difference formulation is used, however is not sufficient for a finite-volume formulation. Because the finite-volume method is primarily concerned with cell-face values and face-fluxes, a more constructive formulation is

$$Co_{ext} = \max\left(\frac{1}{\mathbf{S}_f}\frac{||\mathbf{U}_f \cdot \mathbf{S}_f||}{d_f}\right)\delta t \tag{B.2}$$

$$Co_{sgi} = \frac{1}{2}\max\left(\frac{\sum_f ||\mathbf{S}_f \cdot \mathbf{U}_f||}{\delta V}\right)\delta t. \tag{B.3}$$

As you can see, there are several definitions of the Courant number. Each of the two presented above are in different versions of the OpenFOAM library. This research project used the SGI version of OpenFOAM.

Appendix C

Additional Background in Transport Phenomena Concepts

C.1 The k-ϵ Model

As noted by Wilcox, turbulence is a continuum phenomena in which the length scale of the smallest turbulent eddie (i.e. the Kolmogorov length) is orders of magnitude above molecular scales [133]. Furthermore, due to the random nature of the fluid motion, we are limited in our pursuit of an exact representation[17] to seeking a statistical amalgamation of the flow. For this project the manifestation of this statistical representation is in the form of the Reynolds averaged Navier-Stokes equations (Equation 4.7).

As previously mentioned, one method to close the Reynolds stress term $\left\langle u_i' u_j' \right\rangle$ in Equation 4.7, is to use the Boussinesq eddy viscosity hypothesis in Equation 4.8. The Boussinesq eddy viscosity hypothesis seeks to lump continuum phenomena of turbulence into a turbulent viscosity (ν_t), similar to that of lumping molecular effects in viscosity. Though this may be the main fault of turbulent viscosity methods, the Boussinesq eddy viscosity hypothesis seems to correctly (or rather adequately) account the dissipation of energy through the effects of cascading, giving local information about turbulence [78]. One such class of models for determining the turbulent viscosity are two-equation models such as $k - \epsilon$.

[17]Though we can use direct numerical simulation (DNS) to get an "exact" flow field for velocity, it is neither practical nor possible in most research laboratories due to lack of computational resources.

Models based on turbulent kinetic energy (k) are abundant and are used widely across numerous engineering fields [133]. The adoption of the turbulent kinetic energy (k) as a primary turbulence variable was introduced by Kolmogorov in 1942, giving rise to subsequent models that proposed a second transport equation based on a mixing length variable $z = k^n l^m$, where m and n are constants [78]. Launder notes that ϵ was initially favored as a second equation due to the relative ease at which a second transport equation could be derived and that ϵ already appears as an unknown in the transport equation for k [78]. Nonetheless, conceptualizing ϵ as a rate of turbulent energy dissipation seen in Kolmogorov's Theory or as a mixing length variable, has proven to be of tremendous use for the fluid modeling community through its wide acceptance.

With such a wide variety of choices of turbulence models, one must choose carefully especially in cases where experimental data is lacking. Previous researchers in the field of interstitial-scale packed-bed modeling have used a variety of turbulence models for various reason (Table 1.1). However, the overall conclusion one could make from the variety of models used in interstitial-scale modeling is that 1) The choice of turbulence model is highly specific for heat, mass, and momentum transport conditions; 2) Given a set of experimental validation guidelines (e.g. Pressure Drop, overall heat transfer, etc.), there are several "valid" turbulence models (when used properly) that will yield results with good agreement; 3) None of the models are correct if a more rigorous validation was used. Considering the fact that a model is only as valid as the experimental method used to validate it, the only true method outside of experimentation is the perform DNS.

From the perspective of using DNS to validate results, Sarkar [114] delivered an interesting discussion of low-Reynolds number $k-\epsilon$ models for wall bounded flows (Turbulent Couette flow). With comparison to the Wilcox $k-\omega$ [133] and $k-\tau$ model of Speziale [118], low-reynolds number models designed to agree asymptotically (i.e. dissipate turbulence in the viscous sublayer) were more successful at matching DNS data near the wall and in the bulk. Those not designed to asymptotically agree with the boundary layer, yielded uniform values across the channel and not at the walls, with those designed for wall bounded flows performed better. While not a surprising result, it does bring up the importance of the fact that turbulence models designed for wall bounded flows should be used for wall dominated domains (i.e. packed beds). Considering that interstitial-scale packed-bed simulations are dominated by wall bounded flows and wall shear and that the boundary layer thickness in these simulations is small compared to length scales of the particles, it would be extremely difficult to validate such results experimentally and extremely important to choose the right model.

C.2 The Turbulent Schmidt and Prandtl Numbers

Throughout the last century the turbulent Schmidt (Sc_t) and Prandtl (Pr_t) numbers have played a key role in turbulent passive scalar transport modeling that utilize the gradient diffusion hypothesis (5.6). Specifically, Sc_t or Pr_t are used to relate the turbulent mass or thermal diffusivity in 5.6, to the turbulent viscosity (ν_t) determined by a turbulent-viscosity model (4.8) in the RANS or URANS equations. This leads to the key question of what determines Sc_t and Pr_t and why are they important? This is briefly addressed in the following section.

In general, Sc_t or Pr_t is treated as a global parameter in complex fluid simulations usually set to a default value of 0.7 or unity. The treatment of the turbulent Schmidt and Prandtl number in this sense is ubiquitous, being used in both open source[132] and commercial[58] CFD packages; accepted in scientific literature[63, 82, 46, 126]; and detailed in the scientific literature by Pope[105], Ranade[107], and Fox[45] devoted to the subject of reacting turbulent flows and chemical reaction engineering. Estimates of a global turbulent Schmidt number in homogeneous turbulent flows were discussed thoroughly by Corrsin[34] by comparing Taylor microscales for the scalar ϕ' and velocity u' components. Noted by Corrsin, values of Sc_t were approximately unity or less, in agreement with the work of Batchelor[10, 12]. Although the results of Corrsin were valuable for homogeneous turbulence, inhomogeneous turbulence is often encountered in practice and requires transport equations to model its effects. Nevertheless, the use of a constant global turbulent Schmidt number in packed bed simulations[46], urban diffusion problems[126], and combustion modeling[13, 83] is seen throughout computational fluid dynamics, ranging in values from 0.1 to 2.2.

This large range alludes to the fact that prescribing a global value is problem dependent. In all cases, a sensitivity study of a particular variable correlated to Sc_t should be performed[83].

In contrast to the use of a global Sc_t and Pr_t to lump the complex relationship of turbulent fluctuations and turbulent mixing through values of ν_t and D_t or α_t, very little work in the last few decades has been done to elucidate the factors influencing Sc_t and Pr_t. Addressed in a 1974 article by A.J. Reynolds[109], the previous 25 years of turbulent Schmidt ($Sc_t = \nu_t/D_t$) and Prandtl ($Pr_t = \nu_t/\alpha_t$) number predictions were reviewed. The discussion was restricted to temperature and concentration ranges that do not significantly interact with the flow (i.e. passive scalar flows and level-1 mixing), and assumed an almost perfect analogy between Pr_t and Sc_t numbers, with minor exception in gases. This exception is based on the differences between actual and assumed enthalpy fluxes seen in the flows, where they are negligible and within measured experimental variability. Reynolds noted that Pr_t and Sc_t depended on the corresponding molecular value of Pr and Sc, the position within the flow (i.e. distance from the wall), and the local turbulent intensity of the flow in question. In general, Reynolds summarized these observations into a single general formula

$$Sc_t = C_1 \exp\left[-C_2 Sc^m \left(\nu_t/\nu\right)^n\right], \tag{C.1}$$

where C_1, C_2, m and n are all positive constants. In C.1, the position within the flow is taken into account implicitly through the ν_t term, which is spatially variable and highly influenced by the wall and turbulent boundary layer. C.1 is consistent with the observation that Sc_t decreases as ν_t/ν increases, indicating greater turbulent mixing in more turbulent regions. For ν_t/ν ratios approaching zero, Sc_t approaches C_1. This is inconsistent with the definition of $Sc_t = \nu_t/D_t$, as Sc_t should be undefined since ν_t and D_t equal zero in laminar regions. The boundedness of C.1 in laminar regions makes it's use practical from a modeling perspective, albeit incorrectly overstating turbulent mass flux in the laminar sublayer of boundary layers. Finally, all of the relations presented by Reynolds classified the Sc_t and Pr_t into Prandtl mixing length based models, simple empiricism, and statistical calculations. Reynolds concluded that the mixing length models were not fundamentally advantageous since they do not elucidate transport phenomena, but are practical in the sense they are easy to use and contain realistic information. The remaining models were neither practical nor fundamentally advantageous since they merely served to "remind us how little confidence can be placed in any limited group of measurements"[109]. Though the mixing length models were considered most desirable at the time, Reynolds concluded that more work should be done to account for position from the wall separately from the turbulent intensity.

Since the review of A.J. Reynolds[109], further research related to the prediction turbulent Schmidt and Prandtl numbers has been more empirical. Work by Jischa[65] described Pr_t and Sc_t using simple empirical correlations

$$\begin{aligned} Pr_t &= A + B/Pr \\ Sc_t &= A + B/Sc, \end{aligned} \tag{C.2}$$

with no influence of wall distance or turbulent intensity taken into account. A review by Kays[69] examined experimental data on the turbulent Prandtl number for two dimensional turbulent boundary layer flows in circular pipes or flat ducts. Kays discussed empirical models based on DNS and experimental data to predict Pr_t in different regions of the boundary layer for various fluids (i.e. air, water, oil, liquid metals). The conclusions by Kays further support the conclusions by A.J Reynolds[109] in that the Pr_t and Sc_t depend on molecular Pr and Sc and the distance from the wall. More recent work by Koeltzsch[72] investigated height dependence of Sc_t in boundary layers through wind tunnel experiments. As a result, a power series approximation of Sc_t was used to fit experimental data only to further show the wall dependence of Sc_t in turbulent boundary layer flows. In a different approach, Guo et al.[55] applied the genetic algorithm to optimize a variable Sc_t proportional to three constants to control the magnitude of the Sc_t, the relative importance of turbulent frequency scale, and the affect of asymmetry in the stresses to agree with experimental results for a jet in crossflow. Validated against three different cases, Guo et. al. showed qualitative and quantitative agreement with previously published data. For completeness, a comparison against cases of constant global Sc_t was performed. The work by Guo et al. addresses the need for new spatially and temporally variable Sc_t (rather than a global constant Sc_t) and shows that a variable turbulent Schmidt number fits numerical data to experimental data better than using a constant Sc_t, but fails to elucidate phenomena influencing the Sc_t in turbulent flows.

The overall importance of the Sc_t and Pr_t in CFD is that the value of these two numbers strongly influence the level of turbulent mixing in the modeled system. If small global Sc_t and Pr_t numbers are assumed then large turbulent mass or thermal diffusivity is defined. For instance in combustion, a lower Sc_t intensifies combustion due to enhanced species diffusion and turbulent mixing, while a higher Sc_t may create mixtures that are unable to sustain combustion[13]. Without a more fundamental understanding of the Sc_t and Pr_t numbers, more uncertainty is introduced into CFD simulations.

The implications of an improved ability to predict Pr_t and Sc_t reach beyond CFD to more classical fields of mass and heat transfer correlations using Sherwood and Nusselt numbers. For the turbulent Prandtl, a 2000 review by Churchill[24] noted that "The development of a comprehensive predictive or correlative expression for the turbulent Prandtl number is the principle remaining challenge with respect to the prediction of turbulent forced convection." This can be equally stated for the turbulent Schmidt number in highly complex, boundary layer, and multiscale flows. A. Dudukovic and Pjanovic[42] found that "it is precisely this assumption of $Sc_t = constant$ that leads to misrepresentation of the Reynolds number dependence", referring to a Sherwood number correlation

$$Sh = a_1 Re\sqrt{fSc/Sc_t}, \tag{C.3}$$

for boundary layers in pipe flows and falling films. The conclusions by A. Dudukovic and Pjanovic further stated that Sc_t is not constant and is dependent on turbulent spectra, agreeing with conclusions by A.J. Reynolds[109] concerning the importance of the turbulent intensity in predicting Sc_t.

C.2.1 Churchill's Reinterpretation and A Look to the Future

Reexamination from a different perspective often leads to improvements in fundamental knowledge and understanding. This is true with respect to the turbulent Prandtl number in a published work by Churchill[25] in 2002. Churchill provided a reinterpretation of the turbulent Prandtl number, for a fully-developed turbulent flow in a round tube, in terms of a local fraction of shear stress and fraction of heat flux density due to fluctuations in velocity. The essential steps in the derivation require the stresses in the radial direction, the transport of energy in the negative radial direction, and a formulation of dimensionless shear and heat flux density. As a result, Churchill was able to formulate

$$\frac{Pr_t}{Pr} = \frac{(\overline{u'v'})^{++}(1-(\overline{T'v'})^{++})}{(\overline{T'v'})^{++}(1-(\overline{u'v'})^{++})}. \tag{C.4}$$

C.4 expands the Prandtl numbers in terms of dimensionless shear stresses

$$(\overline{u'v'})^{++} = \frac{-\rho \left\langle u_i' u_j' \right\rangle}{\tau_{ij}}, \tag{C.5}$$

and dimensionless heat flux density

$$(\overline{T'v'})^{++} = \frac{\rho c \left\langle T' u' \right\rangle}{q}, \tag{C.6}$$

where τ_{ij} is the total shear in the radial direction, q is the heat flux density in the y-direction, and ρ and c are the fluid density and heat capacity respectively.

C.4 is an important starting point for relating Prandtl numbers in fully developed turbulent flow in channels and pipes. One can see the possibilities of extending this methodology to spatially variable Pr_t (or Sc_t) in more complex geometries. The crux of using such an approach is the determination of $(\overline{u'v'})^{++}$ and $(\overline{T'v'})^{++}$. Based on the work by Papavassiliou and Hanratty[97], Churchill noted that a Lagrangian form of DNS provided an accurate calculation of these quantities without empiricism. Also, a recent work by Srinivasan and Papavassiliou[119] explored the use of Lagrangian DNS to determine Pr_t for classical Poiseulle channel and plane Couette flows. Pr_t was found to be a function of Pr, which agrees with the observations presented earlier in this paper. The extension of determination of Pr_t for curved geometries (where transport is not necessarily unidirectional) remains an open challenge and is the subject of ongoing research in our laboratory that will be addressed in upcoming publications.

References

[1] gnu.org. http://www.gnu.org/copyleft/gpl.html.

[2] ERCOFTAC special interest group on "Quality and trust in industrial CFD", January 2000.

[3] Openfoam-1.5-dev, 2010.

[4] Ramesh Agarwal. COMPUTATIONAL FLUID DYNAMICS OF WHOLE-BODY AIRCRAFT. *Annual Review of Fluid Mechanics*, 31(1):125–169, January 1999.

[5] Harten Ami. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 135(2):260–278, August 1997.

[6] Rutherford Aris. *Vectors, Tensors and the Basic Equations of Fluid Mechanics.* Dover Publications, January 1990.

[7] Abdelouahab Attou and Gilles Ferschneider. A two-fluid hydrodynamic model for the transition between trickle and pulse flow in a cocurrent gas-liquid packed-bed reactor. *Chemical Engineering Science*, 55(3):491511, February 2000.

[8] M.J. Baker, P.G. Young, and G.R. Tabor. Image based meshing of packed beds of cylinders at low aspect ratios using 3d MRI coupled with computational fluid dynamics. *Computers &amp; Chemical Engineering*, 35(10):1969 – 1977, 2011.

[9] G. K. Batchelor. Diffusion in a field of homogeneous turbulence. i. eulerian analysis. *Australian Journal of Scientific Research A Physical Sciences*, 2:437, December 1949.

[10] G. K. Batchelor. Small-scale variation of convected quantities like temperature in turbulent fluid part 1. general discussion and the case of small conductivity. *Journal of Fluid Mechanics*, 5(01):113–133, 1959.

[11] G. K. Batchelor. *The Theory of Homogeneous Turbulence.* Cambridge University Press, June 1982.

[12] G. K. Batchelor, I. D. Howells, and A. A. Townsend. Small-scale variation of convected quantities like temperature in turbulent fluid part 2. the case of large conductivity. *Journal of Fluid Mechanics*, 5(01):134–139, 1959.

[13] R. A. Baurle. Modeling of high speed reacting flows: established practices and future challenges. *AIAA*, 267:42, 2004.

[14] M. Behnam, A.G. Dixon, M. Nijemeisland, and E.H. Stitt. Catalyst deactivation in 3D CFD resolved particle simulations of propane dehydrogenation. *Ind. Eng. Chem. Res*, 49(21):1064110650, 2010.

[15] N. Bell and M. Garland. Efficient sparse matrix-vector multiplication on CUDA. Technical report, NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, 2008.

[16] Nathan Bell and Michael Garland. Cusp: Generic parallel algorithms for sparse matrix and graph computations, 2010.

[17] R. Byron Bird, Warren E. Stewart, and Edwin N. Lightfoot. *Transport Phenomena, 2nd Edition*. Wiley, 2 edition, July 2001.

[18] R. Blaak, D. Frenkel, and B. M Mulder. Do cylinders exhibit a cubatic phase? *The Journal of Chemical Physics*, 110:11652, 1999.

[19] J. Boussinesq. Thorie de lcoulement tourbillant. *Mem. Prsents par Divers Savants Acad. Sci. Inst. Fr*, 23:46–50, 1877.

[20] P. Cardiff, A. Ivankovic, D. FitzPatrick, R. Flavin, and A. Karac. Contact stress analysis in OpenFOAM: application to hip joint bones. 2011.

[21] R. Caulkin, X. Jia, C. Xu, M. Fairweather, R. A. Williams, H. Stitt, M. Nijemeisland, S. Aferka, M. Crine, A. Leonard, D. Toye, and P. Marchot. Simulations of structures in packed columns and validation by x-ray tomography. *Industrial & Engineering Chemistry Research*, 48(1):202–213, January 2009.

[22] Richard Caulkin, Michael Fairweather, Xiaodong Jia, Richard A. Williams, W. Marquardt, and C. Pantelides. Validation of a digital packing algorithm for the packing and subsequent fluid flow through packed columns. In *16th European Symposium on Computer Aided Process Engineering and 9th International Symposium on Process Systems Engineering*, volume Volume 21, pages 395–400. Elsevier, 2006.

[23] Richard Caulkin, Xiaodong Jia, Mike Fairweather, and Richard A. Williams. Lattice approaches to packed column simulations. *Particuology*, 6(6):404–411, December 2008.

[24] Stuart W. Churchill. Progress in the thermal sciences:AIChE institute lecture. *AIChE Journal*, 46(9):1704–1722, September 2000.

[25] Stuart W. Churchill. A reinterpretation of the turbulent prandtl number. *Industrial & Engineering Chemistry Research*, 41(25):6393–6401, December 2002.

[26] Daniel P. Combest, Palghat Ramachandran, and Milorad P. Dudukovic. Micro-scale CFD modeling of packed-beds. In *6th OpenFOAM Workshop*, Penn State University, USA, June 2011.

[27] Daniel P. Combest and Palghat A. Ramachandran. Micro-scale CFD modeling of packed-beds. In *American Institute of Chemical Engineers Annual Meeting: Computational Fluid Dynamics in Chemical Reaction Engineering*, Salt Lake City, UT, October 2010.

[28] Daniel P. Combest and Palghat A. Ramachandran. Nvidia professor partnership proposal: Implementing fast linear algebraic system solvers for OpenFOAM using CUSP and THRUST. Technical report, Washington University, St. Louis, MO, August 2010.

[29] Daniel P. Combest, Palghat A. Ramachandran, and Milorad P. Dudukovic. Implementing fast parallel linear system solvers in OpenFOAM based on CUDA. In *6th OpenFOAM Workshop: Optimization, HPC, and Pre- and Post-Processing I Session*, Penn State University, USA, June 2011.

[30] Daniel P. Combest, Palghat A. Ramachandran, and Milorad P. Dudukovic. On the gradient diffusion hypothesis and passive scalar transport in turbulent flows. *Ind. Eng. Chem. Res.*, 50(15):8817–8823, 2011.

[31] John Horton Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 2nd edition, November 1992.

[32] NVIDIA Corporation. *Nvidia CUDA C Programming Guide: Version 4.0*. May 2011.

[33] NVIDIA Corporation. Nvidia GPUDirectTM technology overview, 2011.

[34] Stanley Corrsin. The isotropic turbulent mixer: Part II. arbitrary schmidt number. *AIChE Journal*, 10(6):870–877, November 1964.

[35] Bart J. Daly and Francis H. Harlow. Transport equations in turbulence. *Physics of Fluids*, 13:2634–2649, November 1970.

[36] P. V. Danckwerts. *GasLiquid Reactions*. McGraw-Hill Book Company, January 1970.

[37] Arno de Klerk. Voidage variation in packed beds at small column to particle diameter ratio. *AIChE Journal*, 49(8):2022–2029, August 2003.

[38] I. Demirdi and S. Muzaferija. Numerical method for coupled fluid flow, heat transfer and stress analysis using unstructured moving meshes with cells of arbitrary topology. *Computer Methods in Applied Mechanics and Engineering*, 125(1-4):235–255, September 1995.

[39] Paul E Dimotakis. Turbulent mixing. *Annual Review of Fluid Mechanics*, 37(1):329–356, January 2005.

[40] A.G. Dixon, M.E. Taskin, M. Nijemeisland, and E.H. Stitt. CFD method to couple three-dimensional transport and reaction inside catalyst particles to the fixed bed flow field. *Industrial & Engineering Chemistry Research*, 2010.

[41] A.G. Dixon, M.E. Taskin, M. Nijemeisland, and E.H. Stitt. Systematic mesh development for 3D CFD simulation of fixed beds: Single sphere study. *Computers & Chemical Engineering*, 2010.

[42] Aleksandar Dudukovic and Rada Pjanovic. Effect of turbulent schmidt number on mass-transfer rates to falling liquid films. *Industrial & Engineering Chemistry Research*, 38(6):2503–2504, June 1999.

[43] Exa Corporation. Frequently asked questions - physics.

[44] Joel H. Ferziger and Milovan Peric. *Computational Methods for Fluid Dynamics.* Springer, 3rd edition, December 2001.

[45] Rodney O. Fox. *Computational Models for Turbulent Reacting Flows.* Cambridge University Press, December 2003.

[46] S. Frigerio, H. Thunman, B. Leckner, and S. Hermansson. Estimation of gas phase mixing in packed beds. *Combustion and Flame*, 153(1-2):137–148, April 2008.

[47] M. Gan, N. Gopinathan, X. Jia, and R. A. Williams. Predicting packing characteristics of particles of arbitrary shapes. *Kona*, 22:8293, 2004.

[48] M. Garland. Sparse matrix computations on manycore GPU's. In *Proceedings of the 45th annual conference on Design automation*, page 26, 2008.

[49] M. Giese, K. Rottschaefer, and D. Vortmeyer. Measured and modeled superficial flow profiles in packed beds with liquid flow. *AIChE Journal*, 44(2):484490, 1998.

[50] A. Guardo, M. Coussirat, M.A. Larrayoz, F. Recasens, and E. Egusquiza. Influence of the turbulence model in CFD modeling of wall-to-fluid heat transfer in packed beds. *Chemical Engineering Science*, 60(6):1733–1742, March 2005.

[51] Alfredo Guardo, Miguel Coussirat, M. Angels Larrayoz, Francesc Recasens, and Eduard Egusquiza. CFD flow and heat transfer in nonregular packings for fixed bed equipment design. *Industrial & Engineering Chemistry Research*, 43(22):7049–7056, October 2004.

[52] Prashant R. Gunjal, Madhavanand N. Kashid, Vivek V. Ranade, and Raghunath V. Chaudhari. Hydrodynamics of trickle-bed reactors: Experiments and CFD modeling. *Industrial and Engineering Chemistry Research*, 44(16):62786294, 2005.

[53] Prashant R. Gunjal and Vivek V. Ranade. Modeling of laboratory and commercial scale hydro-processing reactors using CFD. *Chemical Engineering Science*, 62(18-20 SPEC ISS):55125526, 2007.

[54] Prashant R Gunjal, Vivek V Ranade, and Raghunath V Chaudhari. Computational study of a singlephase flow in packed beds of spheres. *AIChE Journal*, 51(2):365–378, February 2005.

[55] Yanhu Guo, Guangbin He, and Andrew T. Hsu. Application of genetic algorithms to the development of a variable schmidt number model for jet-in-crossflows. *International Journal of Numerical Methods for Heat & Fluid Flow*, 11(8):744–761, 2001.

[56] Jared Hoberock and Nathan Bell. Thrust: A parallel template library, 2010.

[57] J. C. R. Hunt, A. A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. pages 193–208, December 1988.

[58] Ansys Inc. *Fluent 6.3 user's Guide*. Ansys Inc., September 2006.

[59] Ansys Inc. *Fluent 12.1 user's Guide*. Ansys Inc., October 2009.

[60] H. Jasak and H. G Weller. Application of the finite volume method and unstructured meshes to linear elasticity. 1998.

[61] Hrvoje Jasak. *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. PhD thesis, Imperial College of Science, Technology and Medicine, London England, June 1996.

[62] Tao Jia, Yuwen Zhang, and J.K. Chen. Simulation of granular packing of particles with different size distributions. *Computational Materials Science*, 51(1):172–180, January 2012.

[63] Lei-Yong Jiang and Ian Campbell. Prandtl/Schmidt number effect on temperature distribution in a generic combustor. *International Journal of Thermal Sciences*, 48(2):322–330, February 2009.

[64] Y. Jiang, M.R. Khadilkar, M.H. Al-Dahhan, and M.P. Dudukovic. CFD of multiphase flow in packed-bed reactors: I. k-fluid modeling issues. *AIChE Journal*, 48(4):701715, 2002.

[65] Michael Jischa and Heinz Berend Rieke. About the prediction of turbulent prandtl and schmidt numbers from modeled transport equations. *International Journal of Heat and Mass Transfer*, 22(11):1547–1555, November 1979.

[66] K.R. Jolls and T.J. Hanratty. Transition to turbulence for flow through a dumped bed of spheres. *Chemical Engineering Science*, 21(12):1185–1190, December 1966.

[67] Fabian Peng Karrholm. Rhie-chow interpolation in OpenFOAM, 2006.

[68] Sriganesh R. Karur and P. A. Ramachandran. Solving linear diffusion-reaction networks in porous catalyst particles using BEM. *AIChE Journal*, 42(2):383–390, 1996.

[69] W. M Kays. Turbulent prandtl number. where are we? *ASME Transactions Journal of Heat Transfer*, 116:284295, 1994.

[70] Franklyn J. Kelecy. Using polyhedral cells in fluent 6.3, May 2006.

[71] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low reynolds number. *Journal of Fluid Mechanics*, 177(-1):133166, 1987.

[72] Konrad Koeltzsch. The height dependence of the turbulent schmidt number within the boundary layer. *Atmospheric Environment*, 34(7):1147–1151, 2000.

[73] Vaclav Kolar. Vortex identification: New requirements and limitations. *International Journal of Heat and Fluid Flow*, 28(4):638–652, 2007.

[74] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 10 edition, August 2011.

[75] Zeljko Kuzeljevic. *Hydrodynamics of Trickle Bed Reactors: Measurements and Modeling*. PhD thesis, Washington University, St. Louis, MO, May 2010.

[76] H.M. Kvamsdal, H.F. Svendsen, T. Hertzberg, and O. Olsvik. Dynamic simulation and optimization of a catalytic steam reformer. *Chemical Engineering Science*, 54(13-14):26972706, 1999.

[77] C. K. G. Lam and K. Bremhorst. A modified form of the k-epsilon model for predicting wall turbulence. *Journal of Fluids Engineering*, 103(3):456–460, 1981.

[78] Brian Launder and Dudley Brian Spalding. *Lectures in Mathematical Models of Turbulence*. Academic Press Inc, July 1972.

[79] J.J. Lerou and G.F. Froment. Velocity, temperature and conversion profiles in fixed bed catalytic reactors. *Chemical Engineering Science*, 32(8):853–861, 1977.

[80] V. Levich. *Physicochemical Hydrodynamics.* Prentice Hall, June 1962.

[81] OpenCFD Limited. OpenFOAM programmer's guide version 1.6, July 2009.

[82] A.N. Lipatnikov and J. Chomiak. Effects of premixed flames on turbulence and turbulent scalar transport. *Progress in Energy and Combustion Science*, 36(1):1–102, February 2010.

[83] Ying Liu, Hua Feng, Michael G. Olsen, Rodney O. Fox, and James C. Hill. Turbulent mixing in a confined rectangular wake. *Chemical Engineering Science*, 61(21):6946–6962, November 2006.

[84] S. A Logtenberg, M. Nijemeisland, and A. G Dixon. Computational fluid dynamics simulations of fluid flow and heat transfer at the wall-particle contact points in a fixed-bed reactor. *Chemical Engineering Science*, 54(13-14):2433 2439, 1999.

[85] Rodrigo J. G. Lopes and Rosa M. Quinta-Ferreira. Numerical simulation of trickle-bed reactor hydrodynamics with RANS-Based models using a volume of fluid technique. *Ind. Eng. Chem. Res.*, 48(4):1740–1748, 2009.

[86] Rodrigo J.G. Lopes and Rosa M. Quinta-Ferreira. CFD modelling of multiphase flow distribution in trickle beds. *Chemical Engineering Journal*, 147(2-3):342–355, April 2009.

[87] John L. Lumley and Henk Tennekes. *A First Course in Turbulence.* The MIT Press, March 1972.

[88] Guy B. Marin. *Advances in Chemical Engineering, Volume 31.* Academic Press, 1 edition, December 2006.

[89] Duane Merrill and Andrew Grimshaw. High performance and scalable radix sorting: A case study of implementing dynamic parallelism for GPU computing. *Parallel Processing Letters*, 21:245, 2011.

[90] Gary E. Mueller. Prediction of radial porosity distributions in randomly packed fixed beds of uniformly sized spheres in cylindrical containers. *Chemical Engineering Science*, 46(2):706–708, 1991.

[91] Hubert Nguyen. *GPU Gems 3.* Addison-Wesley Professional, August 2007.

[92] M. Nijemeisland. *Influences of catalyst particle geometry on fixed bed reactor near-wall heat transfer using CFD.* PhD thesis, Worcester Polytechnic Institute, 2003.

[93] M. Nijemeisland and G. D Dixon. Comparison of CFD simulations to experiment for convective heat transfer in a gas-solid fixed bed. *Chemical Engineering Journal*, 82(1-3):231 246, 2001.

[94] Michiel Nijemeisland, Anthony G Dixon, and E. Hugh Stitt. Catalyst design by CFD for heat transfer and reaction in steam reforming. *Chemical Engineering Science*, 59(22-23):5185 5191, 2004.

[95] S.J. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, volume 3, 1998.

[96] Steve Owen. An introduction to mesh generation algorithms, September 2005.

[97] Dimitrios V. Papavassiliou and Thomas J. Hanratty. Transport of a passive scalar in a turbulent channel flow. *International Journal of Heat and Mass Transfer*, 40(6):1303–1311, April 1997.

[98] Suhas Patankar. *Numerical Heat Transfer and Fluid Flow.* Taylor & Francis, 1 edition, January 1980.

[99] S.V Patankar and D.B Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, October 1972.

[100] Marisa N. Pedernera, Juliana Pina, Daniel O. Borio, and Veronica Bucala. Use of a heterogeneous two-dimensional model to improve the primary steam reformer performance. *Chemical Engineering Journal*, 94(1):2940, 2003.

[101] M. Peric and S. Ferguson. The advantage of polyhedral meshes. *Dynamics*, 24:45.

[102] J. W. Perram and M. S. Wertheim. Statistical mechanics of hard ellipsoids. i: Overlap algorithm and the contact function. *Journal of computational physics(Print)*, 58(3):409416, 1985.

[103] John. W. Perram, John Rasmussen, Eigil Praestgaard, and Joel L. Lebowitz. Ellipsoid contact potential: Theory and relation to overlap potentials. *Physical Review E*, 54(6):6565, December 1996. Copyright (C) 2009 The American Physical Society; Please report any problems to prola@aps.org.

[104] Richard Pletcher, John Tannehill, and Dale Anderson. *Computational Fluid Mechanics and Heat Transfer, Second Edition.* Taylor & Francis, 2 edition, April 1997.

[105] Stephen B. Pope. *Turbulent Flows.* Cambridge University Press, 1 edition, January 2000.

[106] P. A. Ramachandran and R. V. Chaudhari. *Three-Phase Catalytic Reactors.* Gordon & Breach Science Pub, 1982.

[107] Vivek Ranade. *Computational Flow Modeling for Chemical Reactor Engineering*. Academic Press, 1st edition, September 2001.

[108] Vivek V. Ranade, Raghunath Chaudhari, and Prahant R. Gunjal. *Trickle Bed Reactors: Reactor Engineering & Applications*. Elsevier, April 2011.

[109] A.J. Reynolds. The prediction of turbulent prandtl and schmidt numbers. *International Journal of Heat and Mass Transfer*, 18(9):1055–1069, September 1975.

[110] L. H. S. Roblee, R. M. Baird, and J. W. Tierney. Radial porosity variations in packed beds. *AIChE Journal*, 4(4):460–464, 1958.

[111] Henrik Rusche. *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fraction*. PhD thesis, Imperial College of Science, Technology and Medicine, London England, December 2002.

[112] Yousef Saad. *Iterative Methods for Sparse Linear Systems, Second Edition*. Society for Industrial and Applied Mathematics, 2 edition, April 2003.

[113] Yousef Saad and Henk A. van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1-2):1–33, November 2000.

[114] A. Sarkar and R.M.C. So. A critical evaluation of near-wall two-equation models against direct numerical simulation data. *International Journal of Heat and Fluid Flow*, 18(2):197–208, April 1997.

[115] OA Saunders and H. Ford. Heat transfer in the flow of gas through a bed of solid particles. *J. Iron Steel Inst*, 141:291, 1940.

[116] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. *URL http://www-2. cs. cmu. edu/jrs/jrspapers. html# cg*, 1994.

[117] G. L.G Sleijpen, H. A Vorst, and D. R Fokkema. BiCGstab (l) and other hybrid bi-CG methods. *Numerical Algorithms*, 7(1):75–109, 1994.

[118] Charles G Speziale, Ridha Abid, and E. C Anderson. A critical evaluation of two-equation models for near wall turbulence. Technical report, June 1990.

[119] Chiranth Srinivasan and Dimitrios V. Papavassiliou. Prediction of the turbulent prandtl number in wall flows with lagrangian simulations. *Industrial & Engineering Chemistry Research*, December 2010.

[120] Alexander Stepanov and Meng Lee. The standard template library. *WG21/N0482, ISO PROGRAMMING LANGUAGE C PROJECT*, 1995.

[121] S. Strobl. FRIEDRICH-ALEXANDER-UNIVERSITT ERLANGEN-NURNBERG.

[122] P. K. Sweby. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 21:995, 1984.

[123] M. Ertan Taskin, Anthony G Dixon, Michiel Nijemeisland, and E. Hugh Stitt. CFD study of the influence of catalyst particle design on steam reforming reaction heat effects in narrow packed tubes. *Industrial and Engineering Chemistry Research*, 47(16):5966 5975, 2008.

[124] M.E. Taskin, A. Troupel, A.G. Dixon, M. Nijemeisland, and E.H. Stitt. Flow, transport, and reaction interactions for cylindrical particles with strongly endothermic reactions. *Industrial & Engineering Chemistry Research*, 2010.

[125] M.E. Taskin, A. Troupel, A.G. Dixon, M. Nijemeisland, and H. Stitt. Intraparticle Diffusion/Reaction modeling for strongly endothermic reactions in low-n tubes with cfd. In *The 2008 Annual Meeting*, 2008.

[126] Yoshihide Tominaga and Ted Stathopoulos. Turbulent schmidt numbers for CFD analysis with various types of flowfield. *Atmospheric Environment*, 41(37):8091–8099, December 2007.

[127] H. van der Vorst. How to write a frequently-cited article. *AUSTRALIAN MATHEMATICAL SOCIETY GAZETTE*, 31(2):94100, 2004.

[128] H. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Prentice Hall, 2 edition, February 2007.

[129] Henk A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, 1 edition, November 2009.

[130] D. Vortmeyer and J. Schuster. EVALUATION OF STEADY FLOW PROFILES IN RECTANGULAR AND CIRCULAR PACKED BEDS BY a VARIATIONAL METHOD. *Chemical Engineering Science*, 38(10):16911699, 1983.

[131] Z. Warhaft. Passive scalars in turbulent flows. *Annual Review of Fluid Mechanics*, 32(1):203–240, January 2000.

[132] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, November 1998.

[133] David C. Wilcox. *Turbulence Modeling for CFD*. Dcw Industries, Incorporated, 3rd edition, November 2006.

[134] H. Yamaguchi. *Engineering Fluid Mechanics*. Springer, 1st edition. edition, November 2010.

[135] B. A Younis, C. G Speziale, and T. T Clark. A rational model for the turbulent scalar fluxes. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 461(2054):575, 2005.