

SIMULATION OF NANOPARTICLE TRANSPORT IN AIRWAYS USING PETROV-
GALERKIN FINITE ELEMENT METHODS

by

Prathish Kumar Rajaraman

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Chemical Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2012

© COPYRIGHT

By

Prathish Kumar Rajaraman

2012

All Rights Reserved

APPROVAL

Of a thesis submitted by

Prathish Kumar Rajaraman

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style and consistency, and is ready for submission to The Graduate School.

Dr. Jeffrey Heys

Approved for the Department of Chemical and Biological Engineering

Dr. Brent Peyton

Approved for The Graduate School

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the library shall make it available to borrowers under rules of the library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Prathish Kumar Rajaraman

April 2011

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Jeff Heys, whose support and encouragement from the very first day I joined the group enabled me to learn and understand various topics in computational fluid dynamics. I also would like to thank all the members of the computational biofluids group, past, present and the future.

Finally, I thank everyone who has stood beside me all this year thru sorrow and joy, which enabled me to be whom I am today.

TABLE OF CONTENTS

1. BACKGROUND	1
LUNG ANATOMY	1
DRUG DELIVERY IN HUMAN AIRWAYS	2
HEALTH IMPACT OF NANOPARTICLES	4
NANOPARTICLE TRANSPORT / DEPOSITION	5
SCOPE OF THE THESIS	7
2. LITERATURE REVIEW	10
AIRWAYS GEOMETRY	10
FLUID PHASE MODELS	12
NANOPARTICLE TRANSPORT MODELS	14
MODEL VALIDATIONS	20
3. METHODS	22
INCOMPRESSIBLE NAVIER-STOKES	22
ADVECTION-DIFFUSION	24
FINITE ELEMENT METHOD (FEM)	26
DISCRETIZATION OF THE NAVIER-STOKES EQUATIONS	27
DISCRETIZE ADVECTION-DIFFUSION	30
ELEMENTS	31
BARICENTRIC COORDINATES AND ISOPARAMETRIC COORDINATES	33
SHAPE FUNCTION	36
SHAPE FUNCTION DERIVATIVES	38
INTEGRATION RULE	38
TEMPORAL DISCRETIZATION	39
STABILIZED ADVECTION DIFFUSION	42
IMPLEMENTATION	44
MODEL VALIDATION	46
4. RESULT AND DISSCUSION	51
GRAETZ SOLUTION	51
HUMAN AIRWAYS SIMULATION	59
5. CONCLUSION AND FUTURE WORK	69
REFERENCES	72

TABLE OF CONTENTS - CONTINUED

APPENDIX A: HUMAN AIRWAYS MODEL IMPLEMENTATION IN C++77

LIST OF TABLES

Table	Page
1. Relationship of particle diameter and dimensionless diffusion.	60

LIST OF FIGURES

Figure	Page
1. Sizes of common material of micro and nano scale	3
2. General structure of human airways and their characteristic	3
3. Different mechanisms of particle deposition in human airways	6
4. Human airways model where B stands for the bifurcation	11
5. Velocity profiles in a human airway model and the different cross sections	13
6. Validated model results of DF (%) as a function of particle diameter	16
7. DEF of nanoparticles for varying flow rates for a human airway model	16
8. DEF of nanoparticle in human airways for a steady flow rate	17
9. Varying DF for different flow rate and particle diameter in a human airways	18
10. Empirical fit for the deposition efficiency in humans	20
11. Control volume with cross flow	24
12. Example of a tetrahedron with 15 nodes	32
13. Isoparametric mapping of curved element onto a standard element	34
14. L_2 error for BDF2 and CN with varying τ parameter.	39
15. Computational time for backward step problem with different solvers	46
16. Computational time for flow around a cylinder problem with different solvers.....	46
17. Vector solution for backward step problem, $t = 2$	47

LIST OF FIGURES - CONTINUED

Figure	Page
18. Vector solution and concentration for flow around a cylinder problem.....	48
19. Concentration solution (a) and vector solution (b) for flow in a cylinder.....	49
20. Meshed cylinder with 4532 elements, which was used for the Graetz solution....	51
21. L_2 error for different stabilization methods using BDF-2 time stepping scheme	52
22. L_2 error for CFEM without any stabilization.	52
23. % Improvement of FEM when applying stabilization with varying diffusion	53
24. L_∞ error when varying number of elements in mesh, when $D = 0.001$	55
25. Simulation results for $D = 0.01$, with various stablization.....	56
26. Simulation results for $D = 0.000001$, with various stablization.....	57
27. Human airways mesh from G0-G3 with approximately 100000 elements.....	59
28. Computational time relationship with number of elements in a mesh.....	61
29. DF as function of number of elements for SUPG stabilization with $D = 10^{-5}$	62
30. DF as function of particle diameter for SUPG and GLS,with 100000 elements...63	63
31. Velocity field for parabolic inlet velocity profile with $Re = 200$	64
32. Concentration distribution for $d = 1(nm)$, with SUPG and GLS	66
33. Concentration distribution for $d = 150(nm)$,with SUPG and GLS.....	66

ABSTRACT

Nanoparticles with various diameters, i.e. $1 \text{ nm} \leq d \leq 150 \text{ nm}$, were studied with respect to their transport and deposition properties in the human airways. A finite element code, written in C++, was developed that solved both the Navier-Stokes and Advection-Diffusion equation monolithically. When modeling nanoparticles, the regular finite element method becomes unstable, and, in order to resolve this issue, various stabilization methods were considered including Streamline Upwind, Streamline Upwind Petrov-Galerkin and Galerkin Least Square. In order to validate the various types of stabilization, the stabilized finite element solution was compared to the analytical Graetz solution. The comparison was done by calculating an approximation of the L_2 - error, and the best stabilization method was found to be Galerkin Least Square. Also in this thesis, we found that the Crank-Nicolson time stepping scheme is not the best option for the human airways simulations problem, and this is due to both the complex nature of the geometry and the Crank-Nicolson method lacks the ability to damp out error when the problem is advection dominated. However, using Crank-Nicolson in straight tube geometry with various stabilization methods provides better accuracy than other second-order time stepping schemes, such as BDF-2. The type of stabilization method used when $d < 10 \text{ nm}$ does matter since Streamline Upwind Petrov-Galerkin introduces higher deposition fraction compared to Galerkin Least Square. This statement is not true when $d > 10 \text{ nm}$, since mesh refinement is important at this range. In the human airways simulation, we found that for $d = 1 \text{ nm}$ the concentration distribution is uniform compared to $d = 150 \text{ nm}$, where localized concentration exists. This implies a potential health risk when inhaling nanoparticles because nanoparticles have a very high surface area and the potential for exposure is much greater. The stabilization methods tested in this thesis show promise for modeling nanoparticle transport in the human airways.

CHAPTER 1

BACKGROUND

Application of nanoparticles in the medical field is an important area of study, especially in targeted drug delivery for the human airways. This application shares the same governing equations with other applications in other fields because the nanoparticle transport and deposition are determined by particle characteristics, flow pattern and the geometry [1-7]. There have been many studies performed on micro-particle transport in the human airways, but very few studies have been done on nanoparticles in human airways. This is due in part to the failure of methods such as the Finite Element Method (FEM) to capture the actual physics of the problem in a numerically stable manner.

Lung Anatomy

The lung's primary job is to exchange oxygen and carbon dioxide between blood and the respiratory gas [1-7]. Modeling the human airway poses several challenges because the exact geometry is not known due to the massive number of branching generations [1-7]. In this thesis we use a very simple model geometry of the human airway, which is Weibel's model [1-7]. Even though other studies have shown that the Weibel model is not accurate [3], it is still very popular among many researchers [7-12]. This will be discussed in much detail in later chapters.

The human airways are composed of two very important regions: the conducting region and respiratory region. The conducting region is built by several sub-sections

which start from the nasal cavity, pharynx, larynx, trachea, bronchi, and bronchioles. The branches in the human airways divide in a very specific way; the main branch divides into two sub-branches that are smaller in diameter and shorter in length [1-7]. One feature of the human airways, which makes them very efficient, is that at the end of every branch point, the branches are doubled and this causes an exponential increase in the interface area. The conducting region, which is shown in figure 2, is from branches 1 to 16; this range is only from the trachea onwards.

The gas exchange happens in the respiratory region, which is also shown in figure 2; the branches/generation are from 17 to 23. The number of branches may vary from 6 to 30 before the respiratory region of the bronchioles, alveolar ducts, and alveolar sacs are reached. The alveolar epithelial in the respiratory zone is where the gas-blood barrier exists and the gas exchange happens. The surface area of the alveolar epithelial is huge because of its thinness (0.1-0.5 μm), and it allows gas exchange by passive diffusion. This thesis will not cover the workings of the human airways in much depth, since the primary focus is to model nanoparticle transport in the human airways.

Drug Delivery in Human Airways

A particle is characterized as a nanoparticle when one of the dimensions is less than 100 nm or 0.1 μm , and new applications for nanoparticles, which require controlling matter at near atomic levels, have become an area of great potential for growth. There have been concerns raised against using nanoparticles since there could be unforeseen

health hazards [6]. Figure 1 shows size comparisons of nanoparticles relative to other well know materials such as protein and DNA.

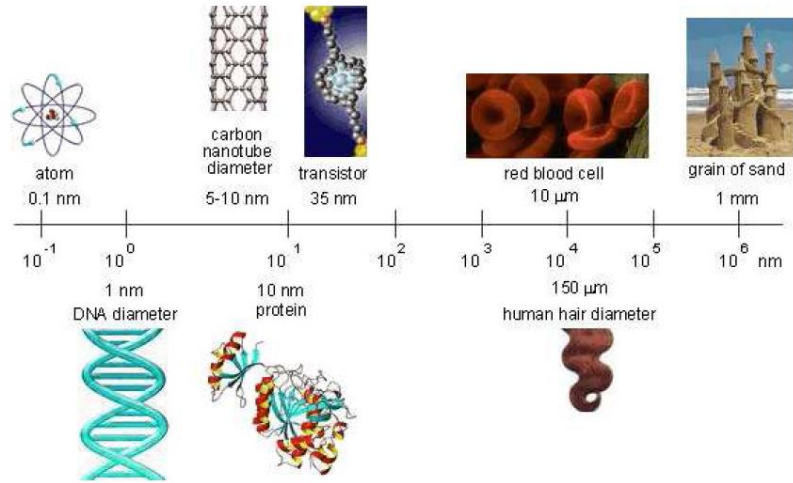


Figure 1. Sizes of common material of micro- or nano-scale (used without permission) [6].

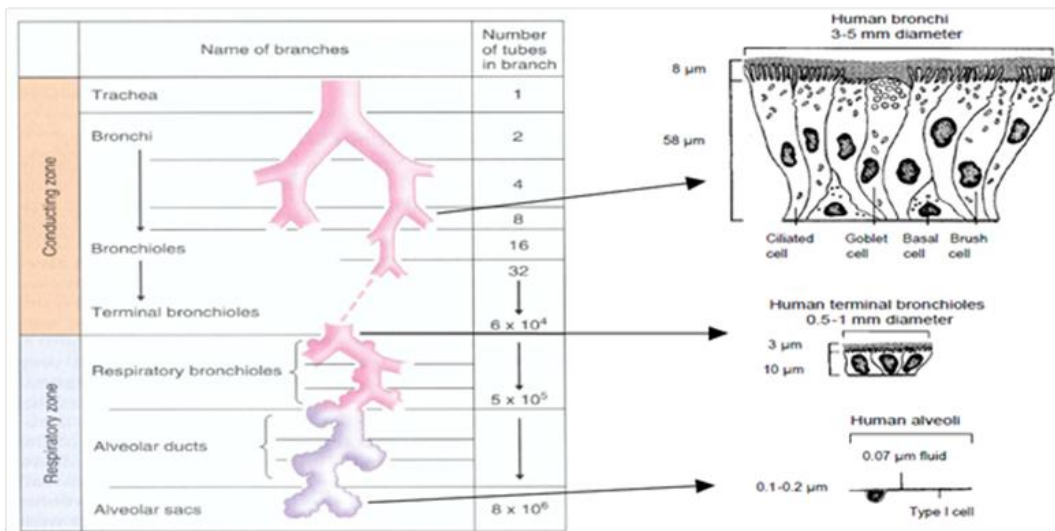


Figure 2. General structure of human airways and their characteristics (used without permission) [7].

Health Impact of Nanoparticles

There are several ways nanoparticles may enter the human body including the skin, gut or human airways, but our primary focus will be studying the impact via human airway access. Previous studies have raised three main disadvantages when using nanoparticles [6].

1. Since nanoparticle possess a large surface area compared to other large particles, the amount of exposure increases dramatically.
2. Some nanoparticles, such as Carbon Nanotubes (CNT), are fiber shaped, which can cause similar, negative impacts as asbestos.
3. If carbon is used as a main component of the nanoparticle, it is expected to be biologically persistent, and it will accumulate in the human airway system.

A nanoparticle that gets deposited in the human airways may be missed by the cilia and will travel further into the terminal bronchioles, compared to when deposited in the proximal airways. Since nanoparticles are very small and have a greater surface area per unit mass, the intrinsic toxicity is increased, and, if the lung clears this nanoparticle slowly, it also increases the exposure time. This could be very damaging to the lungs, as demonstrated by the example of the lung related disease Mesothelioma, cause by prolonged exposure to asbestos. What happens to the nanoparticles that are inhaled depends on the section of the human airways and the characteristics of the nanoparticle [6].

Nanoparticle Transport / Deposition

Nanoparticles, which have a diameter of ≤ 100 nm, will pass into the airways very fast, as has been shown in previous studies [7, 10-14]. These studies used hamsters and aerosol inhalation in humans, where particles were injected into the trachea rather than going thru the natural process of breathing. Nanoparticles in the range from 40 nm to 400 nm are transferred into the blood stream very fast, as demonstrated by a previous study, which was conducted on rats, that showed nanoparticles in the alveolar walls and in pulmonary lymph nodes [7].

The human respiratory tract also functions as a filter that removes particles from inhaled air, and the efficiency of this filter depends on the properties of the particles such as the size, shape density, charge, respiratory tract morphology and also the breathing pattern [6, 7]. All these parameters will impact the regions of the human airway where the particle will get deposited and also the amount of particle deposited. When the branching happens in human airways, the airflow rate decreases very fast, the time any particle spends in the human airways increases and diffusion starts to dominant in the transport of the particle.

There are three types of particle deposition: diffusion, sedimentation and inertial impaction, as shown in figure 3. Particle deposition by inertial impaction is dominant in the extrathoracic and trancheobronchial tree because this region of the airways has very high velocities and directional changes happen rapidly [7, 10-14]. Particles with diameter greater than $10 \mu\text{m}$ have a higher probability of being deposited in the extrathoracic region, where as particles with diameters ranging from 2 to $10 \mu\text{m}$ get deposited in the

tracheobronchial region. Particles deposited by the sedimentation and diffusion mechanisms have a longer residence time. Deposition in the smaller airways and alveoli occurs thru sedimentation; these particles have diameters ranging from 0.5 to 2 μm . Particles with diameters of less than 0.5 μm will be deposited primarily by the diffusion mechanism, which has the highest residence time compared to other transport mechanisms in human airways. Previous studies have shown that the correlation between particle size and the amount of particle deposited to be similar among species [7, 10-15].

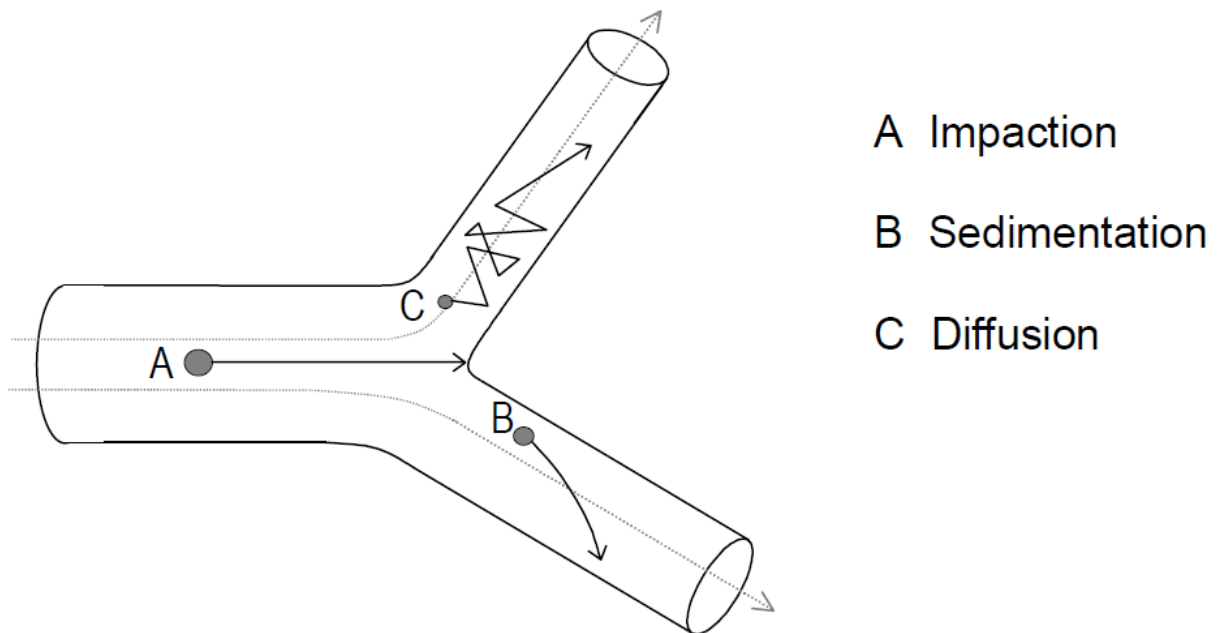


Figure 3. Different mechanisms of particle deposition in human airways (used without permission) [7].

Scope of The Thesis

The primary focus of the thesis is to simulate the behavior, transport and deposition of nanoparticles in human airways. This thesis will focus on nanoparticle inhalation and deposition in human airways. The model equations are solved using the Finite Element Method (FEM), and the governing equations that are used in the model are the Navier-Stokes equations with one or more advection-diffusion equations [6, 9-12, 15, 16]. All governing equations are solved at the same time, or monolithically, and solving the model problem monolithically raises several issues when the matrix is assembled. The matrix problem is solved using the Trilinos library, which is developed and distributed by Sandia National Lab, and the details of the solver are discussed in Chapter 3.

FEM is not stable when modeling nanoparticles due to the small diffusivity relative to the velocity (i.e., the small Peclet number), and modeling these systems requires one of several different methods of stabilization such as Stream-Upwinding (SU), Stream-Upwind Petrov-Galerkin (SUPG) and Galerkin Least Square (GLS) [8, 17, 18]. Another method regularly used to model advection dominated nanoparticle transport is the Discontinuous Galerkin Method (DG), but, while this method is stable for nanoparticle transport modeling, it is very difficult to implement in 3D and very computationally expensive [19]. A 3D FEM model is created here to study nanoparticles in human airways; this model is solved in parallel using the library package METIS. The programming language used for the 3D FEM model is C++, and all packages and libraries required for this model are available for free.

To compare the various stabilization methods, nanoparticle behavior is compared to an analytical solution, the Graetz solution, and the error is computed for flow through a simple cylinder [20]. The nanoparticle models are also compared for a human airway geometry with multiple bifurcations. The results presented will show the amount of particle being deposited for a steady state problem with different particle sizes.

Commercially available software such as ANSYS or COMSOL can model nanoparticle behavior in human airways, but there are many advantages for creating the model from scratch in C++. One obvious reason is the cost because, when writing the model in C++, the user can leverage packages and libraries that are available for free through various universities and organizations. Also, commercial software generally doesn't allow the user to vary certain parameters in order to optimize the model [10-12]. For example this study will show the impact of error with respect to different stabilization methods for advection diffusion equation. In relation to this work, ANSYS is popular software used to model transport in the human airways and is preferred by many researchers, but the user generally does not have any idea of the numerical accuracy of their model.

The outline of this work is presented as follows. Chapter 2 will discuss previous work on modeling human airways and finally conclude with how the current work differs from other previously published work. In Chapter 3, development of the FEM formulation for incompressible flow, advection-diffusion, stabilization techniques and the implementation will be discussed. Next, Chapter 4 will present the simulation results and

discussion. Conclusion and future research will be presented in Chapter 5, and a summary of goals reached in this current work will also be presented.

CHAPTER 2

LITERATURE REVIEW

This chapter will discuss work previously published by other researchers regarding the simulation of nanoparticle in human airways, and the unique aspects of the work in this thesis. This chapter will begin by discussing the previous studies performed on the human airway geometry by Weibel [21]. Discussion on modeling methods for the fluid phase and nanoparticle phase in human airways, including methods such as Euler-Euler and Euler-Lagrange are discussed in the following section. Finally, the chapter will end with the validation of the current algorithm.

Airway Geometry

The human airways are a very complex organ, and creating a computer model of the airway geometry poses a number of challenges. Previous studies have presented various airway geometries with lots of variations and complexity. Zhang and Kleinstreuer presented a model of the human airways based on Weibel's geometric data. Figure 4 shows the model that was used in their simulation. Typical human airway models are from generation 0-3 or -6, and any additional generations create too much computational complexity to be feasibly modeled [6, 9-12].

The Weibel model was the very first model that gave geometric information for the lung such as airway diameter and length, and the Weibel Type A is a very popular model that describes asymmetrically bifurcating airways [9-12]. Another method for

obtaining human airway geometry data is to use a CT scan or MRI, but these experiments are very difficult and expensive to conduct [9-12]. Many researchers build a rough representation of the human airways for meshing in 3D. A particularly interesting problem, which is not addressed in this thesis, is the change in human airways geometry and transport properties due to airways diseases.

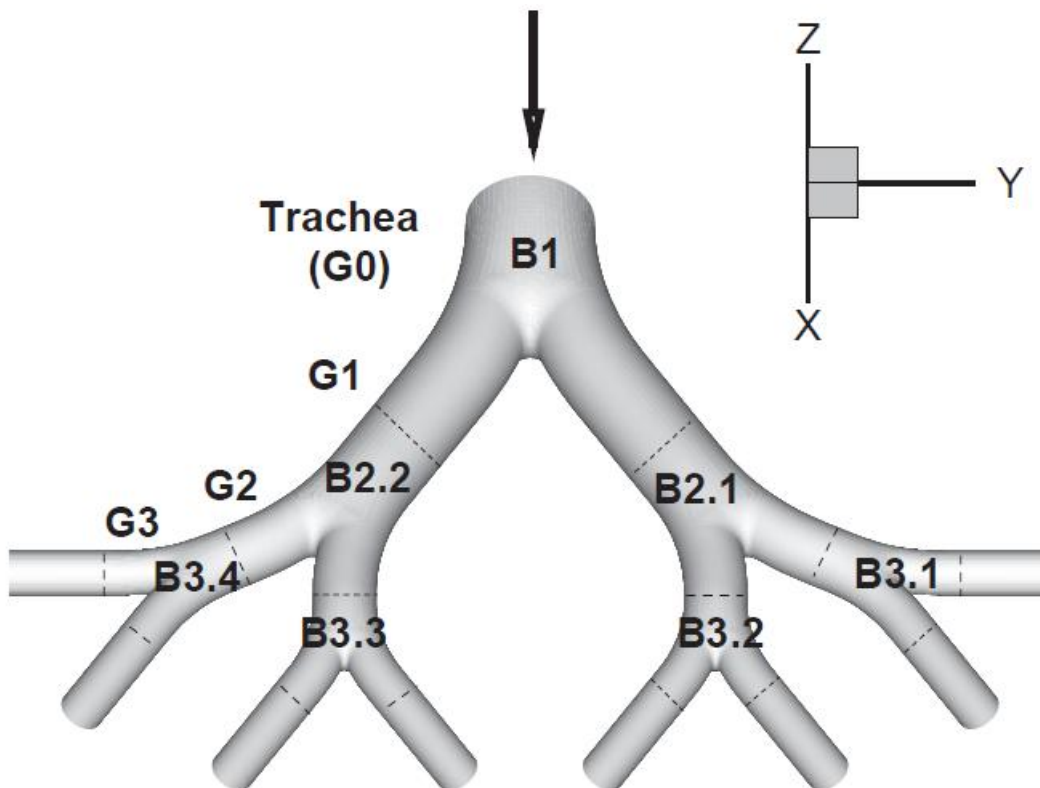


Figure 4. Human airways model for (Generation G0-G3), where B stands for the bifurcation (used without permission) [11].

In studies such as [12], where a human airway model with 16 generations (G0-G15) was created, the model was divided into 5 different stages of bifurcation. The dimensions of this model also follow the Weibel Type A geometry, assuming a lung volume of 3.5 L, and various changes made to the model, such as extending the outlet tubes in order to reduce any outlet and downstream effects.

Fluid Phase Models

Many studies have been done using Computational Fluid Dynamics (CFD) for fluid flow in human airways, especially where flows are laminar, due to the inherent complexity of modeling turbulent flow. Studies including turbulent flow typically used a low-Reynolds-number (LRN) $k-\omega$ model, which is very popular when studying human airways [6-12]. This model performed best at a Reynolds number of 300 to 10^4 , and the model does not require additional wall-layer functions since it captures both laminar and turbulent flow. This method is claimed to produce accurate results and is easy to implement [12].

Finding the velocity profile in human airways is a very difficult task due to experimental limitations [1, 8]. Knowing the velocity profile will help determine where the particles will get deposited, and this can also be related to the breathing pattern of a person. The distribution of velocity in bronchi has a great impact to the particle deposition [1, 8].

In order to find a specific site where the particle are being deposited, a detailed model of the lung is required, and this model must take into account the non-uniform

lobar expansion due to the pressure variation in the pleural cavity [1, 8]. The work of [1, 8] takes into account the uniform air expansion and contraction, but not the non-uniform lobar expansion due to its modeling complexity. Figure 5 shows a typical velocity flow field in a bifurcating human airway and cross sections at specific locations in the airways.

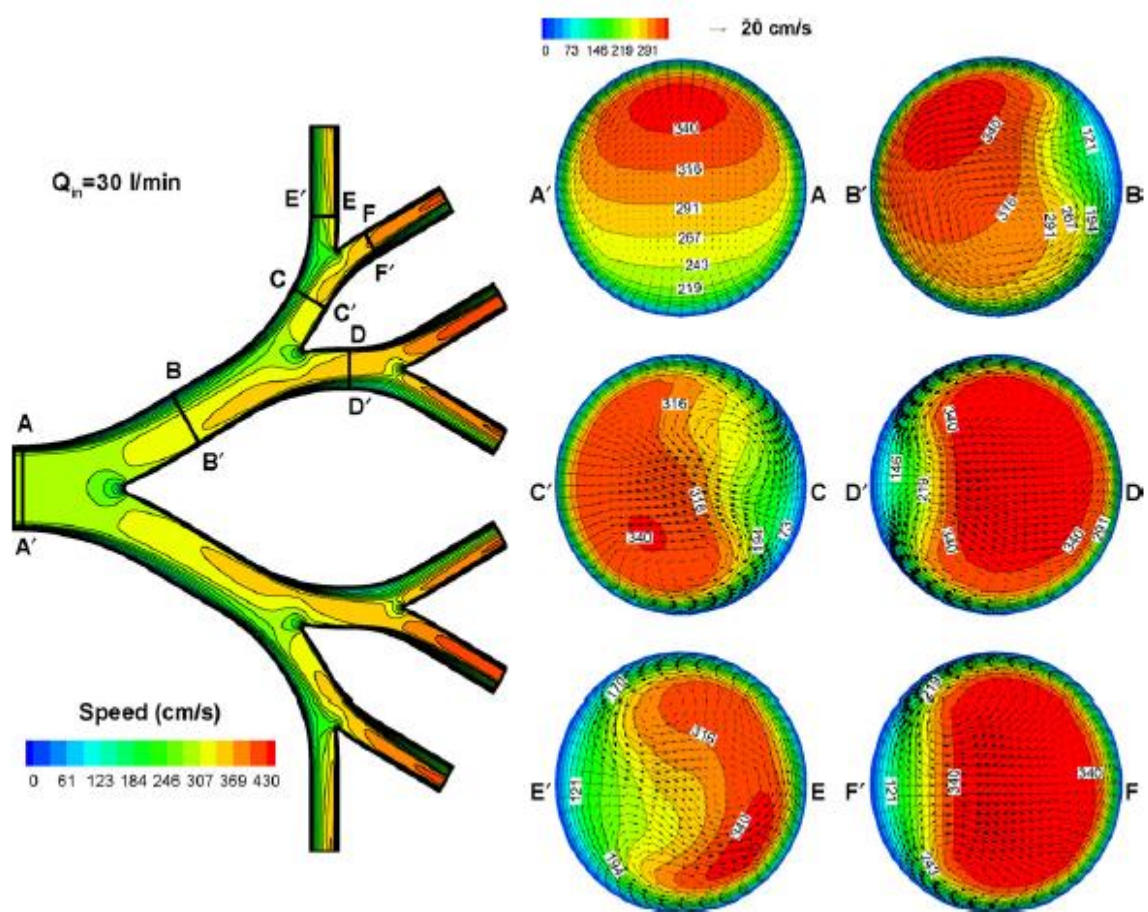


Figure 5. Velocity profiles in a human airway model and the different cross sections (used without permission) [10].

Nanoparticle Transport Models

Studies which compared both microparticle and nanoparticle transport in their model, have found that the inhaled particulate deposition depends greatly on the particle size [10]. Nanoparticles get dispersed due to diffusion and convection, and microparticle transport is governed by convection and sedimentation. The diffusion coefficient of nanoparticles is due to Brownian effects and can be calculated using the Stokes-Einstein relationship, defined as [10].

$$D_{nano} = \frac{k_B T C_{slip}}{3\pi\mu d_p} \quad (1)$$

where k_B is the Boltzmann constant, T is the temperature, μ is the fluid viscosity and C_{slip} is the Cunningham slip correction factor.

Rather than using Equation (1), many researchers have used the Euler-Lagrange approach when modeling nanoparticles. The convection-diffusion equation can be written as [10]:

$$\frac{\partial Y}{\partial t} + \frac{\partial(u_j Y)}{\partial x_j} = \frac{\partial}{\partial x_j} \left[\left(D_{nano} + \frac{\nu_T}{\sigma_Y} \right) \frac{\partial Y}{\partial x_j} \right] \quad (2)$$

where Y is the mass fraction of the concentration, σ_Y is the turbulent Schmidt number for Y and ν_T is the viscosity. The airway walls act as a sink, so the boundary condition at the wall is $Y_w = 0$; this boundary condition is a good assumption for the nearly instantaneous reaction kinetics between the wall and particulate [10]. Since the airway wall is set to $Y_w = 0$, it is straight forward to calculate the maximum amount of deposited particulate per area (DF) in the airways using Equation (3) [10].

$$DF_{region} = \sum_{i=1}^n \frac{-A_i \left(\bar{D} + \left(\frac{v_T}{\sigma_Y} \right) \right) \left(\frac{\partial Y}{\partial n} \right)_{|i}}{Q_{in} Y_{in}} \quad (3)$$

where A_i is the area of the lung wall, i is the local wall cell and n is the number of wall cells in the airways. Equation (4) is used to calculate the deposition enhancement factor (DEF), which shows the distribution of nanoparticles in the human airways [10]:

$$DEF = \frac{\left(\bar{D} + \left(\frac{v_T}{\sigma_Y} \right) \right) \left(\frac{\partial Y}{\partial n} \right)_{|i}}{\sum_{i=1}^n \left[A_i \left(\bar{D} + \left(\frac{v_T}{\sigma_Y} \right) \right) \left(\frac{\partial Y}{\partial n} \right)_{|i} \right] / \sum_{i=1}^n A_i} \quad (4)$$

There have been other studies that have modeled nanoparticle transport in human airways [1, 13, 22, 23], but these studies are not directly relevant to this work.

Nanoparticle Deposition Characteristic in Human Airways

Kleinstreuer et al. [9-12] studied nanoparticle distribution in terms of DEF (Distribution Enhancement Factor), and the flow rate was set to $Q_{in} = 30 \frac{L}{min}$. Figures 6 and 7 show the DEF for nanoparticle diameters from 1 nm to 100 nm for bifurcating airways (G0 to G3). When the nanoparticle diameters range from 1 nm to 10 nm, the deposition specifically occurs at the carinal ridges and the inner wall around this region [11]. This is due to the complex flow characteristics and very large concentration gradients in the carinal region [11]. When the nanoparticles are in the range of 100 nm, the highest DEF are in the third carinal region and uniform at the entrance due to the lower diffusion limit [11].

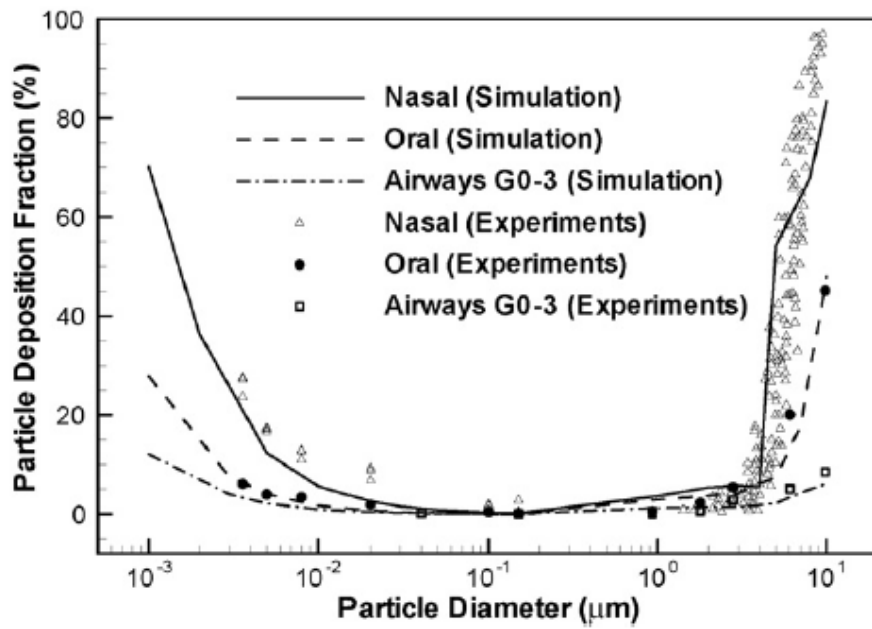


Figure 6. Validated model results of DF (%) as a function of particle diameter (used without permission) [10].

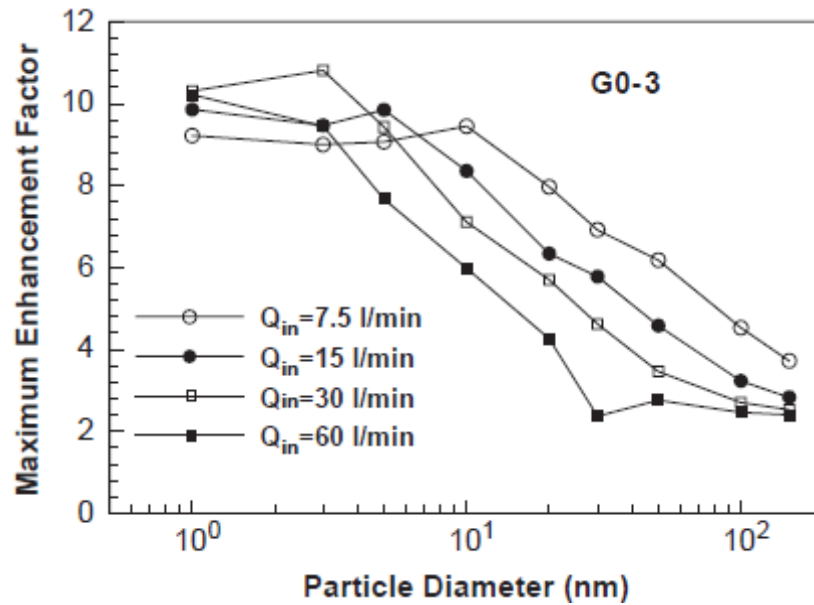


Figure 7. DEF of nanoparticles for varying flow rates for a human airway model (used without permission) [11].

Figure 8 shows the DEF values as a function of nanoparticle diameter and inlet flow rate [11]. The relationship shows that the maximum DEF will decrease when the particle size increases, while for nanoparticles from 10 nm to 30 nm, the DEF will decrease [11]. The deposition fraction (DF) for human airways (G0-G3) as function of inhalation flow rate and particle diameter is shown in Figure 8.

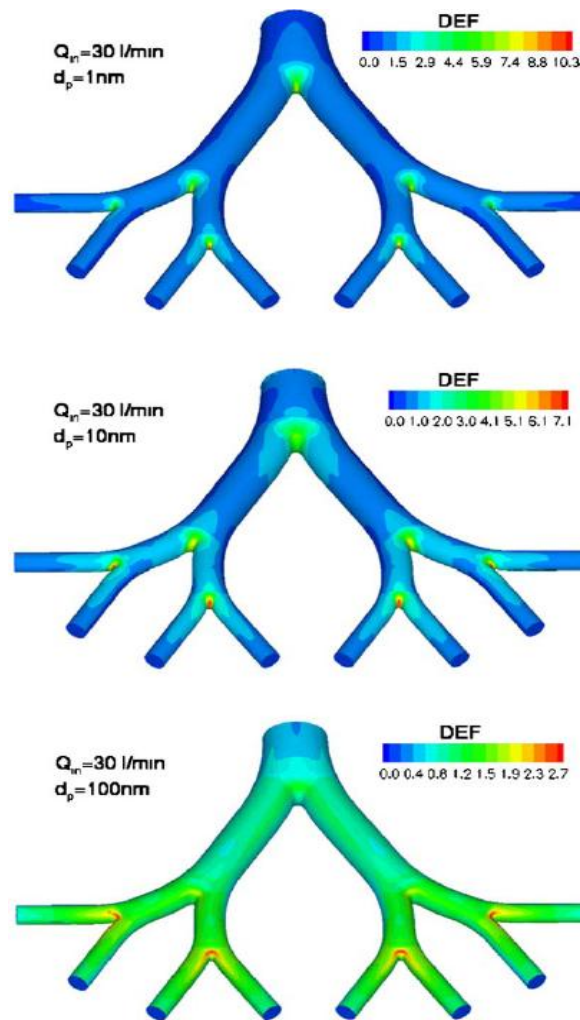


Figure 8. DEF of nanoparticle in human airways for a steady flow rate of $Q_{in} = 30 \frac{L}{min}$ (used without permission) [11].

The DF is calculated using Equation (3), and these results agree with many other studies [11]. When nanoparticles are inhaled, the DF will be lower since the diffusion limit is decreased, but when microparticles are inhaled the DF value increased because of deposition by impaction [11]. As shown in Figure 9, for nanoparticles there are minimal impacts due to the inlet flow rate compared to larger particle size. When modeling nanoparticles, geometric effects are not significant, and the deposition sites for microparticles and nanoparticles are very similar [10].

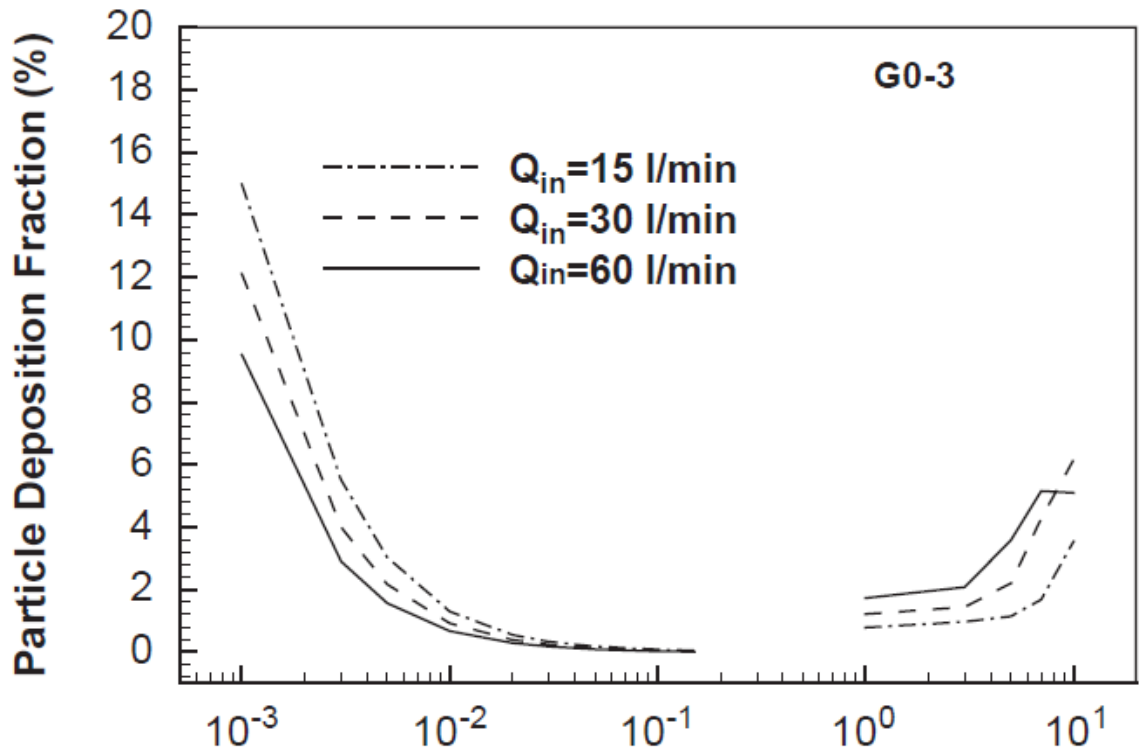


Figure 9. Varying DF for different flow rate and particle diameter in a human airways model (taken without permission) [11].

Model Validations

The model created by, [9-12] as described in the previous section was validated with data from previous studies, which included experimental data for steady and transient laminar flows in human airways. The LRN $k-\omega$ model used in, [9-12] has been extensively validated and has been shown to capture the physics of the problem at hand [11]. The models of nanoparticles in human airways due to diffusion have been compared to both analytical solutions in straight pipes and with experimental data of a double bifurcation airway by Zhang et al. [11]. This method of comparing the model solution with an analytical solution will be covered in the next chapter.

In conclusion, previous studies showed that the computer model agrees very well with the experiment observation [11]. Previous studies have shown that computer simulations are an excellent method for predicting laminar-to-turbulent flow including mass transfer and nanoparticle deposition in 3D bifurcating human airways [9-12].

Other studies, such as [1, 8], have focused on the upper airways using in-vivo and in-vitro deposition data in human airways, and this method has led to the construction of empirical relationships, such as that shown in Figure 10. Using this upper airway empirical data, other researchers have extended this information and applied it to modeling human airways [10]. The empirical data shown in Figure 10 was proposed by Jayaraju et al. 2009 and Stahlhofen et al. [1, 24-31], and this data was obtained using mono-dispersed particles that were tagged with a radiolabel, and the measured DF was combined with the inflow and outflow particle concentration [1, 8].

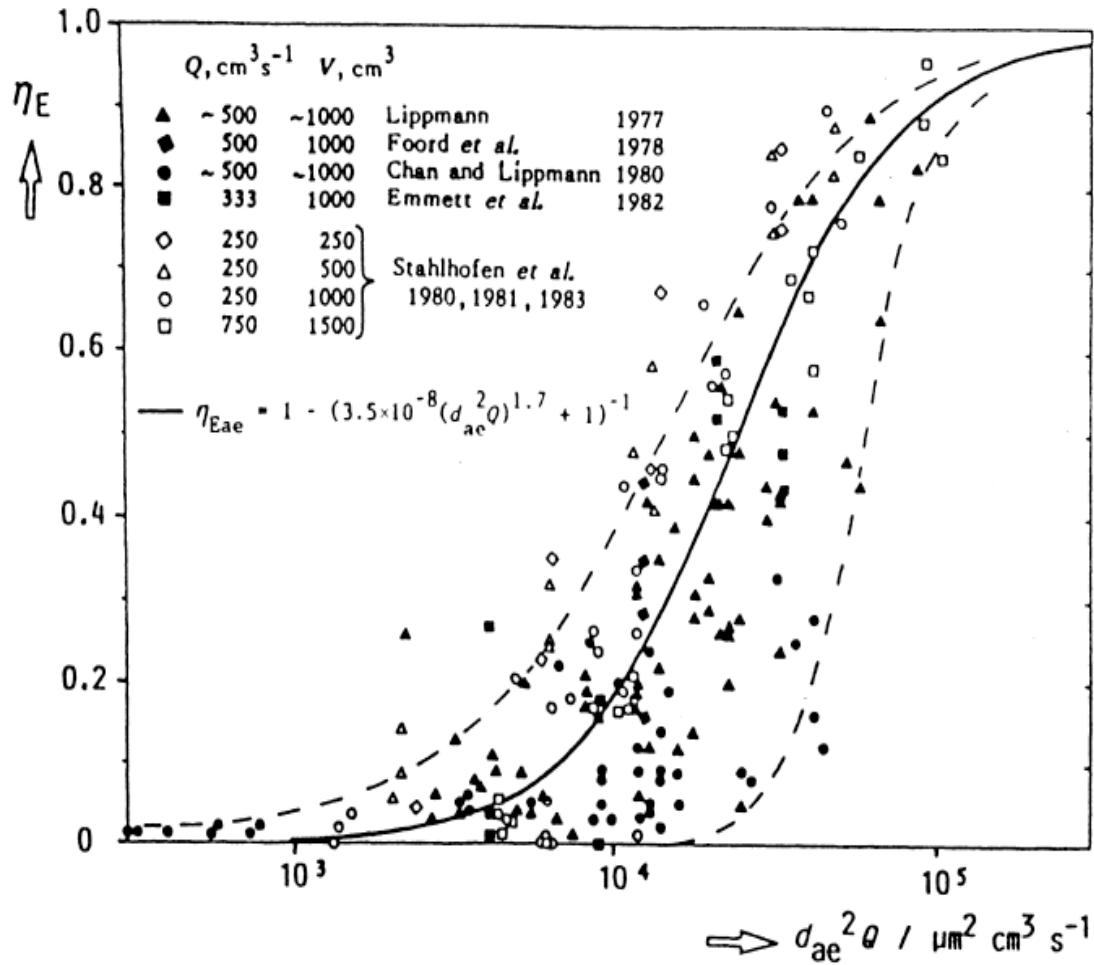


Figure 10. Empirical fit for the deposition efficiency in humans conducted by different studies (used without permission) [1].

CHAPTER 3

METHODS

Before creating a finite element model, the governing model equations are required. In this section, the derivation of Navier–Stokes equations from the continuity and the momentum equation will be shown, and the Advection-Diffusion equation will be included. Next, a brief introduction to the Finite Element Method and the discretization process for Navier-Stokes and Advection Diffusion equations will be presented.

Incompressible Navier-Stokes

Several phenomena in fluid mechanics can be described using the Navier-Stokes equations [32-35]. The equations are derived based on conservation of mass and momentum and can include external effects. The internally conserved quantities are affected by the pressure and viscous nature of the fluid. In this thesis, we will focus on the incompressible Navier-Stokes equations in a laminar regime. Since we assume incompressibility, the continuity equation simplifies as shown in equation (5).

$$\vec{\nabla} \cdot \vec{u} = 0 \quad (5)$$

If body forces are ignored the momentum equation is written as:

$$\rho \left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} \right) = \vec{\nabla} \cdot \sigma. \quad (6)$$

Since the focus of this thesis is looking at Newtonian fluids, σ can be expanded by equation (7), where p the hydrostatic pressure and τ is the total stress which is defined in equation (8).

$$\sigma = -pI + \tau \quad (7)$$

$$\tau = 2\eta D \quad (8)$$

In equation (8), η is the viscosity and D is the rate of strain tensor and is defined in equation (9).

$$D = \frac{1}{2}(\vec{\nabla}\vec{u} + (\vec{\nabla}\vec{u})^T) \quad (9)$$

The final form of the Navier-Stokes equation for an incompressible viscous fluid, neglecting body forces, is achieved by combining equations (6) and (9), and the result is shown in equation (10).

$$\rho \left(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \vec{\nabla} \vec{u} \right) = -\vec{\nabla} p + \vec{\nabla} \cdot 2\eta D \quad (10)$$

In the next section we will discuss how the advection diffusion equation is derived, followed by the process for transforming these two equations into discrete sets of equations for FEM.

Advection-Diffusion

The combination of advection and diffusion happens regularly in nature; in this section we will derive the advection diffusion equation. The principle of superposition states that the advection and diffusion terms may be added if and only if they are linearly independent [36]. Diffusion happens randomly due to molecular motion, and, in one dimension, each molecule moves either to the left or right (figure 11 shows the control volume schematic [36]).

Since we include advection, each of the molecules will also move $u\delta t$ in the cross flow direction as shown in figure 11 [36]. The total flux in the x-direction is J_x since the net movement of the molecule is $u\delta t \pm \delta x$ and the Fickian diffusion is shown in equation (11). Where C is the concentration and D is the diffusion coefficient.

$$J_x = uC + q_x = uC - D \frac{\partial C}{\partial x} \quad (11)$$

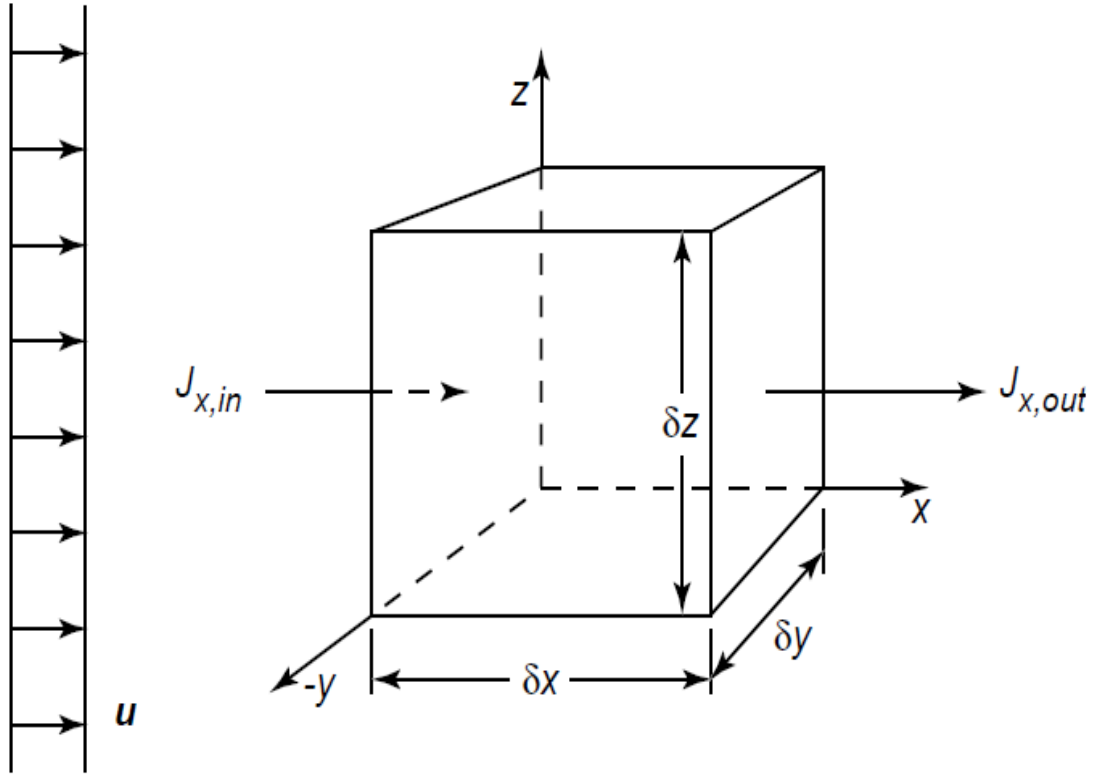


Figure 11. Control volume with cross flow (used without permission) [36].

Using the flux law and conservation of mass we can derive the full advection-diffusion equation using the control volume and the velocity $u = (u, v, w)$ [36]. The net flux from the conservation of mass is shown in equation (12) and (13).

$$\frac{\partial M}{\partial t} = \sum \dot{m}_{in} - \sum \dot{m}_{out} \quad (12)$$

$$\delta \dot{m}|_x = \left(uC - D \frac{\partial C}{\partial x} \right) |_1 \delta y \delta z - \left(uC - D \frac{\partial C}{\partial x} \right) |_2 \delta y \delta z \quad (13)$$

The final form of the advection diffusion equation is shown in equation (14).

$$\frac{\partial C}{\partial t} + \nabla \cdot (uC) = D\nabla^2 C \quad (14)$$

Finite Element Method (FEM)

The finite element method [32-35] is a very popular method for finding solutions to PDE's which are governed by boundary values [37]. Exact solutions for these problems are very hard to obtain when dealing with complex domains or nonlinear equations, but approximate solutions can be found using the FEM [37]. When the Navier-Stokes and advection-diffusion equations are solved monolithically (i.e., simultaneously) a solution vector, \tilde{x} is found that contains an approximation to the exact solution at a finite number of points, usually referred to as the interpolation points. Using basis functions to connect the approximate solution at the interpolation points, a continuous approximation of the solution can be constructed.

In order to derive a set of discrete equations, a standard process is normally used [37]. Using this process, the PDE is transformed into an integral equation using partial integration and weighted-residuals. The Galerkin approach is used to create a linear set of equations from the weak formulation [37]. The spatial discretization is created by dividing the domain Ω into non-overlapping subdomains, which are called elements Ω_e [37]. The collection of elements is often referred to as the mesh.

Discretization of the Navier-Stokes Equations

When solving the Navier-Stokes equations we need to consider both the spatial and temporal discretization. A temporal discretization is achieved by redefining the term $\frac{\partial \vec{u}}{\partial t}$ in equation (10); this is shown in equation (15).

$$\left(\frac{\partial \vec{u}}{\partial t}\right) = \frac{\vec{u} - \vec{u}^n}{\Delta t} \quad (15)$$

$\vec{u} = \vec{u}^{n+1}$ is the variable we want to determine, and \vec{u}^n is the known solution from the previous time step. With this redefinition we can re-write equation (10) using the implicit Euler method. A tiny detail that we have not yet discussed for the Navier-Stokes equation is the non-linear term, $\vec{u} \cdot \vec{\nabla} \vec{u}$, and, in order to avoid this nonlinearity, we must linearize the term using a Newton iteration, which is defined in equation (16).

$$\vec{u}_{i+1} = \vec{u}_i + \delta \vec{u}_i \quad (16)$$

\vec{u}_i in equation (16) is the approximate solution vector from the previous Newton iteration step, and \vec{u}_{i+1} is the approximate solution vector of the current iteration. $\delta \vec{u}_i$ is the change in the approximate solution that is found using equation (17). We are going to redefine $\vec{u} = \vec{u}_{i+1}$ in the rest of the equations in order to have a simplified form.

$$(\vec{u} \cdot \vec{\nabla} \vec{u}) = \delta \vec{u}_i \cdot \vec{\nabla} \vec{u} + \vec{u} \cdot \vec{\nabla} \delta \vec{u}_i - \delta \vec{u}_i \cdot \vec{\nabla} \delta \vec{u}_i \quad (17)$$

The higher-order term in equation (17) is assumed to be small and thrown out. Equation (17) is commonly known as the Newton-Raphson method [32-35, 37, 38].

Applying equation (17) to equation (10) create a new relationship as shown in equation (18).

$$\rho \left(\frac{\vec{u}}{\Delta t} + \delta \vec{u}_i \cdot \vec{\nabla} \vec{u} + \vec{u} \cdot \vec{\nabla} \delta \vec{u}_i \right) + \vec{\nabla} \delta p - \vec{\nabla} \cdot 2\eta D_u = \rho \left(\frac{\vec{u}_n}{\Delta t} \right) \quad (18)$$

The variable D_u is defined as $D_u = \frac{1}{2} \left(\vec{\nabla} \delta \vec{u}_i + (\vec{\nabla} \delta \vec{u}_i)^T \right)$. The unknown now is the change in the approximate solution, δu_i . The method of weighted residuals can be applied to equation (18) and (5), which produces the weak form and is shown in equation (19).

$$\int \vec{v} \cdot \left(\left(\rho \left(\frac{\vec{u}}{\Delta t} + \vec{u}_i \cdot \vec{\nabla} \vec{u} \right) + \vec{\nabla} p - \vec{\nabla} \cdot 2\eta D_u \right) \right) d\Omega = \int \vec{v} \cdot \rho \left(\frac{\vec{u}_i}{\Delta t} \right) d\Omega \quad (19)$$

$\vec{v}(\vec{x})$ in equation (19) is the weight function, which is continuous over the integration domain, Ω . The next step is to do the integration by parts on the terms with second-order derivatives, and the final equation in weak form is shown in equation (20) and (21), with Γ being the boundary of the domain Ω .

$$\int \vec{v} \cdot \left(\rho \left(\frac{\vec{u}}{\Delta t} + \vec{u}_i \cdot \vec{\nabla} \vec{u} + \vec{u} \cdot \vec{\nabla} \vec{u}_i \right) \right) d\Omega + \int D_v : 2\eta D_u d\Omega - \int p \vec{\nabla} \cdot \vec{v} d\Omega \quad (20)$$

$$= \int \vec{v} \cdot (\sigma \cdot \vec{n}) d\Gamma + \int \vec{v} \cdot \rho \left(\frac{\vec{u}_n}{\Delta t} + \vec{u}_i \cdot \vec{\nabla} \vec{u}_i \right) d\Omega \quad (21)$$

The continuity equation (5) must also go through some changes, which are shown in equation (22).

$$\int q \vec{v} \cdot \vec{u} d\Omega = 0 \quad (22)$$

Since we are splitting the domain into many subdomains using elements, the next step is to create interpolation points or nodes. The element type that is used in the thesis is discussed in detail later in this chapter. In order to approximate \vec{u}_h at any point \vec{x} we need the shape function φ_i , and equation (23) is the sum of values of \vec{u} for i nodes per element:

$$u_h(x)|_{\Omega_e} = \sum_{i=1}^n \varphi_i(x) u_i^e = \tilde{\varphi}^T \tilde{u}^e \quad (23)$$

where n is the number of nodes in an element, $\tilde{\varphi}$ is an array of shape function φ_i , and u_i^e are the unknowns, which are held by vector \tilde{u}^e [37]. The Galerkin method is applied when the shape functions are also used for weighting functions \vec{v} , and this is shown in equation (24) [37].

$$v_h(x)|_{\Omega_e} = \sum_{i=1}^n \varphi_i(x) v_i^e = \tilde{\varphi}^T \tilde{v}_e \quad (24)$$

After the substituting (24) into the weak form, the next step is to assemble all the different components, which yields the set of equations for the model problem as shown in equation (25). A much more detailed description of how the assembly is done in C++ is given in appendix A.

$$\frac{1}{\Delta t} \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \tilde{p} \end{pmatrix} + \begin{pmatrix} S + \frac{N(\tilde{u}_i)}{L} & L^T \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \tilde{u} \\ \tilde{p} \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \tilde{g} \end{pmatrix} \quad (25)$$

Discretize Advection-Diffusion

In this section we will discuss how the Advection-Diffusion equation will be discretized, and, since we need to stabilize the advection-diffusion equation, the stabilization portion will be discussed in a later section. In order to proceed with the FEM discretization, an approximation space must be declared first [18]. In the previous section we declared a spatial domain Ω which is divided with elements Ω_e .

In order to create the weak form, the advection-diffusion equation is multiplied by the weight function, and doing integration by parts to reduce the order of the second derivatives. It has been well established that the Galerkin method can be unstable for advection-dominated problem such as nano-particulate transport in the human airways [18]. In order to solve this problem we must add a stabilization term into the standard Galerkin form, and this will be shown in a later section of this chapter, and this technique is often referred to as upwinding [18]. The weak form is shown in equation (26).

$$\int (w^h a \cdot \nabla u^h + \nabla w^h \cdot \nu \nabla u^h) d\Omega = \int w^h f d\Omega \quad (26)$$

Further details of the weak form and implementation can be found in Appendix A. In the next section we will discuss the type of elements that will be used in this model to divide the mesh into Ω_e . Equation (27) shows how the discretized Navier-Stokes and Advection-Diffusion equations are assembled in matrix form. X and 0 in equation (27) are the non-zero entries and zero entries in the matrix and each X and 0 can be looked as

a block with a similar number of nodes. The diagonal entry for pressure is filled in with small number (10^{-8}) to help the iterative solver, but this does allow for a very small amount of compressibility. More detail about the assembly of the matrix can be found in Appendix A.

$$\begin{array}{l}
 N.S - x \\
 N.S - y \\
 N.S - z \\
 AD \\
 P
 \end{array}
 \begin{bmatrix}
 X & X & X & 0 & X \\
 X & X & X & 0 & X \\
 X & X & X & 0 & X \\
 X & X & X & X & 0 \\
 X & X & X & 0 & 10^{-8}
 \end{bmatrix}
 \begin{array}{l}
 U \\
 V \\
 W \\
 C \\
 P
 \end{array}
 \quad (27)$$

Elements

When modeling in 3D using the FEM, the two most popular elements are the hexahedron and the tetrahedron, and in this thesis we will primarily focus on using tetrahedron. Hexahedrons are preferred when the volumes have a regular structure, and tetrahedrons are preferred for irregular shapes or high aspect ratio shapes because they have the ability to mesh irregular volumes without creating badly shaped elements [32-35, 37, 38]. Since we are modeling human airways, using tetrahedrons as elements is the best option because of the complex geometry.

When solving the Navier-Stokes equation, we need to choose the type of pressure approximation to use, and there are two types of pressure approximation: continuous and discontinuous. The continuous approximation will use the vertex nodes of the element, in order to have continuity from element to element [32-35, 37, 38]. These elements are called the Taylor-Hood elements, and they will also ensure mass conservation, but the drawback is that the oscillations in velocity and pressure may be amplified [37]. The next

type of elements is the discontinuous pressure approximation element, which are called the Crouzeix-Raviart elements. The pressure nodes are placed at the center of the element, and this method unlinks the element-level pressure approximation with other neighboring elements and creates the discontinuity of pressure [37]. These elements have mass conservation at the element level but not at the global level like the Taylor-Hood elements [37].

The next focus is how the velocity and concentration should be handled, and this can be done either with a linear or quadratic approximation. It has been well established that quadratic approximations are better and also converge faster [37]. Using higher-order elements will allow curved element edges and surfaces, which leads to accurately meshing the volume with curved boundaries and lessening the number of elements required compared with the linear elements [37].

The distribution of degree of freedom (DOF) is very important once we have chosen the type of element and the approximation for pressure, velocity and concentration. The distribution of DOF must follow the Ladyshenskaya-Babuška-Brezzi (LBB) stability condition [37]. In figure 12 we can see the possible node locations for the tetrahedron. The 15 node tetrahedron is analogous to the 27 node hexahedron, but in this thesis a 10 node tetrahedron is chosen since it is much cheaper in terms of computational cost and easier to implement compared to the 15 node tetrahedron.

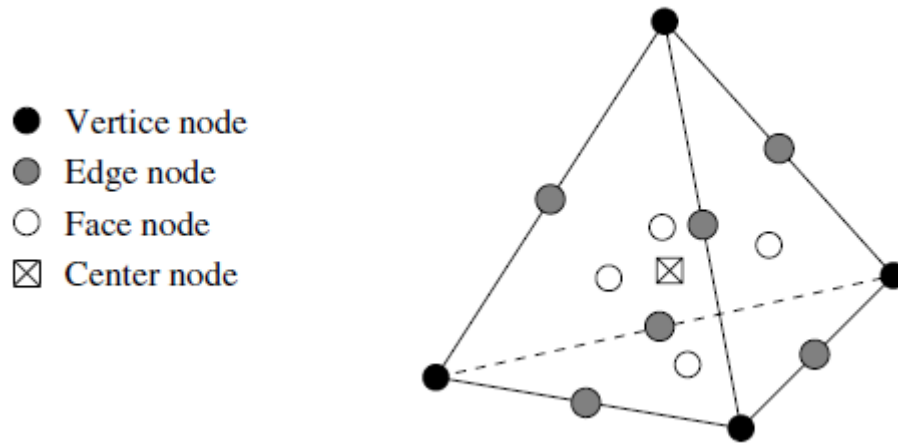


Figure 1. Example of a tetrahedron with 15 nodes (used without permission) [37].

The 10 node tetrahedrons have no nodes placed on the face and in the center of the element, and this element is called P_2P_1 element where the subscripts stands for quadratic (2) or linear (1) approximation [37]. The two P s refers to how the DOF are distributed, so for P_2 the velocity and concentration use a quadratic approximation while P_1 is the linear approximation for pressure. A single tetrahedron element with 10 nodes will have 44 DOF's per element: 10 nodes with velocity in 3 directions along with concentration and 4 pressure nodes on the vertices. In the next section we will discuss Baricentric coordinates and Isoparametric Coordinates.

Baricentric Coordinates and Isoparametric Coordinates

After we have established the type of elements that are going to be used, we need to create a local coordinate system, and, for this, employing Baricentric coordinates is very popular [37]. The coordinate system is based on $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ where λ_i is a linear combination of the global coordinates, x, y, and z [37].

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix} \quad (28)$$

In equation (28), x_i, y_i, z_i are the coordinates of the vertices, and solving equation (28) leads to equation (29).

$$\lambda_i = \pm \frac{a_i + b_i x + c_i y + d_i z}{6V}, i = 1, \dots, 4 \quad (29)$$

The sign \pm refers to $+$ when i is odd and $-$ for even, the constants a_i, b_i, c_i and d_i can be calculated using equation (30) and (31). Finally $6V$ in equation (29) can be calculated using equation (32).

$$a_1 = \det \begin{pmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{pmatrix}, b_1 = -\det \begin{pmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{pmatrix} \quad (30)$$

$$c_1 = -\det \begin{pmatrix} x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \\ x_4 & 1 & z_4 \end{pmatrix}, d_1 = -\det \begin{pmatrix} x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{pmatrix} \quad (31)$$

$$6V = \det \begin{pmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{pmatrix} \quad (32)$$

V is the volume of the tetrahedron, and, even though the Baricentric coordinates work well, one of the downsides is that the information about the other nodes will be lost when there is a mapping between Cartesian and Baricentric coordinate [37]. In order to overcome this problem we need to perform an isoparametric mapping, which is shown in

figure 13. This works by placing one of the vertices at the origin and the other nodes are placed on the coordinate axes at unity distance from the origin [37]. The volume of the tetrahedron is $\frac{1}{6}$, and derivatives are used when we need to transform the curved element into the unity/reference tetrahedron [37].

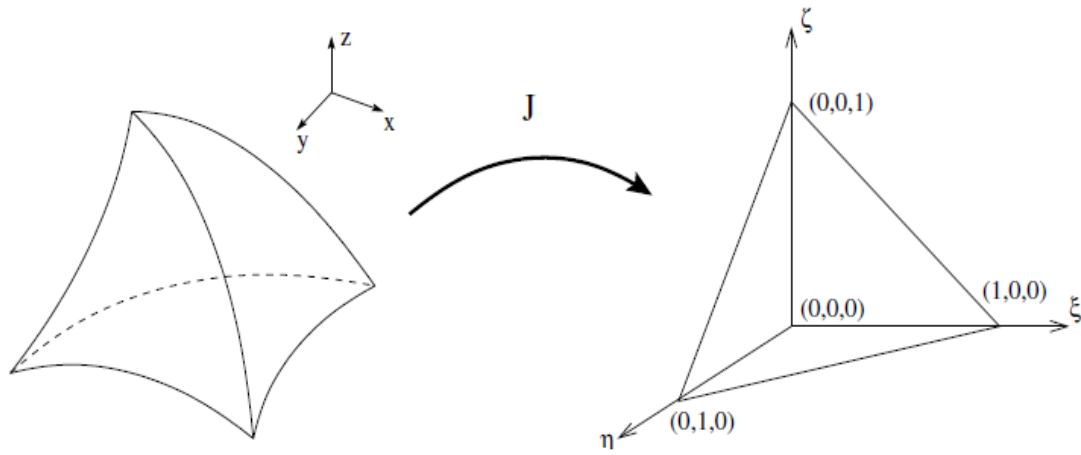


Figure 2. Isoparametric mapping of curved element onto a standard element (used without permission) [37].

$$J = \begin{pmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{pmatrix} \quad (33)$$

The transformation is performed using equation (33) and will be discussed in detail later in this chapter. Once the derivatives and the location of the nodes are

determined for the element, it is transformed back to the original shape using the inverse of equation (33).

Shape Function

In this section we will discuss the shape functions required for a 10 node tetrahedron, and there are several criteria the shape functions must satisfy. First, the quadratic approximation must be smooth on each element Ω_e [37]. Next, the behavior for all types of approximation functions (i.e., linear and quadratic) must be prescribed [37]. Finally, the polynomial expression must be complete. For example, the approximate field must be able to capture an arbitrary polynomial if the nodal values are defined correctly [37].

The DOF's that we are solving in this model, which are pressure, concentration, and velocity, must be chosen in order to satisfy the LBB condition. Any concentration basis can be selected, including the same basis as velocity or pressure. As established in the previous section the pressure approximation will be linear (ψ), while the velocity and concentration will be quadratic (φ), which satisfies the LBB condition.

Starting with the pressure shape function and using the linear interpolation technique, we can find the linear approximation p_h as shown in equation (34).

$$p_h = p_c + \frac{\partial p}{\partial x}(x - x_c) + \frac{\partial p}{\partial y}(y - y_c) + \frac{\partial p}{\partial z}(z - z_c) \quad (34)$$

Here, x_c, y_c and z_c are the location of the vertex nodes in Cartesian coordinates for the element. The definition of the linear shape functions are given in equation (35) and equation (36) will be the final form of the linear approximation of the pressure nodes.

$$\psi_1 = 1$$

$$\psi_2 = x - x_c$$

$$\psi_3 = y - y_c \quad (35)$$

$$\psi_4 = z - z_c$$

$$p_h = p_c \psi_1 + \frac{\partial p}{\partial x} \psi_2 + \frac{\partial p}{\partial y} \psi_3 + \frac{\partial p}{\partial z} \psi_4 \quad (36)$$

Next, the velocity and concentration shape function, φ , for a 10 node tetrahedron will be determined, as established in previous section. The 10 node tetrahedron will only have nodes on the vertex and edges, and this element must satisfy equation (37), which will be transformed into the Baricentric, coordinates later.

$$\text{if } \lambda_i = 1 \text{ then } \varphi_i = 1 \text{ (nodes on vertex } i) \quad (37)$$

$$\text{if } \lambda_i = \frac{1}{2} \text{ then } \varphi_i = 0 \text{ (nodes on edge near } i)$$

The C++ implementation of the shape function is shown in Appendix A, and in the next section we discuss the calculation of the derivatives of the shape functions at the integration points in order to build the matrices.

Shape Function Derivatives

The derivatives of the shape function, depends very much on the shape of each element because when meshing these elements the shape will be irregular according to the geometry of the mesh [37]. When mapping curved elements to the reference element, we need to use the isoparametric coordinates and the derivatives are expressed in equation (38), where $k=1, 2, 3$ [37].

$$\frac{\partial \varphi_i}{\partial \xi_k} = \frac{\partial \varphi_i}{\partial \lambda_1} \frac{\partial \lambda_1}{\partial \xi_k} + \frac{\partial \varphi_i}{\partial \lambda_2} \frac{\partial \lambda_2}{\partial \xi_k} + \frac{\partial \varphi_i}{\partial \lambda_3} \frac{\partial \lambda_3}{\partial \xi_k} + \frac{\partial \varphi_i}{\partial \lambda_4} \frac{\partial \lambda_4}{\partial \xi_k} \quad (38)$$

Using the global coordinates $\vec{x}^p = [x^p, y^p, z^p]$ we can find the derivatives of $\frac{\partial x_j}{\partial \xi_k}$ as shown in equation (39). We have defined J , which is the Jacobian, previously and computing $\det(J)$ gives the volume of the element [37].

$$\frac{\partial \vec{x}^p}{\partial \xi_k} = \sum_{i=1}^{10} \frac{\partial \varphi_i}{\partial x \xi_k} \vec{x}_i \quad (39)$$

Integration Rule

An integration rule plays an important role in finding approximations, and a simple way of calculating an integral numerically is shown in equation (40) [37].

$$I = \int g(\lambda) dx = V \sum_{m=1}^{Nint} g(\lambda_m) w_m + \epsilon \quad (40)$$

In equation (40), λ_m is the location of the integration point m , w_m is the weight function, N_{int} is the total number of integration points used, and ϵ is the error from the quadrature points used [37]. The Gaussian quadrature rule is very popular because it delivers high accuracy compared to other rules [37]. In this thesis we used the 11-point rule, and the implementation of the rule is attached in Appendix A [39].

Previous studies have used other integration rules such as Newton-Cotes, and such integration rules can provide better accuracy [37]. The better accuracy in Newton-Cotes is due to segregating the nodes in 4 types (vertex, edge, face and center), and each type will have a different weight function w_i [37]. Since we use only a 10 node tetrahedron, Gauss quadrature is exact and easier to implement.

Temporal Discretization

In this section we will discuss the temporal discretization of both Navier-Stokes and Advection-Diffusion equations. In previous sections, we have discussed how to linearize the non-linear term in the Navier-Stokes equation using Newton's method. Expanding from there, we have looked at three different methods for temporal discretization, which are the Backward Euler method (BE), Crank-Nicolson (CN) and the second-order Backward-Difference Method (BDF2). Each method has its advantages and disadvantages, and the BE method is first-order accurate in time whereas CN and BDF2 are second-order accurate in time [32-35, 38, 40].

We opted for the second-order methods, CN and BDF 2, for computational efficiency. A preliminary study showed that CN damps out error for oscillatory problems

better than BDF 2 with different stabilization methods. The advection–diffusion results with different time stepping schemes and stabilization are shown in figure 14, and other studies that used BDF2 have shown it to perform much better [33]. The test was performed on a straight tube comparing multiple stabilized methods for the advection–diffusion equation; this result was compared with the Graetz solution and further discussion will be included in the next chapter [20, 41]. The L_2 error shown in figure 14 is an approximation of the actual L_2 , and this approximation is not accurate when calculated and compared with meshes with different numbers of elements. The steps used to calculate the approximate L_2 error are shown in Appendix A.

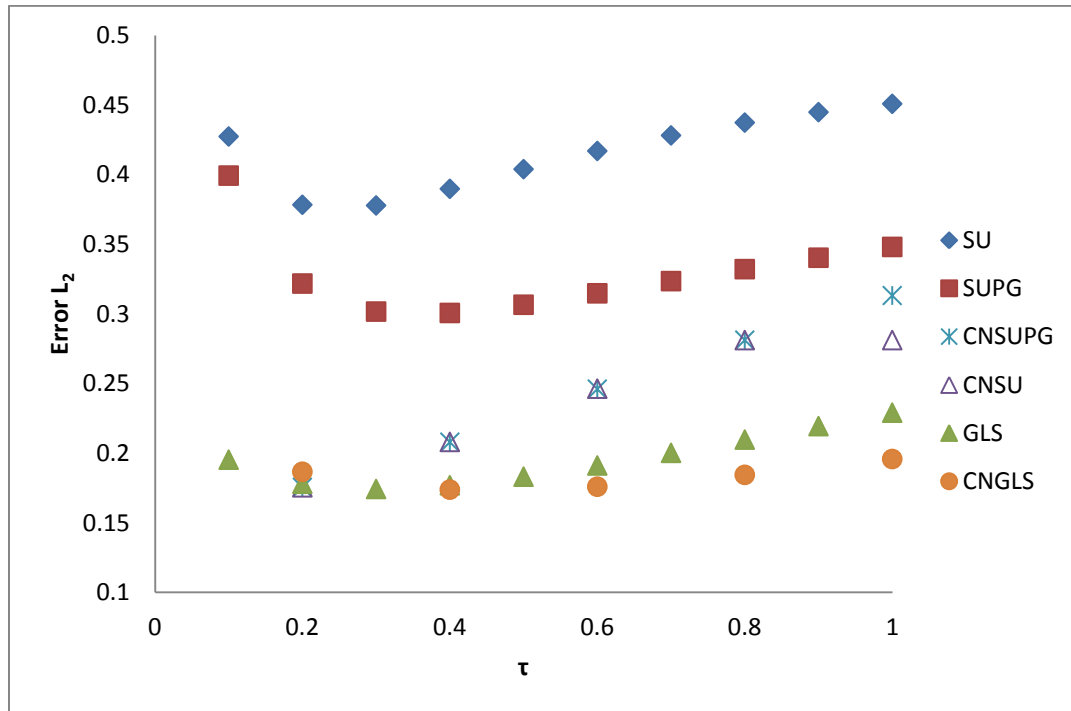


Figure 3. L_2 error for BDF2 and CN with varying τ parameter.

Equation (41) shows how the advection diffusion equation has been discretized temporally with BDF 2. BDF2 is not very stable compared to other methods, such as the implicit midpoint rule (IMR), but, because we are solving an advection dominated problem, BDF 2 will damp an oscillatory solution [33]. In equation (41), y_{n+1} (*new*) is the unknown, y_n (*old*) is the solution at the previous time step, y_{n-1} (*older*) is the solution from two time steps ago, and, finally, Δt is the time step size.

$$\frac{y_{n+1} - y_n}{\Delta t} = \frac{1}{3} \frac{y_n - y_{n-1}}{\Delta t} + \frac{2}{3} f(t, y_{n+1}) \quad (41)$$

Similarly, the CN method was also implemented based on equation (42). As shown in figure 14, the CN method does damp the error better than BDF 2 and CN also tends to slow GMRES convergence for difficult problems such as the current airway model. The final model was implemented with BDF2, because of its faster performance in solving the airway model.

$$\frac{y_{n+1} - y_n}{\Delta t} = \frac{1}{2} f(t, y_{n+1}) + \frac{1}{2} f(t, y_n) \quad (42)$$

Implementing BDF 2 for the Navier-Stokes equation becomes slightly difficult because of the inherent non-linear nature of the Navier-Stokes equation. Equation (43) shows how BDF 2 was applied to the Navier-Stokes equation, and f_{n+1} is the Navier-Stokes equation and GP_{n+1} is the pressure component. Further impacts of changing the temporal discretization will be discussed in the next chapter. Implementation of the temporal discretization in C++ can be found in Appendix A.

$$\frac{y_{n+1} - y_n}{\Delta t} = \frac{1}{3} \frac{y_n - y_{n-1}}{\Delta t} + \frac{2}{3} (f_{n+1} - GP_{n+1}) \quad (43)$$

The next section describes the variety of stabilization methods, such as Streamwise-Upwinding (SU), Galerkin Least Square (GLS) and Streamwise-Upwind Petrov Galerkin (SUPG), used to stabilize the advection-diffusion equation.

Stabilized Advection Diffusion

Different stabilization methods were tested for their accuracy and computational efficiency using the Graetz solution, which will be discussed in detail in the next chapter. First, SU stabilization is presented, and this stabilization method adds diffusion in the flow direction and not transversely using a diffusion operator in tensorial form [32].

The final form of the SU method can be looked at as the standard Galerkin method plus additional terms; this is shown in equation (44).

$$FEM + \int \tau (a \cdot \nabla w^h) (a \cdot \nabla u^h) d\Omega = 0 \quad (44)$$

Where τ is the intrinsic time, which is a constant between 0 and 1, w^h is weighting function, u^h is the unknown, and finally a is the component of the flow velocity [32]. The SU method utilizes the upwind test function only for the advection term, so there exists a potential problem regarding the accuracy of this method [32]. A more consistent stabilization method is needed, and, extending from the SU method, the SUPG method was created.

The SUPG method is very similar to SU, the only difference is an additional term in the weak form that makes the method consistent [17, 32]. Many studies have shown that the SUPG method performs better than SU, but the issue in using the SUPG method is the unknown τ parameter, as shown in equation (45), which introduces stability problems.

$$FEM + \int \tau (a \cdot \nabla w^h) (\sigma u^h + a \cdot \nabla u^h) d\Omega = 0 \quad (45)$$

In equation (45), σu^h is the unknown and can be described as $\frac{u_{new} - u_{old}}{dt}$ after discretization. In equation (45), we can also see that the nonlinear term $(\nabla \cdot (v \nabla u^h))$ from the advection-diffusion equation is not included, and this simplification is done when using linear elements [17, 32]. This simplification for quadratic elements is only an approximation.

The GLS method addresses the stability issues in SUPG; since it adds a symmetric stabilization term consistently [17, 32]. The GLS method works by element-to-element, weighted least-squares of the advection-diffusion equation, and this method provides symmetry, which is very important to achieve stability [32]. The GLS method can be formulated with SUPG and a Galerkin term weighted $1 + \sigma\tau$ [32].

If there exists any instabilities in the Galerkin method, GLS will tend to amplify them more than SUPG [32]. In Equation (46), we can see how the GLS method is being added to the advection-diffusion equation. Here, σw_h is the weighting function, which can also be described as $\frac{w_h}{dt}$ and dt is the time step-size.

$$FEM + \int \tau (\sigma w_h + a \cdot \nabla w^h)(\sigma u^h + a \cdot \nabla u^h) d\Omega = 0 \quad (46)$$

When implementing the different stabilization methods, several implementations were tried. First, each individual stabilization method was implemented, and then a combination of the different methods. When comparing this combined implementation for the Graetz problem, the L^2 -error is lower compared with the traditional implementation, but more study is required on this multiple stabilization method. Another idea was trying to use stabilization for a particular region of the mesh. For example, in a human airways mesh, SUPG could be used in the straighter regions and a different stabilization method, such as GLS, could be used near bifurcations. This concept is very promising and more study should be done on both of these ideas, but this thesis will not discuss these methods any further because they are not the primary focus.

Implementation

This section discusses the implementation of the model in C++, and the several additional packages that are required for the model to work. The main external package for this model is Trilinos, which is a compilation of several linear algebra packages created by Sandia National Lab. Trilinos is built with almost 50 packages, and the main packages that are required for this model are Epetra, AztecOO, and Ifpack. The Epetra package is used for the operator and vector construction routines, including functions for serial and parallel linear algebra packages [42-45]. The Epetra package builds the matrix and vectors for the $Ax = b$ model problem, and passes them to the AztecOO solver.

Trilinos also requires some basic Math packages such as BLAS and LAPACK, and additional packages can be built with Trilinos, but they are not necessary for this model problem.

There are several linear algebra solvers in Trilinos, and the most robust solver is AztecOO using the default ILUT preconditioner [42]. The model problem that we are solving is very difficult, and several validation tests were conducted to test the algorithm and determine the best linear solver for this problem. Solving the Navier-Stokes and advection diffusion equations monolithically is difficult because the operator is ill-conditioned, but a better preconditioner can be helpful and such a preconditioner exists in the Ifpack package [46].

Solving the model problem in 3D requires lots of computational power, and there is a need to parallelize the C++ code. The parallelization is achieved using two packages: OpenMPI, a well-known open source library, and Metis, developed by George Karypis at the University of Minnesota [47]. OpenMPI is a message passing interface, and Metis is a serial graph partitioning algorithm and fill-reducing matrix ordering [47]. Metis does nodal partitioning on a single core and then redistributes the partitioning to all the other cores to finally solve the problem [47].

Finally, in order to build the human airway mesh, Cubit, a meshing program developed by Sandia National Lab, is used. The Cubit program writes a Genesis file, which contains all the information about the mesh necessary to solve the model problem. This Genesis file requires other libraries, such as NetCDF and Exodus II, and these libraries are commonly available.

Additional information regarding the programs and libraries are listed below:-

- Trilinos:- [<http://trilinos.sandia.gov/>]
- Metis:- [<http://glaros.dtc.umn.edu/gkhome/metis/metis/overview>]
- OpenMPI:- [<http://www.open-mpi.org/>]
- Cubit:- [<http://cubit.sandia.gov/>]
- NetCDF:- [<http://www.unidata.ucar.edu/software/netcdf/>]
- Exodus II:- [<http://sourceforge.net/projects/exodusii/>]

Model Validation

In this section we will discuss the steps taken for validating the model; three test problems were used. The three tests are the backward facing step problem, flow around a cylinder and finally flow in a cylinder. The first test was conducted to find the best solver for solving Navier-Stokes and advection-diffusion monolithically. There are four different solvers tested for the backward step problem and the flow around a cylinder problem. The temporal discretization used for these tests was CN, since implementation was easier and the test problems are not as hard as the human airways problem. The solver uses preconditioning to solve the matrix problem $Ax = b$, and GMRES was used as the iteration solver.

Figure 15 shows the computational time for different solvers with different levels of mesh refinement for the backward step problem. As shown in figure 15 the best solvers are the Ifpack Amesos and the AztecOO, while Ifpack Amesos is a complete LU factorization and AztecOO in default Trilinos setting is ILUT (Incomplete LU

factorization with Threshold). Similarly, in figure 16, we can see the results for the flow around the cylinder problem and the best solver is the Ifpack ILU and Ifpack Amesos.

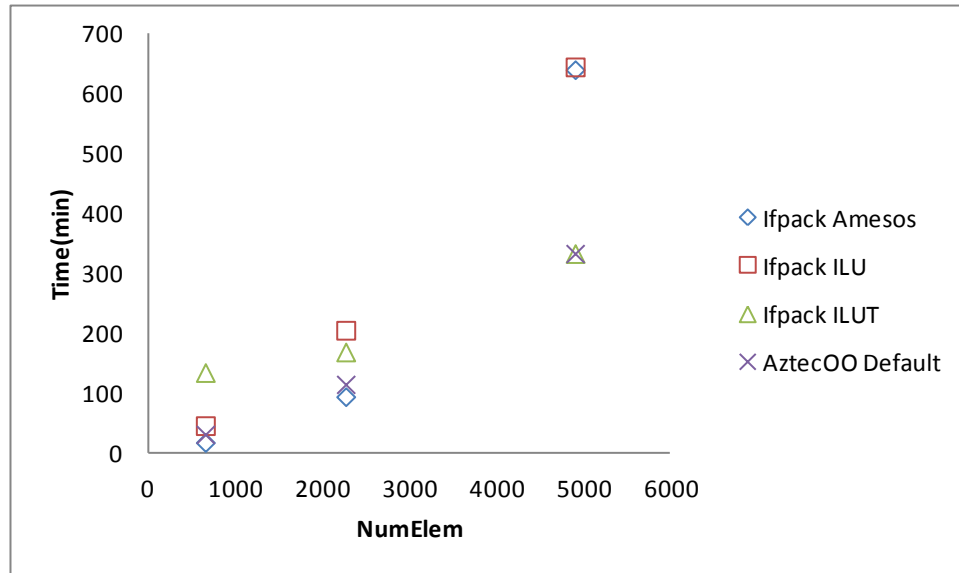


Figure 4. Computational time (min) for backward step problem with different solvers and varying number of elements (NumElem).

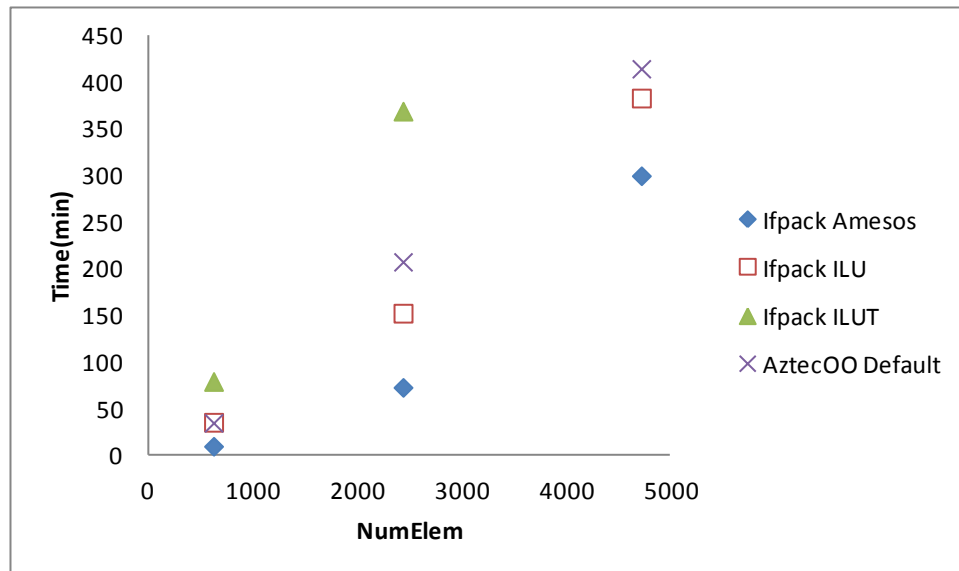


Figure 5. Computational time (min) for flow around a cylinder problem with different solvers and varying number of elements (NumElem).

Both results in figure 15 and 16 show that the model created works without any major error and the vector solutions for both problems are also shown in figures 17 and 18. In figure 17 we can see the backward step problem where a recirculation formed, and we can also see that at the wall the velocity is zero. In order to check the solution for the advection-diffusion problem, a test in a straight cylinder was conducted, and the result can be seen in figure 19.

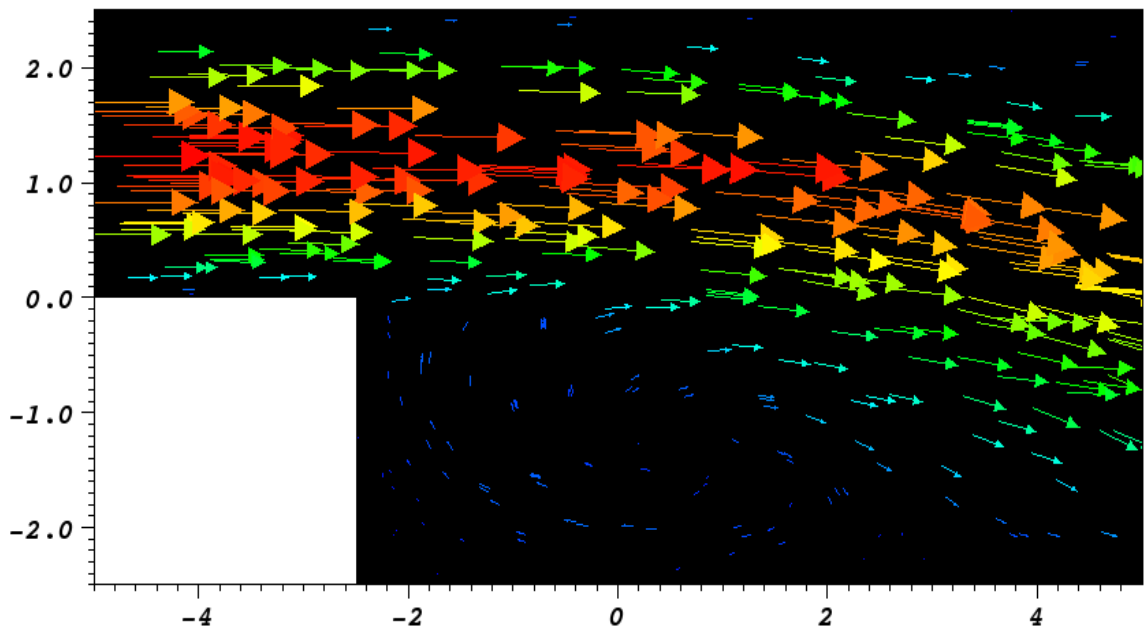


Figure 6. Vector solution for backward step problem with no flux boundary condition at the outlet when $t = 2$.

Figure 18 shows the result for the flow around a cylinder problem, and the simulation was run until $t = 2$ (dimensionless time). The inlet concentration enters the domain from only the top half as shown in figure 18. All the test problems have similar boundary conditions, where the inlet velocity and concentration are set to a parabola with

a peak of 1. The wall boundary conditions are set to zero and finally the outlet is set to a no flux boundary condition.

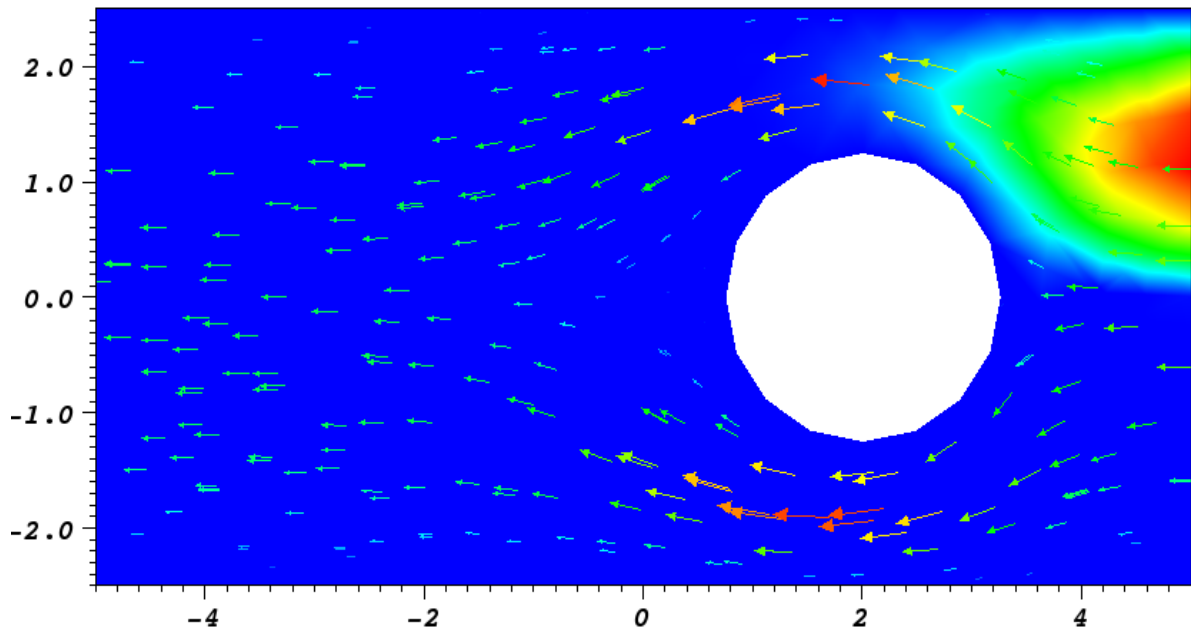


Figure 7. Vector solution and concentration for flow around a cylinder problem with no flux boundary condition at the outlet when $t = 2$.

The final test problem is the flow in a cylinder, which is shown in figure 19 where the later shows the velocity vector solution. The simulation was run until $t = 2$ (dimensionless time) and the diffusion coefficient $D = 0.01$. The solution was also compared with the Graetz solution, and more details about the comparison will be given in the next chapter. The next chapter will also discuss the results of the bifurcation geometry and stabilization results.

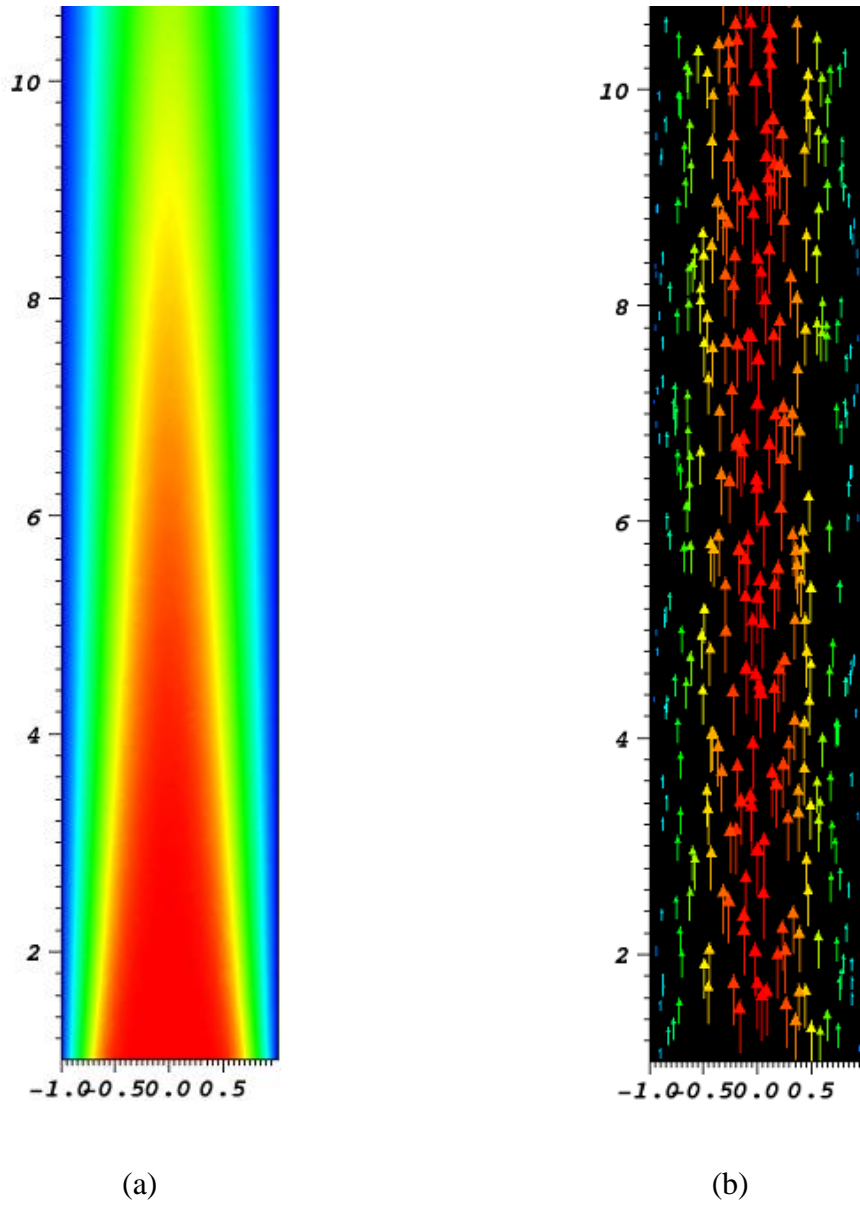


Figure 8. Concentration solution (a) and vector solution (b) for flow in a cylinder problem at the outlet when $t = 2$.

CHAPTER 4

RESULT AND DISSCUSION

Graetz Solution

In this section we will discuss the simulation results for the Graetz problem, which is an important benchmark problem in the study of advection-diffusion problems because it has an analytical solution. The Graetz solution can be used to compare different stabilization methods, and determine the error associated with each method in the L_2 -norm. The Graetz problem can also be used to determine the best τ value for a particular stabilization method.

The analytical solution to the Graetz problem was obtained from previous work, and the result was compared to the solution of the various approximate mathematical methods [41]. A cylinder mesh, as shown in figure 20, was constructed with around 4500 elements, the inlet boundary condition was set as the same as the Graetz solution, at the outlet a no flux boundary condition was applied, and finally the surrounding wall concentration was set to zero. The inlet velocity is set to a parabola with a peak of one, and, finally, the Reynolds number was set one. Since we are solving this model monolithically the Navier-Stokes solution in a cylinder will be close to the analytical solution, so a test of the validity of the Navier-Stokes equation is not necessary. Also, the mesh used for this comparison consists of approximately 4500 elements, and this provides a very good approximation in the finite element space, especially since the elements are third-order accurate. The cylinder is made with a radius (r), $r = 1$ and with

a height (h), $h = 10$. The height and radius are dimensionless variables used in this model.

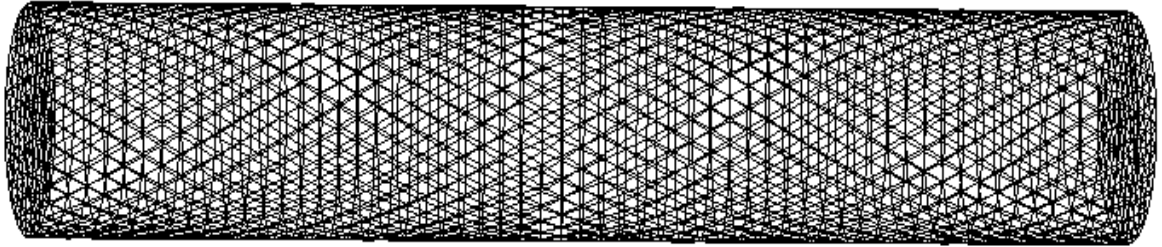


Figure 20. Meshed cylinder with 4532 elements, which was used for the Graetz solution comparison.

The comparison test was conducted with varying diffusion coefficients ($D=0.01$ to 0.000001), and also varying τ (0 to 1.0). Figure 21 shows the approximation error for different values of τ and different stabilization methods for $D = 0.00001$, and we can clearly see that the best method for stabilization of the advection-diffusion equation for this particular problem and diffusivity is the GLS method.

When $\tau = 0.3$, all the different stabilization methods show the smallest error. The second best stabilization method is SUPG, and, finally, SU is the least favorable method. Since previous studies have shown that the time step size Δt does have an impact on the τ coefficient, we will not study this relationship in much detail, and, for these simulations, we set $\Delta t = 0.01$.

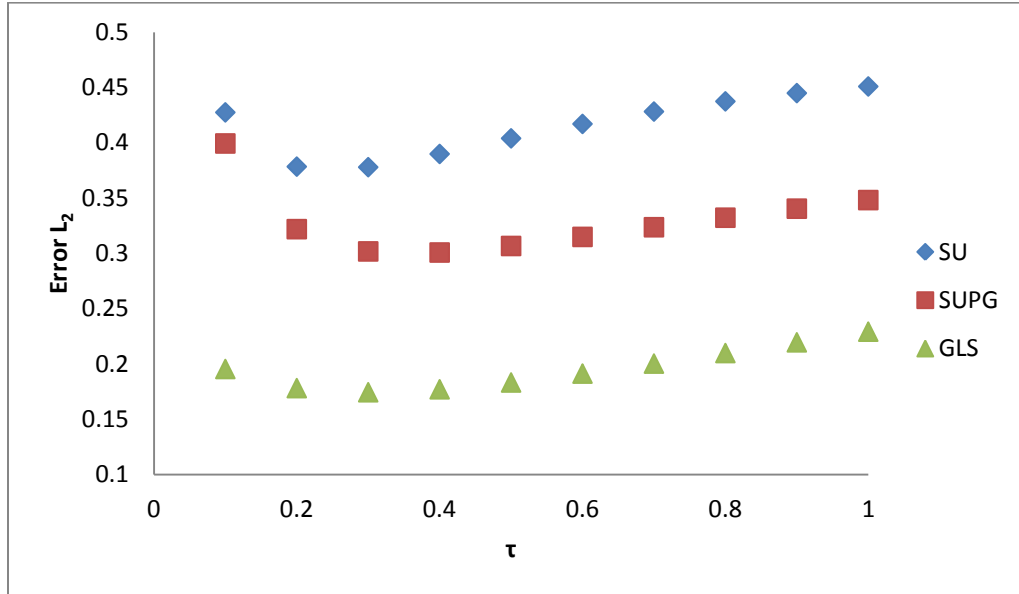


Figure 21. L_2 error for different stabilization methods using BDF-2 time stepping scheme.

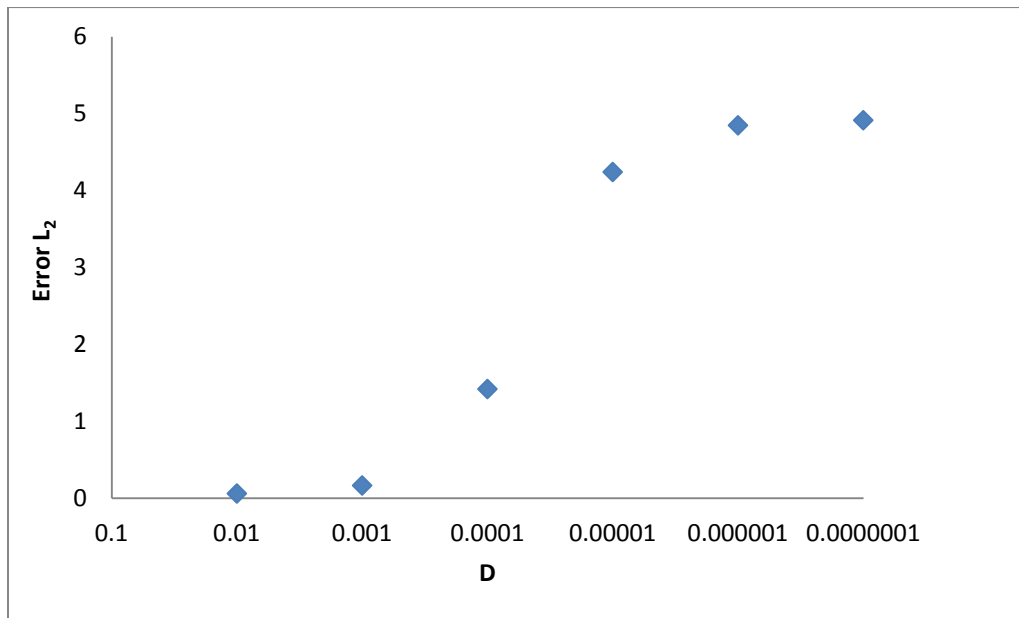


Figure 9. L_2 error for CFEM without any stabilization.

In figure 22 we can see the growth of the error when the regular FEM method is chosen without any stabilization implemented. This shows that stabilization methods are needed for small diffusion coefficients. Since we have the data for the stabilized FEM and regular FEM, we can compute the (% improvement) when utilizing a stabilization method. When $D = 0.01$, stabilization is not required since the regular FEM provides very good approximation, where as when $D = 0.001$ the regular, unstabilized method starts to include more error. Introducing stabilization will help to reduce this error, and this holds true for both SU and SUPG. Figure 23 show the improvement in percentage for both stabilization method SUPG and GLS compared to regular FEM.

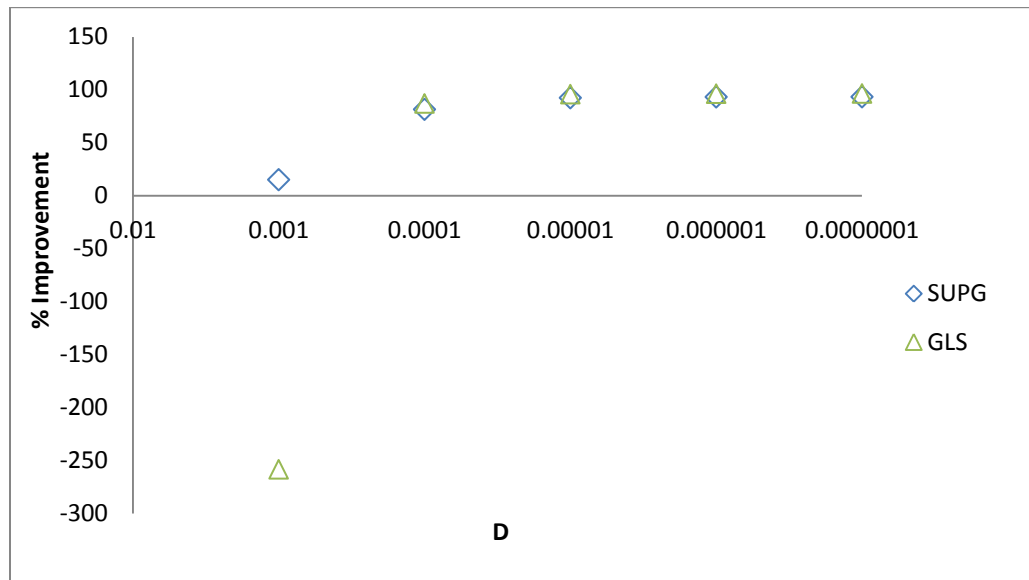


Figure 10. % Improvement of FEM when applying stabilization with varying diffusion coefficient.

As shown in figure 23, the regular FEM starts to become unstable when $D = 0.001$, and, using either SU or SUPG, we can solve this instability problem. When $D < 0.001$, it does not matter which stabilization method is used, since all the stabilization methods provides similar improvements. When using GLS stabilization at $D = 0.001$, the solution is worse since regular FEM error is small and when calculating the % improvement, this produces a large magnitude for % improvement for GLS. The GLS method is also known to amplify any instability in regular FEM when the problem is diffusion dominated. Another method of solving this instability is to refine the mesh, but this introduces other problems such as extremely long computational time and meshing software capabilities [15]. Other studies have shown that as the number of elements increase in a mesh, $\tau \rightarrow 0$ and this is also what we have observed in this study. This is because as the number of elements increase, the better the approximation we can obtain in the FEM space solution as long as the exact solution is also in that same space.

In figure 24, we can see the decay of L^∞ -error when the number of elements (NumElem) in the mesh is increased. If the number of elements is increased even further, we can see that the error will converge to a specific value. This result is unexpected since FEM theory states the error should converge to zero and is due to several factors. When comparing with the Graetz solution, we are ignoring the discontinuity at the inlet and also the SUPG stabilization method, which introduces certain amount of artificial diffusion and error. The test was conducted on similar mesh geometry as stated before, and the boundary conditions are also the same as before. Since we see very similar performance with SU and SUPG, further tests only include SUPG and GLS.

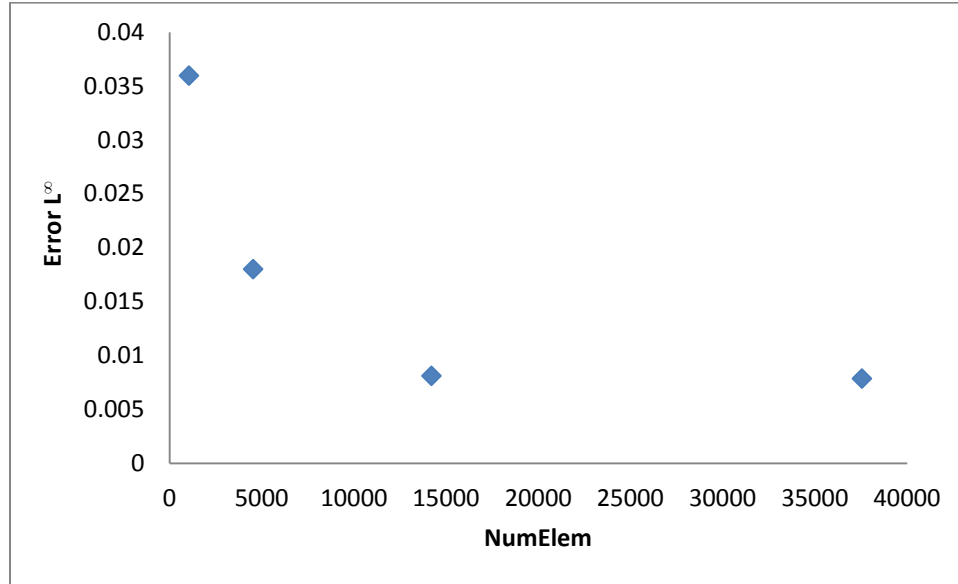
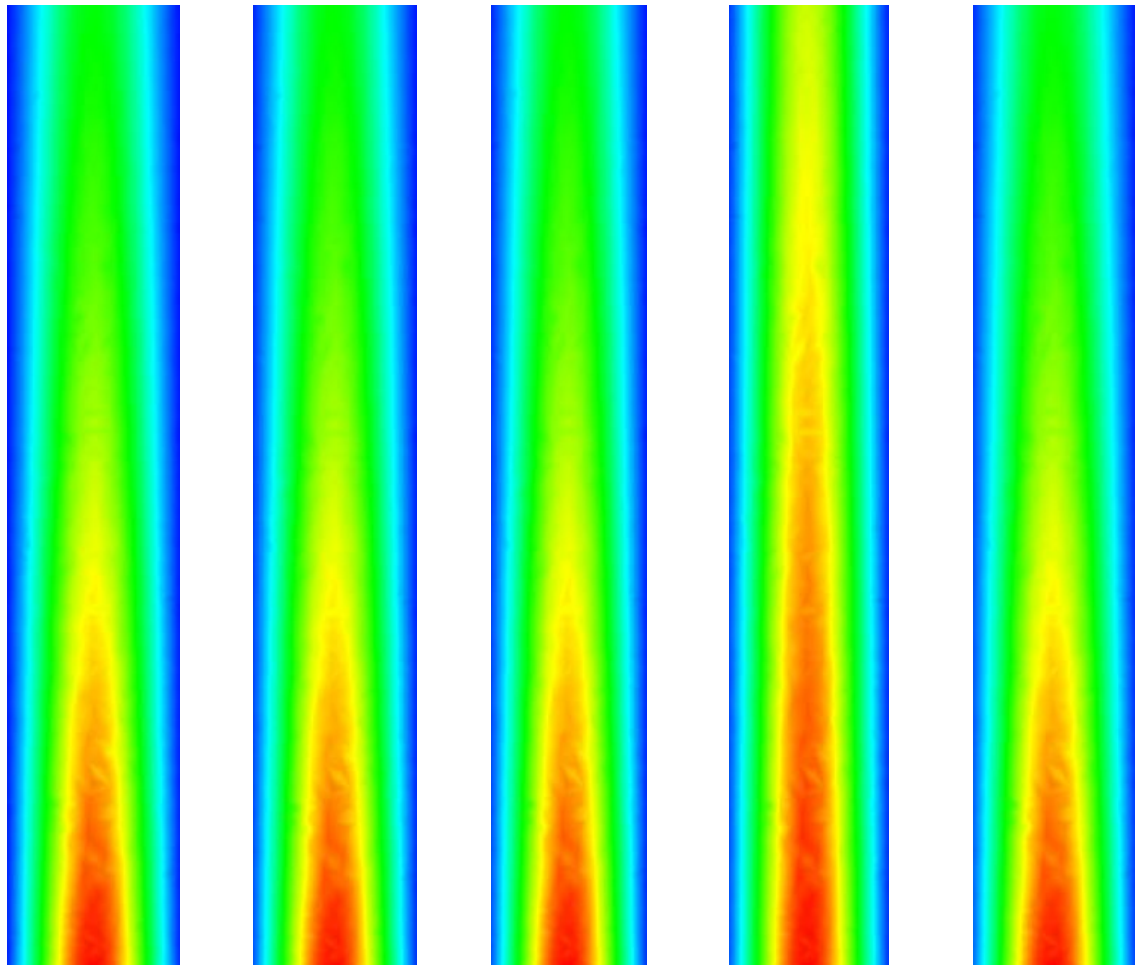


Figure 11. L^∞ error when varying number of elements (NumElem) in mesh, when $D = 0.001$.

Next, we will discuss the simulation results for multiple diffusion coefficients and different stabilization methods. Figure 25 shows a comparison of different stabilization methods, and we can see in figure 25 (d) where the concentration in the middle of the solution is higher compared to the Graetz solution in figure 25 (e). This is due to the artificial diffusion introduced by the GLS stabilization method when the regular FEM is stable, and we observed this trend when calculating the L_2 error. We can conclude that when the $D \leq 0.001$ the GLS stabilization method is not the best option for this particular problem.

Comparing the other methods in figure 25 (b) and (c) with the regular FEM solution (a) and the Graetz solution (e) shows a very similar trend, where most of the concentration sticks to the wall rather than exiting thru the outlet. This is due to the large

diffusion coefficient which causes a larger numbers of particles to stick to the wall and not exit the domain.

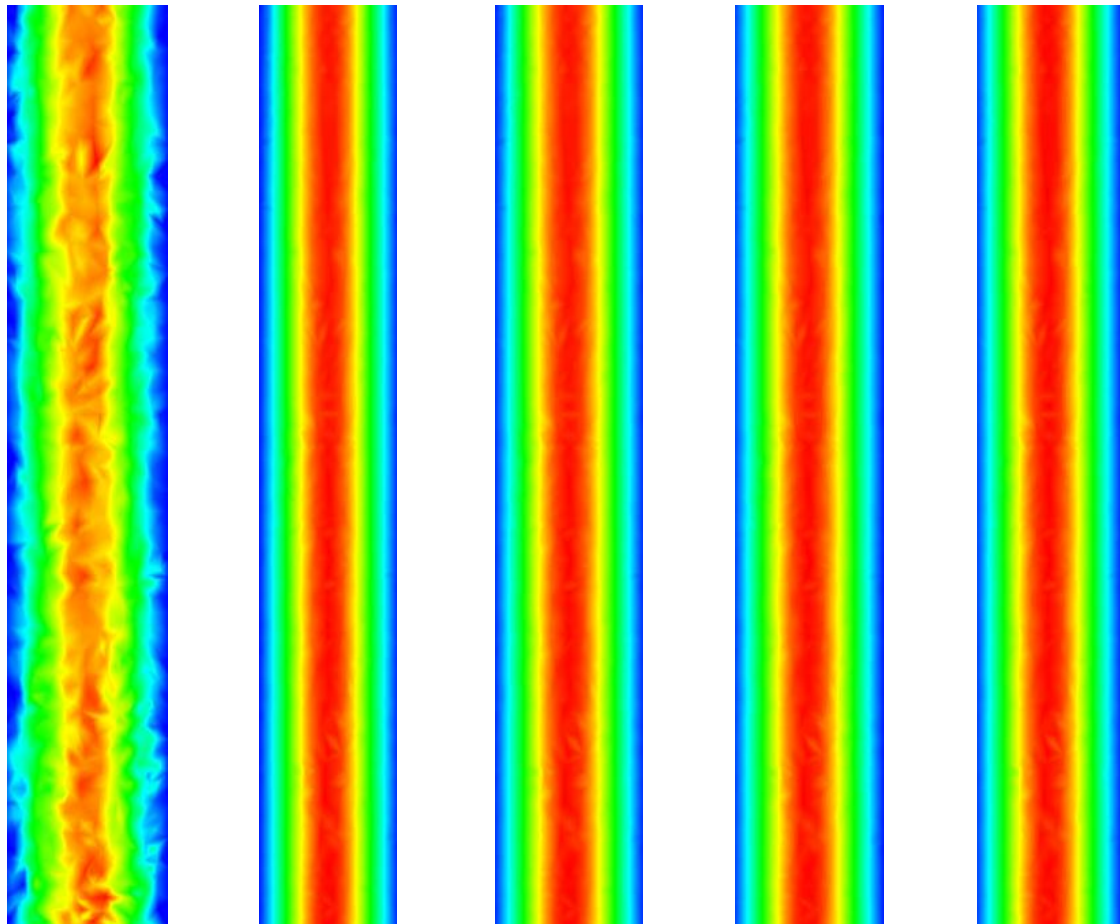


(a) (b) (c) (d) (e)

Figure 12. Simulation results for $D = 0.01$, where (a) is the regular FEM, (b) is SU-FEM, (c) is SUPG-FEM, (d) is GLS-FEM and finally (e) is the Graetz solution.

In figure 26 (a) we can see the regular FEM fails when $D = 0.000001$, and the stabilization methods work excellently, especially GLS. We can see clearly from the simulation results that when $D \leq 0.001$, the concentration at the inlet is largely

conserved through the domain to the outlet and very little diffuses to the wall. The Graetz solution agrees very closely with the GLS solution, which is shown in figure 26 (d).



(a) (b) (c) (d) (e)

Figure 13. Simulation results for $D = 0.000001$, where (a) is the regular FEM, (b) is SU-FEM, (c) is SUPG-FEM, (d) is GLS-FEM and finally (e) is the Graetz solution.

In the next section we will discuss the human airways simulation for nanoparticle deposition, and this will also include a flux calculation at both the inlet and outlet of the domain to give a sense of the accuracy of the method. We will also look at two stabilization methods, SUPG and GLS, since SU provides similar performance as SUPG.

Many of the choices made in this model are motivated by the observations made using the Graetz problem. The boundary condition at the wall the Graetz problem was set as a Dirichlet boundary condition and at the outlet a Neumann boundary condition was applied.

Human Airways Simulation

In this section, the results from the human airway geometry are discussed. The simulations were run with a constant inlet volumetric flow rate of $Q_0 = 15 \frac{V}{min}$, and the flow rate is consistent with that used by other studies [9-12, 14]. This flow rate impacts the Reynolds number (Re), so we need to calculate the Re based on the characteristic velocity, i.e., the inlet velocity (v_0), and the characteristic length (L). After recalculating these variables, we obtain a constant $Re = 200$, which was used in all the simulations. In this work we will not vary the volumetric flow rate and the Re , since a higher Re would require stabilizing the Navier-Stokes equation. Stabilization of the Navier-Stokes equation creates additional complexity since there would be two equations that require stabilization.

The diffusion coefficient for the nanoparticles was based on a previous study and rescaled to fit the current model [9]. Table 1 shows the particle size, the diffusion coefficient (\check{D}) and finally the dimensionless diffusion coefficient (D). The (D) can be calculated using (L) and (v_0), which will be used in the model. The dimensionless diffusion coefficient defined here is the inverse of the Peclet number ($\frac{1}{Pe} = \frac{\check{D}}{v_0}$).

Particle Diameter,d (nm)	Diffusion, \check{D} (m ² /s)	Dimensionless Diffusion,D
1.00E+00	5.34E-06	1.34E-03
5.00E+00	2.15E-07	5.40E-05
1.00E+01	5.44E-08	1.37E-05
5.00E+01	2.38E-09	5.98E-07
1.00E+02	6.76E-10	1.70E-07
1.50E+02	3.15E-10	7.92E-08

Table 1. Relationship of particle diameter and dimensionless diffusion.

Figure 27 shows the human airways mesh. The geometry was meshed with a 10 node tetrahedron and this step was done by the software Cubit. The mesh clearly shows that there is only one inlet and six outlets.

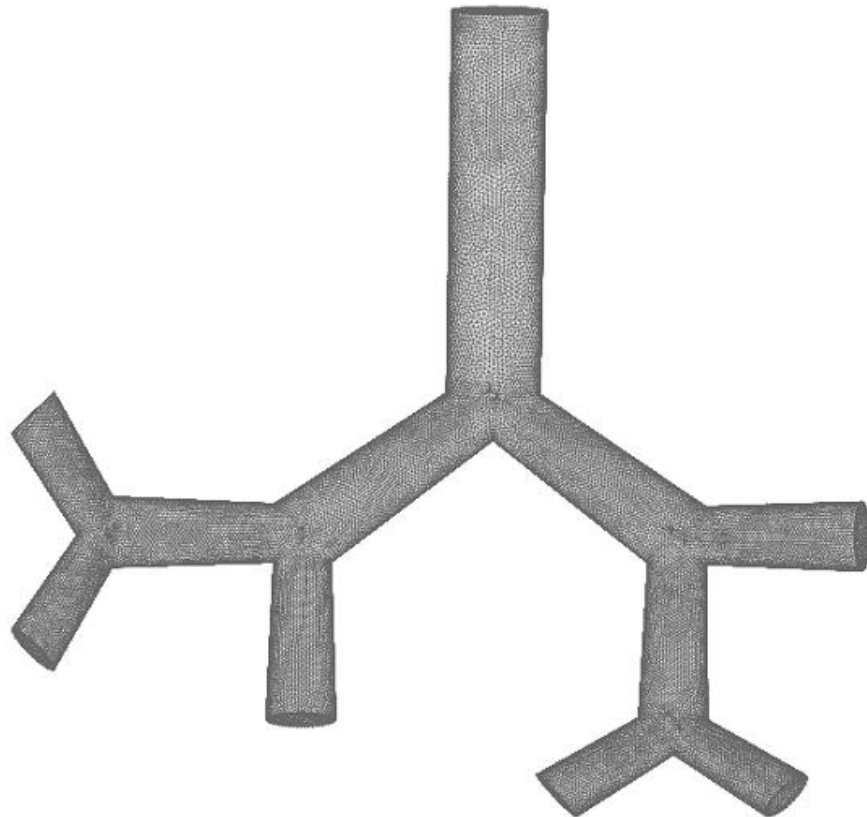


Figure 14. Human airways mesh from G0-G3 with approximately 100000 elements.

The inlet boundary condition for concentration was set as a parabola with a peak of 1, while the inlet velocity was set also to be a parabola with a peak of 2. The outlet boundary conditions are set as a Neumann boundary condition and finally the walls of the mesh were set to zero concentration and velocity. These boundary conditions are commonly used in other studies [2, 4, 6, 9, 11, 12, 14]. As shown in Table 1, the simulation was run for particle diameters ranging from 1 nm to 150 nm, and the simulations used SUPG and GLS stabilizations. The stabilization methods were compared by calculating the flux at both the inlet and outlet of the airways, and the flux calculation is shown in Appendix A. The flux is used to calculate the deposition fraction (DF), defined by equation 3 [9, 11, 12, 14].

Since modeling human airways is very computationally expensive, as stated in the previous chapter, we need to implement the model in parallel. For any computational model, the scalability becomes an important aspect of the algorithm development, and it can also show how efficiently the model has been implemented. In order to show the scalability, multiple human airways mesh with varying numbers of elements was created. The simulation was run to $t = 500$ (dimensionless time); this is to ensure that the solutions have reached steady state and all the simulations were run with 4 processors. Figure 28 shows the scalability plot, and we can see that for a range of problem sizes, there exist an approximately linear relationship between computational time and number of elements.

In order to check the DF calculation, a similar method as the scalability test was employed, and the DF values was calculated for varying number of elements. A base case

was used for all the simulation runs, where SUPG was used as the stabilization method and the diffusion was set to $D = 10^{-5}$. Once again, the simulation was run for $t = 500$ to ensure that the solution reaches steady state. Figure 29 shows the DF results for meshes with varying number of elements, and, from the result, we can conclude that as the mesh becomes finer the DF value converges to a non-zero value, as expected. This trend reflects the improvement in accuracy when more elements are used in the simulation.

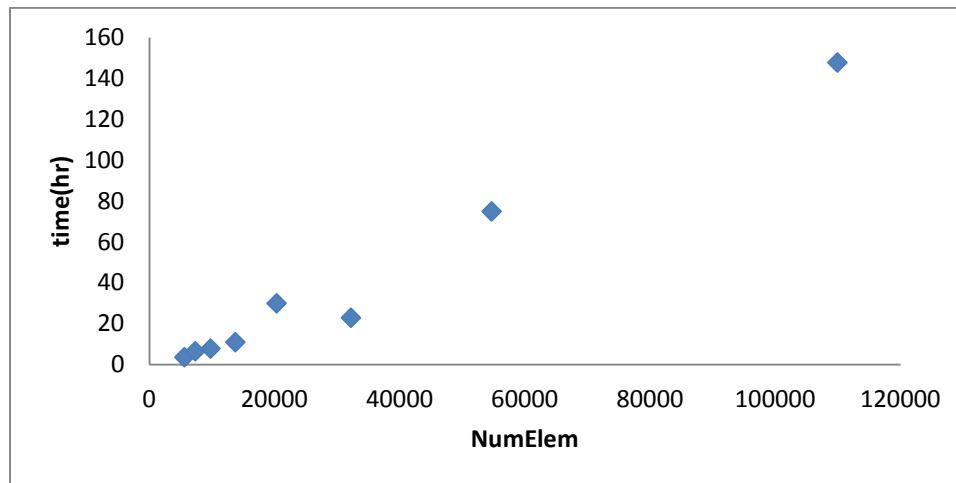


Figure 15. Computational time relationship with number of elements in a mesh.

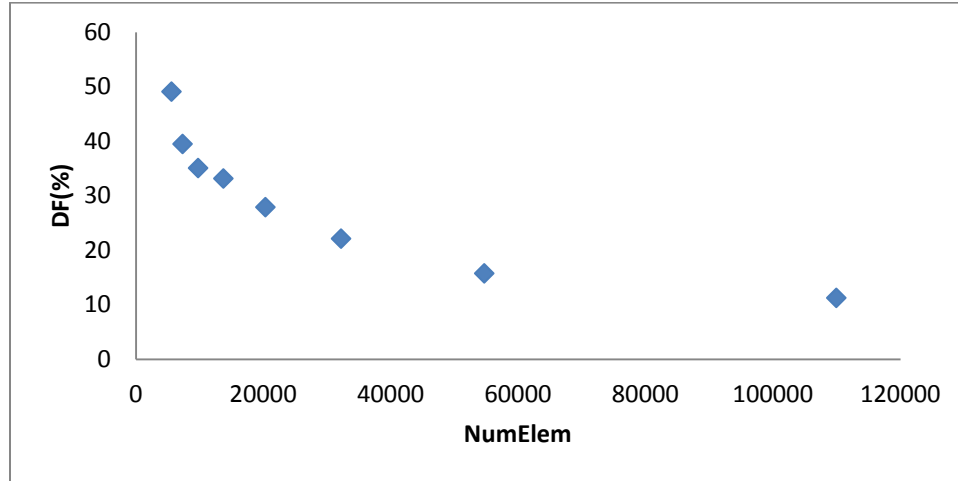


Figure 16. DF as function of number of elements for SUPG stabilization with $D = 10^{-5}$.

The next set of simulations was run for particles with different diameters, and these runs were conducted with two different mesh resolutions. The mesh resolutions that were used had approximately 30000 elements and 100000 elements, for comparison between a coarse and fine mesh. In figure 30 we can see changes in DF with respect to particle diameter for both SUPG and GLS stabilization, SUPG* and GLS* are simulations run using the finest mesh.

There are several important relationships to take from figure 31. First we can see that for particle diameters $d > 10 \text{ nm}$, the stabilization method that is chosen has minimal impact on the DF, and this holds true even when the mesh is refined. Second, when the particle $d < 10 \text{ nm}$, the stabilization method that is used for the model becomes very important. We now know that for the Graetz problem, the GLS stabilization method is much more accurate than its counterpart SUPG for small diffusivities values, and that same result is seen here. These relationships are simply

overlooked in many other studies, especially studies conducted using commercially available software such as ANSYS [9, 11, 12, 14].

This clearly shows that the stabilization method used to stabilize advection-diffusion equation does impact the solution, and another area that we did not cover in this thesis is comparing of different temporal discretization methods, such as 3rd-order and 4th-order. Higher-order temporal discretization is known to damp out error much better than the 2nd-order methods, which were used in these models [32, 33]. When using a higher order time stepping scheme, the τ parameter in the stabilization methods must be redefined since the higher order scheme will amplify the error for advection dominated problems.

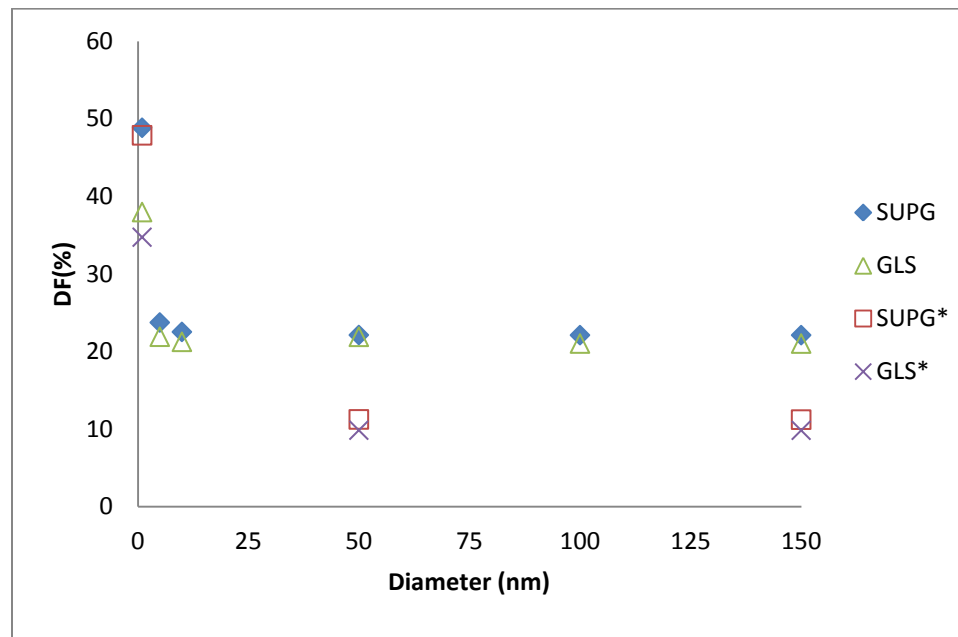


Figure 30. DF as function of particle diameter for SUPG and GLS stabilization, SUPG* and GLS* are simulation with 100000 elements.

We will briefly discuss the velocity solution, which is shown in figure 31, since we are running the simulation at a moderately high Re and there are some recirculation forming in the model. The simulation results are consistent with other studies [9-12, 14]. We can see from figure 31 that in the first bifurcation (G1), the maximum velocity is very close to the wall, and the generation to the right shows the highest airflow. We can also see in the third bifurcation (G3) the inertial forces due to the high Re at the inlet. Even though we set a parabolic boundary condition at the inlet, which is symmetric, the velocity splits in an asymmetric manner. This is due to the asymmetrical geometry of the human airway structure and the bifurcations.

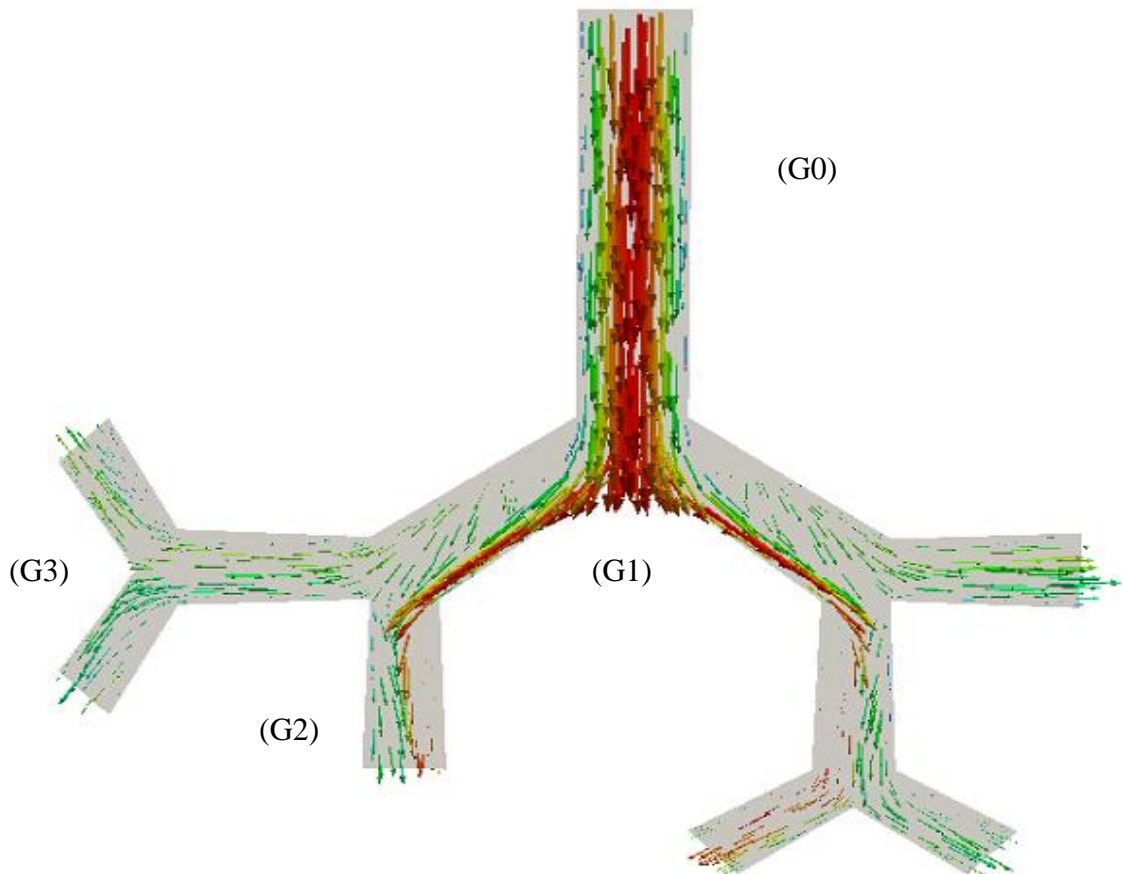


Figure 31. Velocity field for parabolic inlet velocity profile with $Re = 200$.

The velocity field shown in figure 31 can indicate the particle distribution for very small particles. For example, the high velocity region shown in figure 31 implies a high rate of transport for particles in that region. This is true because the advection term dominates the mass transfer in the high velocity areas. Since larger particle can't diffuse as fast as smaller particles, they are more likely to get deposited on the walls.

Figures 32 and 33 shows the concentration distributions for nanoparticles using both SUPG and GLS stabilization methods. The 1 nm particle gets deposited much more rapidly compared to the 150 nm particle. We can conclude that the nanoparticle DF will decrease when we increase particle diameter (up to a point), and this is due to reduced diffusivity. In figure 33 we can see for particles with larger diameter, there exists a local area where the concentration is the highest, and such a local area will not exist when the particle diameter is smaller, as shown in boxed region of figure 32 and figure 33. Re will also have an impact on the DF values but in this work we will not focus on this impact. This is consistent with other nanoparticle in human airways studies [6, 9-12, 14].

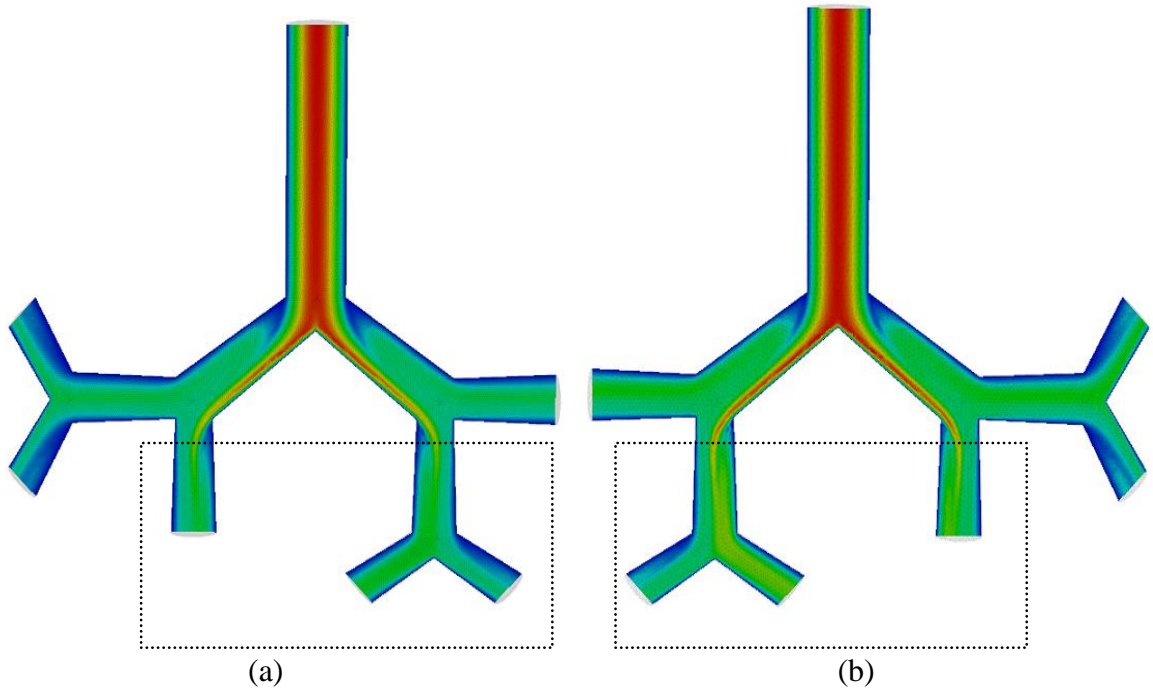


Figure 17. Concentration distribution for $d = 1(\text{nm})$, where (a) is with SUPG stabilization and (b) is with GLS stabilization.

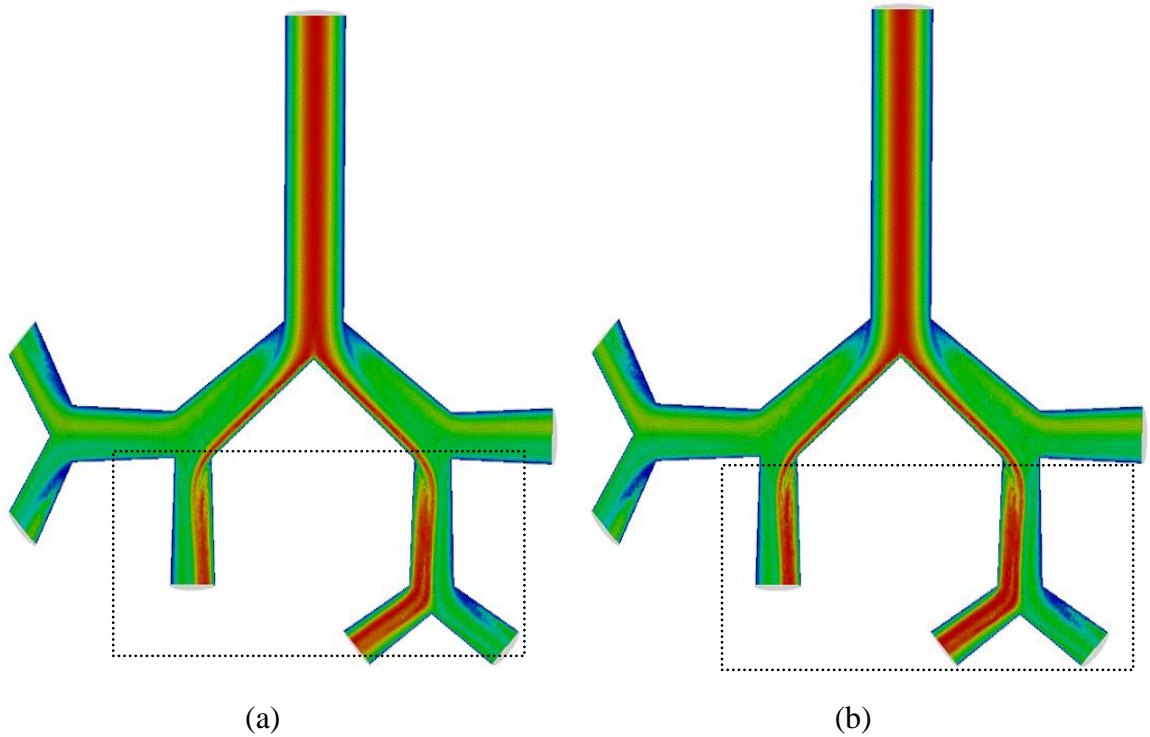


Figure 18. Concentration distribution for $d = 150(\text{nm})$, where (a) is with SUPG stabilization and (b) is with GLS stabilization.

We can conclude from the results presented here that the stabilization method used in these models provides an accurate approximation for the concentration distribution, and the best stabilization method for particles larger than 10 nm is GLS. In the next chapter we will summarize the major findings from this study and recommend future improvements for the model.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The main focus of this thesis is to better understand various diameter nanoparticles in human airways. Even though the particle diameter was very much in focus, other properties such as flow field, the geometry of human airways and stabilization method were also considered. There are many issues associated with the simulation of nanoparticles using regular FEM; especially instabilities when the diffusion coefficient is very small. In order to overcome this problem we tested several stabilization methods such as SUPG and GLS when modeling nanoparticles. While the stabilizations methods tested have been used and test in many other application, there are no other studies comparing these methods to the 3D Graetz problem. We have successfully compared different stabilization method with the Graetz Solution in 3D, where the best stabilization method was found to be GLS.

When testing the different stabilization methods, we observed that the temporal discretization method used has an impact on the error when comparing with the analytical Graetz solution. Previous studies have stated that CN temporal discretization is a poor option when the problem is advection dominated, but we have observed the opposite when there are stabilization terms included in the weak form [32, 33]. This is an interesting result, and more work on temporal discretization with stabilization will provide greater insight on this subject, although similar studies do exist but they only solve simple 1D and 2D problems.

Comparing the human airways results with varying particle diameter has shown that, for particles with large diameter (> 10 nm), the DF value is smaller compared to particle with smaller diameter (< 10 nm). This result agrees with previous studies, but those studies do not include the impact of different stabilization methods. In this thesis, we have shown that the type of stabilization methods chosen when the particle diameter is less than 10 nm do impact the DF value and when the particle diameter is greater than 10 nm the type of stabilization method used does not matter significantly. In the region where the particle size is larger, mesh refinement is important, and this is clearly shown in figure 30. When modeling human airways with commercially available software, such as ANSYS, the results are impacted by the type of stabilization method used in the software, and we do not have any means of knowing exactly which stabilization method is being used because of confidential information. Studies previous done using commercial software suffers greatly from this lack of information.

The human airways simulations have shown that when we inhale superfine particles (1 nm), the distribution is uniform and this leads to a greater deposition fraction and toxicity compared to larger particle (150 nm). This means that the superfine particle will cover more area than the larger particles, which will have localized concentration areas. This is due to smaller particle that have a much larger diffusion coefficient compared to larger sized particles.

Future work on this topic could be focused on creating a much more accurate model (i.e., larger numbers of elements) in order to understand the impacts of inhaling nanoparticle. This could lead to a better understanding and treatment for inhaled toxic

nanoparticles. Improving computing capabilities, for example, running simulation in petascale could enable patient specific diagnosis and targeted drug delivery.

REFERENCES

1. Jayaraju, S.T., *Study of the Air Flow and Aerosol Transport in the Human Upper Airways using LES and DES Methodologies*, in *Mechanical Engineering*. 2009, Vrije Universiteit Brussel: Brussel. p. 196.
2. Liu, Y., R.M. So, and C.H. Zhang, *Modeling the bifurcating flow in an asymmetric human lung airway*. *J Biomech*, 2003. **36**(7): p. 951-9.
3. V Sauret, P.H., et al., *Study of the three-dimensional geometry of the centralo conducting airways in man using computed tomographic (CT) images*. *Journal of Anatomy*, 2002. **200**(2): p. 123-134.
4. Liu, Y., R.M. So, and C.H. Zhang, *Modeling the bifurcating flow in a human lung airway*. *J Biomech*, 2002. **35**(4): p. 465-73.
5. Martonen, T.B., *Mathematical model for the selective deposition of inhaled pharmaceuticals*. *J Pharm Sci*, 1993. **82**(12): p. 1191-9.
6. Hogberg, S.M., *Modeling Nanofiber Transport and Deposition in Human Airways*, in *Department of Applied Physics and Mechanical Engineering*. 2010, Lulea University of Technology. p. 1-246.
7. Tronde, A., *Pulmonary Drug Absorption*, in *Pharmacy*. 2002, Uppsala University: Uppsala. p. 1-86.
8. Choi, L.-T., *Simulation of Fluid Dynamics and Particle Transport in Realistic Human Airways* in *School of Aerospace, Mechanical and Manufacturing Engineering*. 2007, RMIT University. p. 109.
9. H. Shi, C.Kleinstreuer, and Z.Zhang, *Nanoparticle transoprt and deposition in bifurcating tubes with different inlet conditions*. *Physics of Fluids*, 2004. **16**(7): p. 2199-2213.
10. Kleinstreuer, C., Z. Zhang, and Z. Li, *Modelingairflow and particle transport/deposition in pulmonary airways*. *Respratory Physiology & Neurobiology*, 2008. **163**: p. 128-138.
11. Z.Zhang, et al., *Comparison of micro- and nano-size particle depositions in a human upper airway model*. *Aerosol Science*, 2005. **36**: p. 211-233.

12. Zhe Zhang, Clement Kleinstreuer, and C.S. Kim, *Comparison of analytical and CFD models with regard to micron particle deposition in a human 16-generation tracheobronchial airways model*. *Aerosol Science*, 2008. **40**: p. 16-28.
13. Jayaraju, S.T., et al., *Fluid flow and particle deposition analysis in a realistic extrathoracic airway model using unstructured grids*. *Journal of Aerosol Science*, 2007. **38**(5): p. 494-508.
14. Kleinstreuer, C., *Two-phase flow : theory and applications*. 2003, New York ; London: Taylor & Francis. xiv, 454 p.
15. Wininger, C.W. and J.J. Heys, *Particle transport modeling in pulmonary airways with high-order elements*. *Math Biosci*. **232**(1): p. 11-9.
16. Thushar Gohil, et al., *Simulation of oscillatory flow in an aortic bifurcation using FVM and FEM : A comparative study of implementation strategies*. *International Journal for Numerical Methods in Fluids*, 2011. **66**: p. 1037-1067.
17. Tayfun Tezduyar and S. Sathe, *STABILIZATION PARAMETERS IN SUPG AND PSPG FORMULATIONS*. *Computational and Applied Mechanics*, 2003. **4**(1): p. 71-88.
18. Whiting, C.H., *STABILIZED FINITE ELEMENT METHODS FOR FLUIDS DYNAMICS USING HIERARCHICAL BASIS*, in *Mechanical Engineering*. 2009, Rensselaer Polytechnic Institute: Troy.
19. P.Rajaraman, et al., *Comparison of Continuous and Discontinuous Galerkin Finite Element Methods for Parabolic Differential Equations Employing Implicit Time Integration*. Submitted.
20. Tyree, C.A. and J.O. Allen, *Diffusional particle loss upstream of isokinetic sampling inlets (vol 38, pg 1019, 2004)*. *Aerosol Science and Technology*, 2005. **39**(7): p. 673-673.
21. Weibel, E., *Morphometry of the Human Lung*. Springer Verlag, 1963.
22. Jayaraju, S.T., et al., *Large eddy and detached eddy simulations of fluid flow and particle deposition in a human mouth-throat*. *Journal of Aerosol Science*, 2008. **39**(10): p. 862-875.
23. Jayaraju, S.T., et al., *Contribution of upper airway geometry to convective mixing*. *Journal of Applied Physiology*, 2008. **105**(6): p. 1733-1740.

24. Stahlhofen, W., *Experimental-Determination of the Regional Deposition of Aerosol-Particles in the Human Respiratory-Tract*. American Industrial Hygiene Association Journal, 1980. **41**(6): p. 385-398.
25. Stahlhofen, W., *Experimentally Determined Regional Deposition of Aerosol-Particles in the Human Respiratory-Tract*. Clinical Respiratory Physiology-Bulletin Europeen De Physiopathologie Respiratoire, 1980. **16**(2): p. P145-P147.
26. Stahlhofen, W., *Deposition of Inhaled Particles in the Human Respiratory-Tract*. Zentralblatt Fur Bakteriologie Mikrobiologie Und Hygiene Serie B-Umwelthygiene Krankenhaushygiene Arbeitshygiene Praventive Medizin, 1985. **181**(3-5): p. 458-458.
27. Stahlhofen, W., J. Gebhart, and J. Heyder, *Biological Variability of Regional Deposition of Aerosol-Particles in the Human Respiratory-Tract*. American Industrial Hygiene Association Journal, 1981. **42**(5): p. 348-352.
28. Stahlhofen, W., et al., *Intercomparison of Regional Deposition of Aerosol-Particles in the Human Respiratory-Tract and Their Long-Term Elimination*. Experimental Lung Research, 1981. **2**(2): p. 131-139.
29. Stahlhofen, W., et al., *Human-Lung Clearance of Iron-Oxide and Teflon Particles*. Journal of Aerosol Science, 1981. **12**(3): p. 216-217.
30. Stahlhofen, W., et al., *New Regional Deposition Data of the Human Respiratory-Tract*. Journal of Aerosol Science, 1983. **14**(3): p. 186-188.
31. Stahlhofen, W., et al., *Deposition Pattern of Droplets from Medical Nebulizers in the Human Respiratory-Tract*. Bulletin Europeen De Physiopathologie Respiratoire-Clinical Respiratory Physiology, 1983. **19**(5): p. 459-463.
32. Donea, J. and A. Huerta, *Finite Element Method for Flow Problems*. 2003: John Wiley & Sons Ltd. 350.
33. Gresho, P.M., R.L. Sani, and M.S. Engelman, *Incompressible flow and the finite element method*. 2000, Chichester [England] ; New York: Wiley.
34. Reddy, J.N., *An introduction to the finite element method*. 2nd ed. McGraw-Hill series in mechanical engineering. 1993, New York: McGraw-Hill. xix, 684 p.
35. Reddy, J.N. and D.K. Gartling, *The finite element method in heat transfer and fluid dynamics*. 1994, Boca Raton: CRC Press. 390 p.

36. Socolofsky, S.A. and G. H.Jirka, *CVEN 489-501: Special Topics in Mixing and Transport Processes in the Environment*. 2005. p. 184.
37. Bessems, D., *Development of an Extended Quadratic Tetrahedron for Finite Element Analysis of Navier-Stokes Problems*, in *Departement of Mechanical Engineering*. 2003, Eindhoven University of Technology. p. 37.
38. Faires, J.D. and R.L. Burden, *Numerical methods*. 3rd ed. 2003, Pacific Grove, CA: Thomson/Brooks/Cole. xii, 622 p.
39. Šolin, P., K. Segeth, and I. Dole*zel, *Higher-order finite element methods*. Studies in advanced mathematics. 2004, Boca Raton, FL: Chapman & Hall/CRC. xx, 382 p.
40. Greenbaum, A., *Iterative methods for solving linear systems*. Frontiers in applied mathematics 17. 1997, Philadelphia, PA: Society for Industrial and Applied Mathematics. xiii, 220 p.
41. Tyree, C. and J. Allen, *Diffusional particle loss upstream of isokinetic sampling inlets*. *Aerosol Science and Technology*, 2004. **38**(10): p. 1019-1026.
42. Heroux, M.A., *AztecOO User Guide*. 2007: p. 41.
43. Heroux, M.A., *Epetra Performance Optimization Guide*. 2009: p. 13.
44. Heroux, M.A., et al., *An overview of the Trilinos Project*. *Acm Transactions on Mathematical Software*, 2005. **31**(3): p. 397-423.
45. Heroux, M.A. and M. Sala, *The design of Trilinos*. *Applied Parallel Computing: State of the Art in Scientific Computing*, 2006. **3732**: p. 620-628.
46. Sala, M. and M.A. Heroux, *Robust Algebraic Preconditioner using IFPACK 3.0*. 2005: p. 29.
47. Kumar, G.K.a.V., *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*. 1998: p. 44.

APPENDIX A

HUMAN AIRWAYS MODEL IMPLEMENTATION IN C++

Human Airways Model Implementation In C++

This appendix is a summary of the C++ code that is used to build the human airways model. The important parts of the code will be discussed in detail. The main function is call fem3d, which includes all the prototypes, as shown below. There are 5 prototypes in this model: the first prototypes is build_local, which is the function where the isoparametric mapping is done, the assemble function is where the matrix and vector is assembled, initZero is where the matrix is initialized to zero before assembling the matrix, sol_out is the function called to write the results of the model, and, finally, mass is the function called to do the flux calculations.

```
int build_local(int, TriMesh *, RefElem *, double **, double *,
Epetra_Vector *, Epetra_Vector *, Epetra_Vector *, double);

void assemble(int, int, int, Epetra_CrsMatrix *, Epetra_Vector *,
double **, double *, TriMesh *, double, Epetra_Vector *);

void initZero(int, int, Epetra_CrsMatrix *, Epetra_Vector *, TriMesh *);

void sol_out(int *, Epetra_Vector *, TriMesh *, int, double);

int mass(TriMesh *, Epetra_Vector *);
```

```
for (int i=0; i<NumNodes; i++)
    pnop[i] = -1;
int counter = 0.0;
for (int e=0; e<NumElem; e++)
    for (int i=0; i<NPpE; i++)
        if( pnop[nop[e][i]] == -1)
            pnop[nop[e][i]] = counter;
            counter++;
mymesh.partition(myid,NumProc);
int counter = 0;
for (int e=0; e<NumElem; e++){
    for (int i=0; i<NPpE; i++){
        if (oldnew[nop[e][i]]== -1){
            oldnew[nop[e][i]]= counter;
            newold[counter] = nop[e][i];
            nop4[e][i]= counter;
            counter++;
        }else{
            nop4[e][i]=oldnew[nop[e][i]];
        }
    }
}
NumNodes4=counter;
METIS_PartMeshNodal(&NumElem, &NumNodes4, tmp_elem, &etype, &numflag,
&NumProc, &edgecut, epart, mpinpart);
```

The code above is called the Trimesh object and the RefElem object. The Trimesh object will read in the mesh file and organize the data from the file. In Trimesh we will also do the mesh partitioning for parallel computing using the Metis library, which is

shown at the very end of the code. In Trimesh, `pnop()` is the pressure `nop()` where, given an element number and a node number, we can know whether the node has a pressure unknown. Similarly, `nop()` is created for the other degree of freedom such as velocity and concentration, since all these unknowns are on every node in the element, we can lump them into a single `nop`. The `RefElem` object is where the basis functions for the tetrahedron are create and the Gauss Quadrature is defined.

```
//Create Map
Epetra_Map dofmap(-1,mymesh.num_local_dof,mymesh.MyLocalDof.Values(),
0, Comm);

Epetra_Map solmap(mymesh.NumDof,mymesh.NumDof, 0, Comm);

Epetra_Import FullImporter(solmap,dofmap);

//Create Vector & Matrix
Epetra_CrsMatrix A(Copy, dofmap, 10 * mymesh.localDof);
Epetra_Vector b(dofmap);
Epetra_Vector dsol(dofmap);
Epetra_Vector fulldsol(solmap);
Epetra_Vector sol(solmap);
Epetra_Vector sol_old(solmap);
Epetra_Vector sol_older(solmap);
```

The code above creates the maps, matrix, and vectors for the model, and this is done by using the Trilinos Epetra package.

```
for(int e=0; e<(mymesh.num_local_elem); e++){
    initZero(mymesh.MyElem[e],myid, &A, &b, &mymesh);//intiliaze A to
    zero
}
for (j=0; j<(mymesh->NpE);j++){
    dof = mymesh->nop[e][j];//dof= local nodes
    if (mymesh->npart[dof]==myid){
        switch(mymesh->BCType[mymesh->nop[e][j]]){
            case 1:
            case 2:
                NumEntries = 1;
                vals[0] = 0.0;
                cols[0] = dof;
                A->InsertGlobalValues(dof, NumEntries, vals, cols);
                break;
            case 3:
                NumEntries = 1;
                vals[0] = 0.0;
                cols[0] = dof;
                A->InsertGlobalValues(dof, NumEntries, vals, cols);
                break;
            default:
                NumEntries = 3*mymesh->NpE+mymesh->NpPE;
                for (i=0; i < mymesh->NpE; i++){
                    cols[i] = mymesh->nop[e][i];
                    vals[i] = 0.0;
                }
            for (i=0; i < mymesh->NpE; i++){
```

```

    cols[mymesh->NpE+i] = mymesh->NumNodes + mymesh->nop[e][i];
    vals[mymesh->NpE+i] = 0.0;
}
for (i=0; i < mymesh->NpE; i++){
    cols[2*mymesh->NpE+i] = 2*mymesh->NumNodes + mymesh->nop[e][i];
    vals[2*mymesh->NpE+i] = 0.0;
}
for (i=0; i < mymesh->NPpE; i++){
    cols[3*mymesh->NpE+i] = 4 * mymesh->NumNodes + mymesh->pnop[mymesh-
>nop[e][i]];
    vals[3*mymesh->NpE+i] = 0.0;
}
A->InsertGlobalValues(dof, NumEntries, vals, cols);
break;
}
}
}

```

The code above initializes the matrix A by filling it in with zeros, and the very first line is the call in the main() function. The next part of the code is the routine used to fill the matrix in zeros first, and this code is only a part of the function initZero. The code shown here will only fill the first row of the matrix. The case() statements are used to set the boundary conditions.

```
Ifpack Factory;

    Ifpack_Preconditioner *Prec;
    string PrecType = "ILU";
    Prec = Factory.Create(PrecType, &A);
    assert (Prec != 0);
    Teuchos::ParameterList List;

    Epetra_LinearProblem LP(&A, &dsol, &b);
    AztecOO Solver(LP);

    Prec->SetParameters(List);
    Prec->Initialize();

    Solver.SetPrecOperator(Prec);
    Solver.SetAztecOption(AZ_solver, AZ_gmres);
    Solver.SetAztecOption(AZ_output, 100);
```

The code above implements the Ifpack solver from the Trilinos package, and the solver uses ILU-type preconditioner with GMRES. Additional information about the solver can be found in Trilinos documentation.

```
for (int t_step=0; t_step < total_steps; t_step++){
    for (int i_ter=0; i_ter<iter; i_ter++){
        currentTime = (t_step+1.0) * dt;
        A.PutScalar(0.0); b.PutScalar(0.0);
        dsol.PutScalar(0.0); fulldsol.PutScalar(0.0);
```

```

    for(int e=0; e<(mymesh.num_local_elem); e++){
build_local(mymesh.MyElem[e], &mymesh, &basis, A_local, b_local, &sol,
&sol_old, &sol_older,dt);

assemble(mymesh.MyElem[e],myid, mymesh.NpE, &A, &b, A_local, b_local,
&mymesh, currentTime, &sol);

    }

    Prec->Compute();

    Solver.Iterate(1000, TOL);

    fulldsol.Import(dsol,FullImporter,Add);

    for (int i=0; i<mymesh.NumDof; i++){
        sol[i] = sol[i]-fulldsol[i];
    }

    //Condition for Newton

    b.Norm2(&norm);

    if(norm < TOL){
        i_ter = iter;
    }

} // End of iteration loop

//Write output

if(myid==0){
    if((t_step % 10) == 0){
sol_out(&ex_id, &sol, &mymesh, out_step, currentTime);

out_step++;

        //Solution update

        for (int i=0; i<mymesh.NumDof; i++){
            sol_old[i] = sol[i];
            sol_older[i] = sol_old[i];
        }
    }

} // End of time loop

```

The code above implements the time loop and Newton iteration steps, and after the iteration loop and time step loop we insert zeros into the matrix and vectors using Trilinos commands. After this step we loop thru the elements and call the two functions: `build_local()` and `assemble()`. The convergence condition for Newton is done by calculating the norm of the residual, which are calculated using the Trilinos package. After ending the iteration loop we write the solution to an Exodus file using the function `sol_out`. After writing the solution to a file we will update the solution for the old and older time steps since this is need for the temporal discretization before ending the time loop.

In `build_local ()`, the main steps are the isoparametric mapping and the weak form implementation, and these steps are shown below. The weak form shown below is for the advection-diffusion equation only.

```

for (int i=0; i<(basis->NpE); i++){
    x_xsi += mymesh->getNodeLocX(mymesh->nop[e][i]) * basis-
>dphi10_xsi[i][gpt];
    x_eta += mymesh->getNodeLocX(mymesh->nop[e][i]) * basis-
>dphi10_eta[i][gpt];
    x_del += mymesh->getNodeLocX(mymesh->nop[e][i]) * basis-
>dphi10_del[i][gpt];
    y_xsi += mymesh->getNodeLocY(mymesh->nop[e][i]) * basis-
>dphi10_xsi[i][gpt];
    y_eta += mymesh->getNodeLocY(mymesh->nop[e][i]) * basis-
>dphi10_eta[i][gpt];
    y_del += mymesh->getNodeLocY(mymesh->nop[e][i]) * basis-
>dphi10_del[i][gpt];
    z_xsi += mymesh->getNodeLocZ(mymesh->nop[e][i]) * basis-
>dphi10_xsi[i][gpt];
    z_eta += mymesh->getNodeLocZ(mymesh->nop[e][i]) * basis-
>dphi10_eta[i][gpt];

```

```

        z_del += mymesh->getNodeLocZ(mymesh->nop[e][i]) * basis-
>dphi10_del[i][gpt];
    }

detJ = (x_xsi*y_eta*z_del)+(z_eta*y_xsi*x_del)+(z_xsi*x_eta*y_del)-
(x_xsi*z_eta*y_del)- (y_xsi*x_eta*z_del)-
(z_xsi*y_eta*x_del);

//Isoparametric mapping

for (int i=0; i<(basis->NpE); i++){

    phi_x[i] = ((y_eta*basis->dphi10_xsi[i][gpt]*z_del)+(y_xsi*basis-
>dphi10_del[i][gpt]*z_eta)+(y_del*basis->dphi10_eta[i][gpt]*z_xsi)-
(y_del*basis->dphi10_xsi[i][gpt]*z_eta)-(y_xsi*basis-
>dphi10_eta[i][gpt]*z_del)-(y_eta*basis-
>dphi10_del[i][gpt]*z_xsi))/detJ;

    phi_y[i] = ((x_xsi*basis-
>dphi10_eta[i][gpt]*z_del)+(x_del*basis-
>dphi10_xsi[i][gpt]*z_eta)+(x_eta*basis->dphi10_del[i][gpt]*z_xsi)-
(x_xsi*basis->dphi10_del[i][gpt]*z_eta)-(x_eta*basis-
>dphi10_xsi[i][gpt]*z_del)-(z_xsi*basis-
>dphi10_eta[i][gpt]*x_del))/detJ;

    phi_z[i] = ((x_xsi*basis-
>dphi10_del[i][gpt]*y_eta)+(x_del*basis-
>dphi10_eta[i][gpt]*y_xsi)+(x_eta*basis->dphi10_xsi[i][gpt]*y_del)-
(x_xsi*basis->dphi10_eta[i][gpt]*y_del)-(x_eta*basis-
>dphi10_del[i][gpt]*y_xsi)-(x_del*basis-
>dphi10_xsi[i][gpt]*y_eta))/detJ;

}

for (int i=0; i<(basis->NpE); i++){

    for (int j=0; j<(basis->NpE); j++){

        A_local[3*basis->NpE+i][j]+=dt*(BDF1*basis-
>phi10[j][gpt]*c_x*basis->phi10[i][gpt] + BDF1*t_gls*basis-
>phi10[j][gpt]*((1.0/t_gls)*(basis->phi10[i][gpt])*c_x + phi_t*phi_x[i]
+ 2.0*unew*phi_x[i]*c_x + phi_x[i]*vnew*c_y + phi_x[i]*unew*c_z +
vnew*phi_y[i]*c_x + wnew*phi_z[i]*c_x))*detJ*basis->gw[gpt];

    }

    for (int j=0; j<(basis->NpE); j++){

        A_local[3*basis->NpE+i][basis->NpE+j]+=dt*(BDF1*basis-
>phi10[j][gpt]*c_y*basis->phi10[i][gpt] + BDF1*t_gls*basis-
>phi10[j][gpt]*((1.0/t_gls)*(basis->phi10[i][gpt])*c_y + phi_t*phi_y[i]
+ unew*phi_x[i]*c_y + unew*c_x*phi_y[i] + 2.0*vnew*c_y*phi_y[i] +
phi_y[i]*wnew*c_z + wnew*phi_z[i]*c_y))*detJ*basis->gw[gpt];

```

```

    for (int j=0; j<(basis->NpE); j++){
        A_local[3*basis->NpE+i][2*basis->NpE+j]+=dt*(BDF1*basis-
>phi10[j][gpt]*c_z*basis->phi10[i][gpt] + BDF1*t_gls*basis-
>phi10[j][gpt]*((1.0/t_gls)*(basis->phi10[i][gpt])*c_z + phi_t*phi_z[i]
+ unew*phi_x[i]*c_z + vnew*phi_y[i]*c_z + phi_z[i]*unew*c_x +
phi_z[i]*vnew*c_y + 2.0*wnew*phi_z[i]*c_z))*detJ*basis->gw[gpt];

        for (int j=0; j<(basis->NpE); j++){
            A_local[3*basis->NpE+i][3*basis->NpE+j]+=(basis-
>phi10[i][gpt]*basis->phi10[j][gpt] +
dt*(BDF1*D*phi_x[i]*phi_x[j]+BDF1*D*phi_y[i]*phi_y[j]+BDF1*D*phi_z[i]*p
hi_z[j]+BDF1*unew*phi_x[j]*basis-
>phi10[i][gpt]+BDF1*vnew*phi_y[j]*basis->phi10[i][gpt] +
BDF1*wnew*phi_z[j]*basis->phi10[i][gpt] +
BDF1*t_gls*((1.0/(t_gls*dt))*(basis->phi10[i][gpt]*basis-
>phi10[j][gpt]) + (1.0/t_gls)*(basis->phi10[i][gpt]*unew*phi_x[j]) +
(1.0/t_gls)*(basis->phi10[i][gpt]*vnew*phi_y[j])+(1.0/t_gls)*(basis-
>phi10[i][gpt]*wnew*phi_z[j]) + (1.0/dt)*(basis-
>phi10[j][gpt]*unew*phi_x[i]) + (1.0/dt)*(basis-
>phi10[j][gpt]*vnew*phi_y[i]) + (1.0/dt)*(basis-
>phi10[j][gpt]*wnew*phi_z[i]) + (unew*unew*phi_x[i]*phi_x[j]) +
(unew*phi_x[i]*vnew*phi_y[j]) + (unew*phi_x[i]*wnew*phi_z[j]) +
(vnew*phi_y[i]*unew*phi_x[j]) + (vnew*vnew*phi_y[i]*phi_y[j]) +
(vnew*phi_y[i]*wnew*phi_z[j]) + (wnew*phi_z[i]*unew*phi_x[j]) +
(wnew*phi_z[i]*vnew*phi_y[j]) +
(wnew*wnew*phi_z[i]*phi_z[j])))*detJ*basis->gw[gpt];
        }

    for (int j=0; j<(basis->NPpE); j++){
        A_local[3*basis->NpE+i][4*basis->NpE+j] += 0.0*detJ*basis->gw[gpt];
    }

    b_local[3*basis->NpE+i] += (cnew*basis->phi10[i][gpt] -
cold*basis->phi10[i][gpt] + dt*(BDF1*D*c_x*phi_x[i] +
BDF1*D*c_y*phi_y[i]+ BDF1*D*c_z*phi_z[i] + BDF1*unew*c_x*basis-
>phi10[i][gpt]+BDF1*vnew*c_y*basis->phi10[i][gpt] +
BDF1*wnew*c_z*basis->phi10[i][gpt] + BDF1*t_gls*((1.0/t_gls)*(basis-
>phi10[i][gpt])*phi_t + (1.0/t_gls)*(basis-
>phi10[i][gpt])* (unew*c_x+vnew*c_y+wnew*c_z) + phi_t*(unew*phi_x[i] +
vnew*phi_y[i] + wnew*phi_z[i])) + (unew*phi_x[i] + vnew*phi_y[i] +
wnew*phi_z[i])*(unew*c_x + vnew*c_y + wnew*c_z) +
(1.0/dt)*((BDF2*cold*basis->phi10[i][gpt] - BDF2*colder*basis-
>phi10[i][gpt]))))*detJ*basis->gw[gpt];
}

```

Assemble is the function used to fill the matrix and residual and where the

boundary conditions are set. The case() statement indicates where the boundary conditions are set.

```

for (j=0; j<(mymesh->NpE); j++){
    dof = 3*mymesh->NumNodes + mymesh->nop[e][j];
    if(mymesh->npart[mymesh->nop[e][j]]==myid){
        if (A->LRID(dof) == -1){
            cout << "ERROR in assemble" << endl;
        }
        x = mymesh->getNodeLocX(mymesh->nop[e][j]);
        y = mymesh->getNodeLocY(mymesh->nop[e][j]);
        z = mymesh->getNodeLocZ(mymesh->nop[e][j]);
        switch(mymesh->BCType[mymesh->nop[e][j]]){
            case 1:
                NumEntries = 1;
                vals[0] = 1.0;
                cols[0] = dof;
                A->ReplaceGlobalValues(dof, NumEntries, vals, cols);
                vals[0] = (*sol)[dof]-1.0*((1.0-y)*(1.0+y)*(1.0-x)*(1.0+x));
                b->ReplaceGlobalValues(NumEntries, vals, &dof);
                break;
            case 3:
                NumEntries = 1;
                vals[0] = 1.0;
                cols[0] = dof;
                A->ReplaceGlobalValues(dof, NumEntries, vals, cols);
                vals[0] = (*sol)[dof]-0.0;
                b->ReplaceGlobalValues(NumEntries, vals, &dof);

```

```

break;

case 2:

    NumEntries = 1;

    vals[0] = 1.0;

    cols[0] = dof;

A->ReplaceGlobalValues(dof, NumEntries, vals, cols);

    vals[0] = (*sol)[dof]-0.0;

b->ReplaceGlobalValues(NumEntries, vals, &dof);

break;

default:

NumEntries = 4*mymesh->NpE;

for (i=0; i < mymesh->NpE; i++){

    cols[i] = mymesh->nop[e][i];

    vals[i] = A_local[3*mymesh->NpE+j][i];

}

for (i=0; i < mymesh->NpE; i++){

    cols[mymesh->NpE+i] = mymesh->NumNodes + mymesh->nop[e][i];

    vals[mymesh->NpE+i] = A_local[3*mymesh->NpE+j][mymesh-
>NpE+i];

}

for (i=0; i < mymesh->NpE; i++){

    cols[2*mymesh->NpE+i] = 2*mymesh->NumNodes + mymesh-
>nop[e][i];

    vals[2*mymesh->NpE+i] = A_local[3*mymesh->NpE+j][2*mymesh-
>NpE+i];

}

for (i=0; i < mymesh->NpE; i++){

    cols[3*mymesh->NpE+i] = 3 * mymesh->NumNodes + mymesh-
>nop[e][i];

    vals[3*mymesh->NpE+i] = A_local[3*mymesh->NpE+j][3*mymesh-
>NpE+i];

```

```

    }

    A->SumIntoGlobalValues(dof, NumEntries, vals, cols);

    NumEntries = 1;

    vals[0] = b_local[3*mymesh->NpE+j];

    b->SumIntoGlobalValues(NumEntries, vals, &dof);

    break;

}

}

}

```

The final section of the main function is the flux calculation. Sum1 is the flux calculation for the inlet of the human airways. A similar calculation is done for the outlet; the flux calculation is done by multiplying the average area of the triangle, average concentration and finally the normal velocity.

```

double sum1 =0.0;

for (int e=0; e<mymesh->NumElem; e++){

    int counter = 0;

    for (int i=0; i<mymesh->NpE; i++){

        if(mymesh->BCTYPE[mymesh->nop[e][i]]== 1){

            counter++;}}

    if (counter == 6){

        int locCount = 0;

        for (int i=0; i<mymesh->NpE; i++){

            if(mymesh->BCTYPE[mymesh->nop[e][i]]== 1){

                nop1[locCount] = i;

```

```

        locCount++;}}

    a_1=mymesh->nop[e][nop1[0]];
    b_1=mymesh->nop[e][nop1[1]];
    c_1=mymesh->nop[e][nop1[2]];
    x1 = mymesh->getNodeLocX(a_1);
    y1 = mymesh->getNodeLocY(a_1);
    z1 = mymesh->getNodeLocZ(a_1);
    x2 = mymesh->getNodeLocX(b_1);
    y2 = mymesh->getNodeLocY(b_1);
    z2 = mymesh->getNodeLocZ(b_1);
    x3 = mymesh->getNodeLocX(c_1);
    y3 = mymesh->getNodeLocY(c_1);
    z3 = mymesh->getNodeLocZ(c_1);

    BA_x=x2-x1;
    BA_y=y2-y1;
    BA_z=z2-z1;
    CA_x=x3-x1;
    CA_y=y3-y1;
    CA_z=z3-z1;
    BC_x=x2-x3;
    BC_y=y2-y3;
    BC_z=z2-z3;

    distance_x=0.5*((BA_y*CA_z)-(BA_z*CA_y));
    distance_y=0.5*((BA_z*CA_x)-(BA_x*CA_z));
    distance_z=0.5*((BA_x*CA_y)-(BA_y*CA_x));

    dot_x=distance_x*distance_x;
    dot_y=distance_y*distance_y;
    dot_z=distance_z*distance_z;
    add_1=(dot_x+dot_y+dot_z);

```

```
area_1=pow(add_1,0.5); // area of inflow

U_x=a_1;
U_y=DOF_y+a_1;
U_z=DOF_z+a_1;
V_x=b_1;
V_y=DOF_y+b_1;
V_z=DOF_z+b_1;
W_x=c_1;
W_y=DOF_y+c_1;
W_z=DOF_z+c_1;
C_x=DOF_c+a_1;
C_y=DOF_c+b_1;
C_z=DOF_c+c_1;

Vel_1=(*sol)[U_x]*(*sol)[U_x]+(*sol)[U_y]*(*sol)[U_y]+(*sol)[U_z]*(*sol)
)[U_z];

Vel_2=(*sol)[V_x]*(*sol)[V_x]+(*sol)[V_y]*(*sol)[V_y]+(*sol)[V_z]*(*sol)
)[V_z];

Vel_3=(*sol)[W_x]*(*sol)[W_x]+(*sol)[W_y]*(*sol)[W_y]+(*sol)[W_z]*(*sol)
)[W_z];

Con_total=( (*sol)[C_x]+(*sol)[C_y]+(*sol)[C_z])/3.0;

Vel_mag1=pow(Vel_1,0.5);
Vel_mag2=pow(Vel_2,0.5);
Vel_mag3=pow(Vel_3,0.5);

Vel_total=(Vel_mag1+Vel_mag2+Vel_mag3)/3.0;

sum1+=area_1*Vel_total*Con_total;
```

The final implementation that is shown below is the function which calculation the Graetz solution, this function is called the eigen function. Further details can be found in the reference paper [20].

```

for (int j=0; j<(mymesh->NumNodes); j++){
    x = mymesh->getNodeLocX(j);
    y = mymesh->getNodeLocY(j);
    z = mymesh->getNodeLocZ(j);

    z_new = (z) / (Re*Sc);
    rad= ((x*x) + (y*y));
    r= pow(rad,0.5);
    rm= pow(r,2.0);
    phi_rz[j] = 0.0;

    for(int n=1; n<=Nmax; n++){
        Z_n = 0.0;
        Y_n[n-1] = 0.0;
        m=1.0;
        for (int i=1; i <= 10; i++){
            F_n=1.0;
            for (int p=1; p<=i; p++){
                F_n *= (4*p-2.0-ev[n-1]);
            }
            dn=pow(4,i);
            kn=pow(ev[n-1],i);
            rn= pow(r,(2*i));

```

```
m=factorialFinder(i);
    Z_n += ((F_n*(kn*rn))/(dn*m*m));
}
en= -1.0*((ev[n-1]*rm)/(2.0));
Y_n[n-1] = exp(en)*(1.0 + Z_n);
double zn = pow(ev[n-1],2.0);
phi_rz[j] += (ec[n-1]*Y_n[n-1]*exp(-1.0*zn*z_new));
}
}
}
int factorialFinder (int num) {
    if (num==1){
        return 1;
    } else {
        return num*factorialFinder (num-1);
    }
}
```
