MODELING OF HUMAN RESPIRATORY FLOW

A Thesis in
Aerospace Engineering

by
Gulkiz Dogan

© 2012 Gulkiz Dogan

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2012
The thesis of Gulkiz Dogan was reviewed and approved* by the following:

Robert F. Kunz
Professor of Aerospace Engineering
Thesis Advisor

Philip J. Morris
Boeing/A.D. Welliver Professor of Aerospace Engineering

George A. Lesieutre
Professor of Aerospace Engineering
Head of the Department of Aerospace Engineering

*Signatures are on file in the Graduate School
This work represents a contribution to and advancement of several components of a semi-automated subject-specific end-to-end medical image through CFD (computational fluid dynamics) analysis capability for the human respiratory system. Specifically, this thesis presents geometric segmentation, manipulation and grid generation algorithms for the human lung, development of quasi-one-dimensional (Q1D) geometric and flow modeling approaches for the unresolved “convective regime,” verification and validation of pulsatile pressure-forced boundary conditions, a script-based multidisciplinary simulation framework for respiration, and a number of representative CFD simulations of respiration.

In the area of geometric modeling, four specific contributions are made: i) semi-automated processing of medical image data to derive upper airways and lobe geometries for in-vivo subjects, ii) creation of partitioning and truncation algorithms, with application to the conducting airways of a rubber cast model of a dead subject, iii) automated unstructured 3D gridding of the trachea through generation 5-8, and, iv) interfacing this upper bronchi and lobe geometry with a volume filling algorithm for the sub-resolved bronchi.

For unsteady pressure-forced flows, as in the respiration simulations pursued here, the specification of well posed and accurate streamwise boundary conditions are of concern, especially where inflow and outflow from pressure boundaries arise. The adequacy of the boundary condition approaches taken in this work are demonstrated by comparison of unsteady 2D/3D CFD simulations with known analytical solutions to the
incompressible Navier-Stokes equations for non-dimensional frequencies and Reynolds numbers of relevance to respiration.

These contributions are summarized in detail and employed in concert with other elements of a respiratory simulation framework under development at Penn State University.
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... vii

LIST OF TABLES .............................................................................................................. x

ACKNOWLEDGEMENTS ................................................................................................. xi

Chapter 1 INTRODUCTION ......................................................................................... 1

1.1 Motivation ............................................................................................................... 1
1.2 Anatomy of the Lung ............................................................................................. 2
   1.2.1 Anatomy of the Branching Airways ............................................................. 2
   1.2.2 Anatomy of the Lung Lobes ........................................................................ 4
1.3 Previous Work: Geometric Modeling of the Lung .................................................. 5
1.4 Previous Work: Experimental Studies of Respiration ......................................... 9
1.5 Previous Work: Human Respiratory System Simulations ..................................... 11
   1.6 Previous Work: Particle Deposition ............................................................... 15
1.7 Summary ................................................................................................................ 16
1.8 Organization of the Thesis ..................................................................................... 17

Chapter 2 LUNG GEOMETRIES ............................................................................. 18

2.1 Rubber Cast Model ............................................................................................... 18
2.2 Image Processing from Computed Tomography ................................................... 21
   2.2.1 Segmentation of Airways .......................................................................... 23
   2.2.2 Thinning ..................................................................................................... 29
   2.2.3 Segmentation of Lung Lobes ..................................................................... 32
2.3 Processing of Medical Imaging Geometry for CFD ............................................. 34
   2.3.1 Partitioning ................................................................................................ 34
   2.3.2 Truncation .................................................................................................. 37
   2.4 Branching Algorithm / Filling the Lobes with Artificial Bronchioles... 41

Chapter 3 GRID GENERATION FOR LUNG GEOMETRIES ................................. 45

3.1 Introduction .......................................................................................................... 45
   3.1.1 Structured Meshing ..................................................................................... 46
   3.1.2 Unstructured Grid Generation .................................................................... 46
   3.1.3 Gridding for the Lung Geometry ............................................................... 48
3.2 Octree Based Gridding of the Upper Airways ...................................................... 49
3.3 Quasi-One-Dimensional Gridding of the Lower Branches .................................... 55
   3.4 Interfacing the Upper and Lower Branch Grids ................................................. 57

Chapter 4 THEORETICAL FORMULATION ......................................................... 59
LIST OF FIGURES

Figure 1.1: Diagrammatic representation of conductive, transitional, and respiratory zones of the airways, [1]. ................................................................. 4

Figure 1.2: Diagram from http://www.le.ac.uk/pa/teach/va/anatomy/case2/2_1.html ................................. 5

Figure 1.3: a) Tree structure for the Weibel airway model, Ref. [1]. b) Tree structure for the Horsfield airway model, Ref. [16]. ............................................. 6

Figure 2.1: Photograph of the rubber cast model with overlaid CFD ......................... 19

Figure 2.2: View of STL file of rubber cast model .................................................. 20

Figure 2.3: Skeleton of rubber cast model based on topological information .......... 20

Figure 2.4: 3-D view of a CT scans in MEVISLAB ................................................. 22

Figure 2.5: Dicom header file in MEVISLAB ......................................................... 22

Figure 2.6: Automatically segmented image by AMIRA ......................................... 24

Figure 2.7: Pool for automated segmentation of lung mask in AMIRA .................... 26

Figure 2.8: Lung mask after removing of islands in HARPOON ............................ 26

Figure 2.9: Showing of leakage in airway geometry in automated segmentation ... 27

Figure 2.10: Segmentation tool bar and removing islands option in AMIRA ........... 28

Figure 2.11: The upper airway is saved as closed geometry in STL format in AMIRA ............................................................. 29

Figure 2.12: Script to obtain the skeleton of upper airways. .................................. 30

Figure 2.13: Skeleton of upper airways, curved lines ............................................. 31

Figure 2.14: Skeleton of upper airways, straight lines .......................................... 32

Figure 2.15: a) Right upper (green), middle (blue) and lower lobe (pink) segmentations ................................................................. 33

Figure 2.16: Edge (branch) and node (bifurcation) numbering .............................. 35

Figure 2.17: Partitioning of airways by GEO_LUNG ............................................. 37
Figure 2.18: Schematic representation of the truncation of surface triangles for inlet. ................................................................. 38

Figure 2.19: Truncated outlets closed with new surfaces ............................................................. 39

Figure 2.20: Truncation of the rubber cast model ........................................................................... 40

Figure 2.21: Surface representation of right upper lobe (green), right middle lobe (blue), right lower lobe (purple), left upper lobe (yellow) and left lower lobe (orange) with and without upper airways (red). ...................................................... 43

Figure 2.22: Artificially generated bronchioles (after 5th generation). ........................................ 44

Figure 3.1: HARPOON mesh with hanging nodes ........................................................................... 49

Figure 3.2: HARPOON’s configuration file for upper airways ...................................................... 52

Figure 3.3: Octree based HARPOON upper airway grid. Surface geometry with six different parts and 32 outlets ...................................................................................... 53

Figure 3.4: Octree based HARPOON upper airway grid. a) Cut plane xz. b) Cut plane xy. c) Cut plane yz. .................................................................................................................. 54

Figure 3.5: HARPOON tetra mesh of lobes .................................................................................... 55

Figure 3.6: Q1D grid for a typical subtree section, ns=4, nc=12. Wall-less bifurcation element indicated by red circle. .......................................................... 56

Figure 3.7: Wall-less bifurcation element interfacing upper and lower airways. Element indicated by red circle. .......................................................... 57

Figure 3.8: View of a complete lung grid. 207,903 elements in upper airways, 754492 elements in lower airways. .......................................................... 58

Figure 4.1: Hierarchal data structure in NPHASE-PSU ................................................................. 61

Figure 4.2: Geometry nomenclature for cell face evaluations .................................................... 62

Figure 4.3: Predicted surface particle volume fraction and deposition efficiency vs. generation for a 5-7 generation CFD model. From Kunz et al. 2003 [14] ......... 67

Figure 4.4: Predicted surface oxygen content at a particular timestep during inspiration for a 10-12 generation CFD model. From Leemhuis 2004 [12] ........ 68

Figure 4.5: Schematic representation of the software infrastructure ........................................... 72

Figure 5.1: Bessel functions of the first kind at order zero, given in Eq. 5.14. ........ 79
Figure 5.2: Poiseuille velocity profile, nondimensionalized by its maximum, $u_{\text{steady}}(0)$. .................................................. 82

Figure 5.3: Oscillatory flow profiles at different phase angles $\omega t$, nondimensionalized by $u_{\text{steady}}(0)$. Phase angle ranges from 0 to 360 degrees by 45 degrees. Nondimensional frequency, $\Omega=3$. .................................................. 83

Figure 5.4: Oscillatory flow profiles at different phase angles ($\omega t$), nondimensionalized by $u_{\text{steady}}(0)$. Phase angle ranges from 0 to 360 degrees by 45 degrees. Nondimensional frequency, $\Omega=1$. .................................................. 84

Figure 5.5: Straight rectangular duct ................................................................. 85

Figure 5.6: Structured grid for cubic rectangular duct........................................ 89

Figure 5.7: Velocity profiles for periodic channel flow. Phase angle ranges from $3.6^\circ$ to $39.6^\circ$. ................................................................. 92

Figure 5.8: Velocity profiles for periodic channel flow. Phase angle ranges from $36^\circ$ to $180^\circ$. ................................................................. 95

Figure 5.9: Velocity profiles for periodic channel flow. Phase angle ranges from $216^\circ$ to $360^\circ$. ................................................................. 96

Figure 5.10: Velocity profiles for periodic channel flow. A closer look at the transition from inflow to outflow. .................................................. 97

Figure 6.1: Erna upper airways pressure distribution and velocity....................... 99

Figure 6.2: Pressure distribution of Q1D lower airways of Erna; maximum generation number is 10................................................................. 101

Figure 6.3: Pressure distribution of Q1D lower airways of Erna; maximum generation number is 13................................................................. 101

Figure 6.4: Pressure distribution of Q1D lower airways of Erna; maximum generation number is 19................................................................. 102

Figure 6.5: Predicted local pressure distributions at four times during a periodic cycle; maximum inspiration, full inhale, maximum expiration and full exhale... 104
LIST OF TABLES

Table 5.1: Flow parameters for the cubic channel oscillatory flow test case ............90

Table 6.1: Simulated Q1D models.................................................................100
ACKNOWLEDGEMENTS

This thesis program is supported by an NIH research grant. The research is conducted under the supervision of Dr. Robert F. Kunz from Penn State University. I would like to thank Dr. Robert Kunz for his support, guidance and suggestions.

I also would like to thank Dr. Philip J. Morris from Penn State University for reading and evaluating the thesis. I would also like to thank Dr. Daniel C. Haworth from Penn State University for providing the volume filling algorithm and to thank Dr. Andres Kriete from Drexel University for provision of 3D medical images.
Chapter 1

INTRODUCTION

1.1 Motivation

Respiration is the process by which organisms absorb oxygen and excrete carbon dioxide. Like many system level biological functions, respiration is characterized by: coupled structural, flow and biochemistry components, complex geometries, dynamic function and a wide range of scales (macro, micro and molecular). These physics challenge the modeling of breathing, particularly in humans, where experimental observation is limited in detail, and geometries and function vary widely (i.e., different subjects, disease states, age).

Despite these challenges, there are a number of motivating benefits to respiratory modeling in humans. First, basic understanding can be enhanced through parametric exercising of a suitably physics-instrumented model. For example, such a model could improve our understanding of the deteriorating functionality of the lung as we age, and could improve our understanding and diagnosis of emphysema or advanced lung failure. More specifically, it could help us better understand the geometric characteristics of asthma, where the effective inside diameter of the affected bronchioles decreases, and their flow resistance becomes higher. Improved understanding of the effect of smoking,
pollution and other inhaled substances and the performance of inhaled pharmaceuticals are also approachable with modeling.

A second benefit to respiration modeling is clinical in nature. Indeed, the National Institutes of Health (NIH) research grant that funded this work is focused on developing a clinically usable simulation process. The ultimate goal of that research, to which the present thesis contributes, is a capability where a physician acquires an MRI (or other class of) medical image of a patient, and through an automated process, the image is transformed into a geometry, a mesh is generated, the flow is solved and finally useful diagnostic information is generated. The physician could then look at the efficiency of the lungs, distinguishing healthy, damaged or diseased tissue. With this information he or she could potentially diagnose disease and determine an appropriate course of treatment, including a specific dosage of a drug. Potentially, a useful quantitative result from the model could be obtained. For example, deposition efficiency could be used to help defining a dosage of a drug.

1.2 Anatomy of the Lung

1.2.1 Anatomy of the Branching Airways

As illustrated in Figure 1, there are approximately 23 generations of airways in the adult human lung. Airways distribute air to and from alveoli, the smallest gas-exchange unit in the lung. Generally, the first 3 or 4 generations are surrounded by
cartilage, a dense connective tissue, and these are referred to as branches. The remaining airways are termed bronchi.

Adopting one of several common conventions, the 0th generation corresponds to the trachea, which extends from the throat to the branch bifurcation of the left and right lungs. The trachea and the following 16 generations are the conducting airways. There are no alveoli in the conducting airways and therefore no gas exchange in this region, [1].

Approximately three generations following the conducting airways are partially alveolated respiratory bronchioles. They are followed by approximately another three to four generations which are fully alveolated. These post-conduction regions of approximately seven generations are termed the respiratory units or acini.

The lung of a newly born baby has the same number of conducting airways as an adult, but the number of generations in the respiratory units increases with age, [1].
A bifurcation is the geometrical region where an airway bifurcates, that is, divides into separate airways. A morphologically realistic bifurcation is constructed from a parent and two daughter airways. Daughter branches are narrower than their parent and their cross-section usually narrows as they decline from each other at the branching angle.

1.2.2 Anatomy of the Lung Lobes

As illustrated in Figure 1.2, after the trachea, the respiratory system is separated into the right lung and the left lung. In the right lung, there are three lobes, upper, middle and lower. In the left lung, due to the presence of the heart, there are only two lobes, upper and lower.
1.3 Previous Work: Geometric Modeling of the Lung

Due to the paucity of experimental data and inherent resolution limitations of modern medical scanning technology, theoretical considerations have historically had to be applied to model the anatomy of the lung.

Classical anatomical models define lung airway geometry in a symmetric fashion, including bronchiol length, diameter and parent-daughter branching angles as a function only of airway generation. The Weibel model, Ref. [1], is the best known in this class. In Weibel generation numbering, branches are counted from the trachea. The trachea is designated generation 0. Each child branch is numbered one higher than its parent as shown in Figure 3a. If generation number is represented by n, then each generation has $2^n$ branches. Lengths and diameters of branches are also function of generation number, that
is, all branches in a generation are geometrically identical to one another. Generally, symmetric models can represent bulk respiration physics such as ventilation or gas mixing, and are therefore important for understanding the global physics of the problem but are not useful for local conditions assessment or for deriving subject specific conclusions.

The Horsfield model, introduced by Horsfield et al. [16] in 1971 is an example of an asymmetric model. Asymmetry is defined by $\delta$, the difference in Horsfield order of two daughter branches that arise from the same parent branch. The lowest order is 1 and it is assigned to the terminal branches while the parent branches are one order higher than the daughter branch of the highest order. A tree with $\delta=0$ is symmetric. A schematic for a tree with $\delta=0$ is given in Figure 1.3b.

![Tree structure for the Weibel airway model, Ref. [1].](image1)

![Tree structure for the Horsfield airway model, Ref. [16].](image2)

In addition to models that describe the lung geometry statistically, such as the symmetric and asymmetric models described above, a number of researchers have
focused on more detailed geometric modeling of branch bifurcations themselves, since
gas flow distributions, secondary flows and particle deposition are strongly dependent on
bifurcation geometry. While some researchers do not explicitly define the geometry of
the central transition zone in three dimensional bifurcation models, or in some cases use
square cross section bifurcations, other researchers have attempted to characterize the
central zone in a more biologically representative manner. Hegedus et al. [30]
summarized such models: Gradon and Orlicki [56] described a branching model
consisting of a sequence of interpenetrating cylindroids defined by three classes of
rational functions without incorporating branching asymmetry. Balásházy and Hofmann,
[57] and Balásházy, [58] constructed idealised “narrow” and “wide” bifurcations by
connecting cylindrical parent and daughter branches with different type transition zones.
Zhang and Kleinstreuer, [59] and Zhang et al. [60] have published airway bifurcation
geometries with curved carina and smooth transitions between the airways based on
computer-aided design (CAD) models of the surface information of experimental glass
tube bifurcation models, however, there is no further information about this technique
and there is no detailed mathematical description of the geometry presented in their
studies. Heistracher and Hofmann, [61] followed a study given by Horsfield et al. in 1971
to describe physiologically realistic bifurcation (PRB). PRB geometry approximates the
shape of human airway bifurcations in an appropriate way, an asymmetric airway
bifurcation with smooth transitions. In Ref. [30], Hegedus et al. describe an asymmetrical
morphologically realistic bifurcation model (MRB). Although the geometry is basically
the same as given by Heistracher and Hofmann, the mathematical description is quite
different. Specifically, the geometry is constructed of a cylinder and a ring with
continuously changing diameter and fixed curvature radius, and terminating with another cylinder, as shown in Figure 1.4. Eleven input parameters characterize the structure: lengths and diameters of the airways, curvature radii of daughter branches, maximal curvature radius of rounding carinal junction, and branching angles of the daughter airways. The geometry is restricted by the assumption that the three axes of the bifurcation are in one plane and thus the branching has at least one symmetry plane.

As medical imaging technology matures, researchers in many biomedical fields have moved towards the direct use of clinical imagery in defining geometry. Identification of the regional boundaries of the lung, lobes and airways from clinical images is known as segmentation. The current state-of-the-art in magnetic resonance imaging (MRI) and CAT scan (CT) technology allows the physician to view areas of the lung in cross-section, down to about the 10th or 12th generation, but there are limitations to this technology, [38]. First, it is nearly impossible to distinguish the smaller bronchi in these images to a spatial resolution that enables accurate geometric reconstruction.
Second, the highest resolution MRIs offer only static images, rather than dynamic sequences of a functioning lung during the respiratory cycle (although dynamic MRI technology is advancing). Nevertheless, several groups have pursued this approach for lung geometry modeling, [14], and, as described in Chapter 2, this is the approach taken in this thesis.

1.4 Previous Work: Experimental Studies of Respiration

Numerous experimental studies in the area of human respiration have appeared in the literature, though by virtue of the complexity of instrumenting live human subjects, \textit{in-vivo} studies are of limited scope. For example, it is extremely difficult to perform experiments which target local particle deposition in the lungs of humans (although some work has been done using radioactive gases and medical imaging to assess local perfusion, i.e., \textit{gas} transport/diffusion.) Also, experiments with animals are of limited use because deposition and diffusion processes are highly dependent on geometry, and all animals have lung structures that vary significantly from humans.

The human \textit{in-vivo} studies that have been performed are generally limited to quantification of breathing dynamics parameters and other global (non-local) parameters of interest. For example, the recent experimental study of Ref. [13] aimed to quantify fractal scaling properties of human respiratory dynamics and to determine whether these properties vary with age and gender. For this purpose, an experimental pool was created from healthy subjects of varying age and gender. The following results were obtained: i) there are fractal correlations in human respiratory dynamics; ii) the mean and range of the
interbreath interval did not show an important dependence on age or gender; iii) interbreath interval fractal correlations have both age and gender dependence.

*In-vitro* studies are more invasive and local measurements can be made at the expense of the accuracy of in-vivo geometry, boundary conditions and dynamics. They are based on postmortem human airway casts. Mucosa may shrink after death and processes in postmortem airways can distort the cast. Therefore, models obtained from postmortem airways may not represent a one to one living human airway.

In 1960 by using water and dye, investigation of flow distributions in a plastic airway cast of the trachea and the first bronchi was made Ref. [47]. Particle deposition and detailed geometrical description of a cast down to the segmental level were studied in 1976, Ref. [48]. Velocity profiles in trachea and main bronchi at different tracheal flow rates were measured with a hot film anemometer in a model of a human lung in 1983, Ref. [49].

Some experimental studies use scale modeling. Since the lower airways are successively smaller, most experiments in the literature to study the physical scale of the airflow in the lung are limited to 3rd or 4th generation. Schroter and Sudlow, Ref. [40] made another experimental study of airflow dynamics within a human respiratory network. They performed flow visualization experiments using smoke in a single symmetric bifurcation model (two generations) and the flow was at Reynolds numbers (based on local diameter) ranging from 50 to 4500. They concluded that the inspiratory flows were independent of Reynolds number and entry velocity profiles. Pedley, Schroter and Sudlow experimentally studied the pressure drop in the flow of Weibel’s idealized model airways, Ref. [51]. Schreck measured velocities in again Weibel’s bifurcation
model by using a hot wire anemometer, Ref. [52]. Experimental studies conducted by Chang, Isabey, Pedley, Zhao Refs. [41], [42], [43] and [44] respectively, concluded that the respiratory flow patterns were likely dependent on the airway geometry and except for the studies made by Chang, Isabey and Menon, Refs. [41], [42] and [45], respectively who investigated the airflow dynamics in the central airway up to the third generation of the bifurcation, most experiments used a simply symmetric bifurcation of the airway. On the other hand, studies using tubes with canonical contractions are important for modeling the aerosol deposition in the lungs of patients with diseases that cause airway contraction. Ref [28] is an example of a study of the particle deposition in tubes with conical contractions. In this study, a vibrating orifice aerosol generator was used to validate the numerical modeling of the axisymmetric laminar flow fields in contracted tubes.

1.5 Previous Work: Human Respiratory System Simulations

As indicated in Section 1.4, respiratory fluid mechanics experimental studies are inherently limited in their scope. Accordingly, numerical studies are indispensable to study the respiration fluid mechanics. A major challenge in flow analysis of respiration is starting with a reasonably accurate geometry.

Numerical studies that have used mathematically modeled geometry can be classified into two categories: typical path models and multiple path models. Typical path models idealize the lung geometry leading to a totally symmetrical lung. They use one typical path to represent the whole conduction zone and various loss formulas to calculate
deposition in each region. The lower generations can be assumed to be symmetrical, but the upper airway generations exhibit significant asymmetry. Such symmetrical approaches lead to unrealistic distributions of airflow to the lung lobes, which in turn can lead to inaccurate calculation of gas uptake and particle deposition within the airways.

Multiple path models incorporate the physically modeled asymmetry of the airways in the lung branching structure. Multiple path models can return more accurate upper airway flow distributions and thereby improved local particle deposition and gas uptake in the lung.

Mortonen made a numerical study in the Weibel model and compared the results to Schrecks’s experimental results, Ref. [53]. Wilquem simulated the air flow in a two dimensional Weibel ideal airway model for axial flow patterns, Ref. [4]. Liu simulated three dimensional airflow in Weibel’s airway geometry of up to 7th generation, Ref. [3]. Calay used the three-dimensional asymmetric bifurcation model of the central airway based on the morphological data given by Horsfield to simulate the unsteady oscillatory respiration, [5]. Ramuzat studied the Weibel geometry numerically and validated this study by particle image velocimetry in a three dimensional bifurcation model, [6].

Recently, numerical studies have begun to employ subject-specific geometries, that is, real lung airway geometries. This capability has arisen from advancements in magnetic resonance imaging technology and computer based scanning technology. Today, it is possible to obtain subject specific geometry of individuals’ lobes and upper airways. Researchers at the University of Iowa have the capability of having high quality imagery. In Ref. [38] the authors describe the steps of image processing for lung problems.
In Ref. [7], Kabilan, Lin and Hoffmann studied the airflow numerically in a sheep lung geometry segmented from CT images. They compared this flow with the flow in Weibel’s human lung model. They concluded that the flows in these models are completely different: no agreement between the two models at any of the cross sections.

In Ref. [54], Gemci et al. simulated the flow in 17 generation anatomical reference model of the human lung by a commercial CFD code. The geometry is obtained from a study of another research group, Schmidt, Ref. [55]. Large eddy simulation is applied. Pressure distribution of entire geometry and the velocity magnitude at generation 1 are presented. They obtained more complex velocity fields than Weibel based symmetric or asymmetric models. It is shown that the nature of the secondary vertical flows varies with the specific anatomical characteristics of the branching airways, [54].

In Ref. [50], Lin and Hoffmann studied the transport of Xe and He in the CT-based human lung geometries, especially in the wash out and wash in processes at upright and supine body posture. They applied a Galerkin finite element method to solve the three-dimensional incompressible variable-density Navier-Stokes equations. Here Xe is the denser gas while He is the light gas. The simulation results show that the gas transport process depends on the gas density and the body posture (gravity). They observed the Rayleigh-Taylor instability which may occur if dense gas is introduced into light gas from above during wash in process.

In Ref [8], Luo and Liu from The Hong Kong Polytechnic University, used a five generation CT scanned human lung model with an extended trachea to investigate flow characteristics at different Reynolds number and breathing rates by a low Reynolds number κ–ω turbulent model. They observed that in the inferior lobar bronchi, there are
two stems in which the axial velocity is stronger but secondary velocity is weaker while secondary flow in the lateral bronchi is stronger than the medial ones. With increasing Re number, the air flow increases in the middle, inferior lobes and left main bronchus, i.e., flow biases to left and downward.

Another study from the same university examined the effect of inlet velocity profile on the flow features in obstructed airways, Ref. [9]. They compared the four types of inlet boundary conditions: uniform, parabolic, positive-skewed parabolic and negative-skewed parabolic. They put the obstruction at either the second generation or the third generation of medial branches of natural human lung. The results showed that the inlet velocity profile has significant influence on the flow patterns, mass distributions, and pressure drops in either the symmetric model, or the obstructed models. They concluded that the inlet velocity distributions should be neither uniform, nor symmetric parabolic, but skewed-parabolic due to having been skewed by the upper carina ridges.

CFD studies based on subject specific medical imaging are important to accommodate differences in the branching upper airway and lobe geometries between individuals, leading to more accurate ventilation and particle transport. Researchers from The University of Iowa is pursuing the generation of a lung geometry atlas, Refs. [38], [39] and [11]. These studies focus on subject groups sharing particular anatomical features. These features will be related to airway damage via inhaled substances, or to accelerated damage to their functional tissue because of different stress/disease distributions. Once the atlas is completed, other individuals can be diagnosed based on the place in atlas even before the any diagnosis related to any diseases. Additionally, this
atlas should serve to provide validation data for techniques such as that advanced in this thesis.

In Ref. [46], Lin, Tawhai, McLennan and the Hoffmann from University of Iowa, recently worked on the effect of the mouth on the airflow patterns within the lung. They developed two geometries for the same individual: one includes intra-thoracic airways of up to six generations while the other includes mouthpiece, the mouth, the oropharynx, the larynx in addition to airways. Their CFD results showed that a curved sheet-like turbulent laryngeal jet appears in the geometry which starts from the mouth while the turbulence is negligible if the geometry starts from airways. They showed that this turbulence significantly affects airway flow patterns as well as wall shear stresses at the thrachea. Thus, airflow modeling, particularly subject specific evaluations, should consider upper as well as intra-thoracic airway geometry.

1.6 Previous Work: Particle Deposition

Leemhuis studied the particle deposition in the human lung in her M.S. thesis, [12], which is also supervised at Penn State University by the adviser of this thesis. Since first chapter of Leemhuis’ thesis presents the particle deposition studies in the literature, it will not be repeated here. A paper, Ref. [15], summarizes the results of the multiphase CFD analysis of particle transport and deposition research of this group. The geometry, the trachea and first five generations of bronchial tree, segmented from the CT scan of a human lung rubber cast, is meshed by unstructured hybrid meshes. The conservation equations are solved for multiple particulate fields at different characteristic sizes of
pharmaceutical interest. The carrier field is air. Steady state simulations are performed for nominal adult human inhalation. Computed streamlines for the air and particle fields at different sizes are presented. Cumulative deposition rates of particles at each generation and total deposition for the entire model for different particle diameters are also presented. Some of their results are discussed at Chapter 4.

1.7 Summary

The general goal of the present work is to contribute to the state of the art in the modeling of respiratory fluid dynamics. The specific goal of this work is to advance and automate the coupled use of modern medical imaging technology, image processing techniques, octree-based grid generation, and time-accurate, unstructured, parallel CFD technology, by developing a complete software package for a subject specific semi-automated modeling.

The approaches taken and contributions of the present work are: 1) the application of modern medical imaging technology and commercial image processing tools to segment lung images, within the AMIRA [23] software package, that are suitable for CFD analysis, 2) the development of grid generation schemes based on the octree method, within the HARPOON [29] software package, for discretization of the macro-scales, 3) the development of software written for quasi-one-dimensional (Q1D) geometric, grid generation and flow modeling approaches for the unresolved micro-scale convective regime, 4) the development of a multi-disciplinary, script-based simulation framework for macro-through-micro scale respiration modeling, 5) the assessment of
different boundary condition approaches for the pulsatile flows/scales of interest in respiration, and, 6) the application of the overall simulation scheme to study human respiration.

The ultimate goal of this research is the automation of the process from image processing through CFD analysis so that the physician will have additional information for making diagnostic, dosage or surgical option decisions.

1.8 Organization of the Thesis

Chapter 2 focuses on the geometric modeling elements of the research, including: image processing of CT scans, airway processing (segmentation, thinning, partitioning, truncation,) lobes segmentation. A brief description of the branching algorithm used to fill the lung lobes with artificial branches is presented. Chapter 3 discusses the grid generation. In Chapter 4, the theoretical formulation for the problem, including the governing equations, physical modeling, and numerical methods, is presented. In Chapter 5, the analytical solution for fully developed pulsating channel flow is obtained. This solution is used to validate the flow solver for pressure driven oscillating flows. In Chapter 6, flow through an entire lung airway geometry is solved. Final conclusions and recommendations for future advancements are outlined in Chapter 7. The appendices include software developed by the author in this research: a skeleton script for AMIRA, geometry tool GEO_LUNG, python script breath.py and compna.py for analytical numerical solution comparison for Chapter 5.
Chapter 2

LUNG GEOMETRIES

2.1 Rubber Cast Model

Two upper airway geometries have been used to develop and demonstrate the geometric manipulation tools used in this thesis. The first geometry was obtained as a triangulated surface (STL file) from Professor Andres Kriete of Drexel University (Dr. Kriete is a co-investigator in the NIH research grant that funded this work). This geometric model is not based on clinical imaging of a subject. Specifically, the existence of bones and tissue, motion artifacts and allowable radiation doses limit the resolution available in a clinical image of a live subject. Rather, this first model is based on a rubber cast of a deceased subject’s lung shown in Figure 2.1. From the rubber cast, Kriete’s colleagues produced a set of digital volume data using high-resolution computed tomography (HRCT) with a pixel dimension of 0.35 mm × 0.35 mm, and with 0.4 mm distance between slices. They segmented the resulting volume data set to obtain the representation of the trachea and the connected bronchial tubes. The final surface representation of the upper airways goes up to 12th-13th generation. The topological information defining branch connectivities, diameters and branching angles of 1453 bronchi as well as asymmetric and multifractal characterization of the lung were provided as well [22]. The photograph of rubber cast model with overlaid CFD is given in Figure
2.1. This extensive upper airway geometry, shown in Figure 2.2, was used to challenge and verify the geometric partitioning and truncation algorithms developed in this work. It has also been used for direct CFD applications of the first 12-13 generations (i.e. without coupling to the quasi-one-dimensional lower-bronchi approach described below). Figure 2.3 shows the skeleton of rubber cast model.

Figure 2.1: Photograph of the rubber cast model with overlaid CFD
Figure 2.2: View of STL file of rubber cast model.

Figure 2.3: Skeleton of rubber cast model based on topological information.
2.2 Image Processing from Computed Tomography

The second, lower resolution, upper airway geometry was obtained from clinically obtained computed tomography (CT) scans of an elderly patient. These images were processed using the commercial image processing software AMIRA [21].

The CT data for each patient, as provided to us by Dr. Kriete, consists of 3-D image data, stored according to sgi TIFF standard, compiled with a DICOM header of the clinical image slices. AMIRA is incapable of reading this 3-D image format. A free software package, Libtiff was used to read the 3-D image data and rewrite it as a sequence of 2-D image slices in bmp format. The DICOM header files, which are read by the free software MEVISLAB [2], provides voxel information to AMIRA, stores clinical information about the patient as well as technical information about the CT images.

Figure 2.4 illustrates the MEVISLAB interface for the CT data considered here. Figure 2.5 illustrates the MEVISLAB interface for this data.
Figure 2.4: 3-D view of a CT scans in MEVISLAB

Figure 2.5: Dicom header file in MEVISLAB
2.2.1 Segmentation of Airways

Segmentation is the name of the process of identification of regional boundaries. Automated segmentation of airways is an important step in end-to-end clinical image through CFD analysis process. Many papers in the literature discuss the region growing technique for automated segmentation, such as [17], [18], [19], [38]. However, the leakages in between black-white borders of tissue and air make the robust fully automated segmentation by region growing technique virtually impossible.

Lung mask can be defined as the initial segmentation of the lung from the 3-D medical images, a region of interest for the airway segmentation and lobe segmentation processes. Pool is the name of object oriented coding window of AMIRA; top right window in Figure 2.7 and 2.10.

The following steps describes the application of the region growing technique to CT images to obtain the raw data of segmented airways covered by lung mask by using AMIRA. The total process takes several minutes on a two-processor, 2 GHz, 4 GB RAM desktop PC.

i) Image data is imported as stacks of numbered 2D images. The coordinates of the lower left front corner are set as (0, 0, 0) and the accurate voxel size is entered by the user.

ii) From the Labeling module, LabelVoxel module is chosen. This module provides a simple threshold segmentation algorithm applicable to CT or MR image data. Up to five different regions separated by four different thresholds can be extracted. For lung tissue, Exterior-Interior is set with
a threshold value of 40. In a few seconds, this will separate the air from the tissue illustrated in Figure 2.6.

Figure 2.6: Automatically segmented image by AMIRA.

iii) The next step is surface generation. The SurfaceGen module computes a triangular approximation of the interfaces between different tissue types in a LabelField with either uniform or stacked coordinates. The option of unconstrained smoothing, add border and adjust coords are chosen. Unconstrained smoothing and constrained smoothing generate sub-voxel weights, such that the surface is naturally smooth. Constrained smoothing guarantees that no label be modified. The default amount of smoothing is 5 for the unconstrained and 4 for the constrained case. add border, ensures that the resulting surfaces will be closed. Depending on the resolution of the LabelField the resulting triangular surface may
have a huge amount of triangles. If this creates a problem in computer memory usage, the label field can be resampled before the surface generation. The Resample module will create or update the probability information of the LabelField. This way the loss of information caused by the resampling process will be minimized. In few minutes, a closed surface of the interior will be generated.

iv) For the lung CT used here, the resulting surface has more than 2 million triangles. Therefore it needs to be simplified before it is saved. In the Simplifier module, an appropriate number of faces is set, like 300,000, in this case and the Simplifier is run. If the number of faces set is too small, then the resulting surface will lose its accuracy. This process takes few minutes.

v) The resulting triangular surface can be visualized using the SurfaceView module. If Draw Style is set to transparent, then it is possible to see the airway segmentation behind the outer mask of the lung.

vi) The resulting surface is saved in an ascii STL format by using save data from file menu.

The resulting surface actually includes many “islands” and needs to be processed to obtain the proper closed upper airway geometry for CFD application. In this work, the STL file is opened in a commercial automated grid generator HARPOON, [29]. HARPOON can automatically split the geometry if the separate closed surfaces are represented within a single STL file. By using the option Geometry/Separate/by Region, islands within the lung mask are separated and then deleted, resulting in the geometry
illustrated in Figure 2.7. The final step is removing the triangles which do not belong to airways but the outer surface, as illustrated in Figure 2.8. Once the outer peel is removed, the leakage is so clearly seen in Figure 2.9. The region growing technique applied on CT image is not giving clearly segmented airway geometry.

Figure 2.7: Pool for automated segmentation of lung mask in AMIRA.

Figure 2.8: Lung mask after removing of islands in HARPOON.
Contrary to the region growing segmentation process in AMIRA described above, the classical manual segmentation methods are highly time consuming. However, considering the quality of CT scans that have been used in this thesis, this is the only way to segment the upper airways. By using the tools in segmentation tool bar, the upper airways are segmented up to generation 4, 5 or 6. By applying the Remove Islands option in the Segmentation menu cleans the small islands, Figure 2.10 (since this geometry is smaller than the lung mask geometry described above, AMIRA can remove the islands).

Figure 2.9: Showing of leakage in airway geometry in automated segmentation.
Again, the same procedure is used to generate a triangular representation of the closed, smoothed and simplified surface. For upper airways up to 6th generations, the number of faces could be simplified to 10000. The resulting upper airways geometry is saved in ASCII STL format, Figure 2.11.

Figure 2.10: Segmentation tool bar and removing islands option in AMIRA.
2.2.2 Thinning

The thinning is an iterative layer by layer erosion until only the skeleton of the closed upper airway geometry is left. The skeleton of the upper airway geometry is necessary for assigning generation numbers, partitioning and truncation processes. For automated thinning, an AMIRA script was written. This script is shown in Figure 2.12. AMIRA has a built-in Tcl interface and it can be scripted. Once the script file is written, it is loaded into the pool like the any other AMIRA module as illustrated in Figure 2.12. The running time of skeleton script is approximately one minute (on a two-processor, 2 GHz, 4 GB RAM desktop PC) for upper airways up to 6th generation.
The following modules are used in the thinning script, [23]:

Mosaic module is the container of spatial data objects. It stores a reference to the location on file and the bounding box information. MosaicToLargeDiskData module takes a Mosaic containing bricks of overlapping image data and converts them to one LargeDiskData object stored on disk. Trilinear standard interpolation is chosen. Thinner module takes labels that have been stored as LargeDiskData, and a distance map stored as LargeDiskData as input. It runs a thinning procedure on these to extract the center lines of the structure contained in the labels. DistanceMap computes a 3D distance field of a 3D object. Each voxel is assigned a value depending on the distance to the nearest object boundary. The boundary voxels of the object are assigned a value of zero whereas the assigned value increases as the distance increases. TraceLines module takes a LargeDiskData image as input. This image may be the result of the Thinner module. The image should contain only lines represented by voxels. The module traces these lines and builds a line set and/or a cluster out of it. Smooth makes a cleanup on the skeleton and

Figure 2.12: Script to obtain the skeleton of upper airways.
smoothes it. EvalOnLines module takes a LineSet and a LargeDiskData field as input. The field is evaluated at each vertex of the lineset and the result stored in the lineset.

The resulting skeleton line set is showing in Figures 2.13 and 2.14. This data is saved as a text file specified in the script file. This file includes the x,y,z coordinates of the points of each line set and the corresponding local diameters.

Figure 2.13: Skeleton of upper airways, curved lines.
2.2.3 Segmentation of Lung Lobes

The fissures separating lung lobes can be more visible if a light color map is used in AMIRA. By following these features/curves within 2-D transverse slices, and by saving the smoothness of the anatomical curves, lobes are manually segmented, i.e., the surface of each lobe is evolved and exported as a standard STL file after smoothing.
removing islands, filling holes and simplifying the surfaces. The resulting five lobe segmentations for the present subject are showing in Figure 2.15.

Figure 2.15:  a) Right upper (green), middle (blue) and lower lobe (pink) segmentations  
b) Left upper (yellow), and lower (orange) lobe segmentations.
2.3 Processing of Medical Imaging Geometry for CFD

2.3.1 Partitioning

An in-house code (written in C), GEO_LUNG was developed by the author to truncate the airway trees. The source code listing for this software is included in Appendix A. The lung airway tree is a very complicated geometry for mesh generation. Partitioning is needed to assign different grid generation attributes (e.g., cell size, number of prism layers) to different branches/bronchioles, with the same grid attributes typically assigned to branches that have the same generation number. Manually partitioning would be very time consuming for a geometry with numerous branches, so by automating partitioning, significant time is saved. Partitioning also allows the truncation of the airway tree at a desired location, a process described in the next section. The following algorithmic steps were developed and coded within GEO_LUNG to partition an airway tree:

- The surface geometry definition is input as an STL file: normals and vertex coordinates are stored.
- The centers of the surface triangles are calculated as a reference point for each triangle.
- The skeleton file is input and each branch segment is defined as “edge”. The starting and ending points of each edge, (bifurcation points) are defined as “nodes”. Nodes and edge numberings are stored. The first node of the branch is that which connects the new edge to the parent branch, and the second node
connects the edge to the daughter branches of that edge/branch, as shown in, Figure 2.16.

Figure 2.16: Edge (branch) and node (bifurcation) numbering.

- The diameter of each edge is stored. If instead of diameters, the length and volume data is provided in the topological file, then the diameters are calculated from these two data.
- The surface triangles are grouped as trachea, left lung and right lung to reduce the time needed in search loops.
- Unit normals of planes perpendicular to each edge are defined. Normals are in the direction from second node to first node.
• For each branch, a plane which is perpendicular to the edge at the first node is defined. The equation of a plane with a normal \((n_x, n_y, n_z)\) and containing point \((x, y, z)\) is as follows:

\[
n_x x + n_y y + n_z z + d = 0
\]  

(2.1)

In Hessian normal form, the distance between a point with coordinates of \((x_0, y_0, z_0)\) and a plane with a unit normal of \((n_x, n_y, n_z)\) is:

\[
D = n_x x_0 + n_y y_0 + n_z z_0 + d
\]  

(2.2)

Where \(d\) can be calculated from Eq. 2.1. If distance \(D\) is negative, the point is above the plane.

• To match the triangles with the branches (edges), a search loop is set for each triangle. The distance \(D\) given in the Eq. 2.2 is calculated between the center of the triangle and the top normal plane. If the triangle is between the top and bottom normal planes of an edge, then this edge is a candidate for the triangle’s assignment.

• The next criterion to look at is the minimum distance between the center of the triangle and the candidate edges. The smallest distance determines the correct edge. Once the correct edge is found, the generation number belonging to this edge is assigned to the triangle.

Figure 2.17 illustrates the application of the foregoing partitioning algorithm to the rubber cast model. This model challenges the algorithm significantly since there are
1453 branches in the model. Each color in the figure corresponds to a different generation number.

2.3.2 Truncation

GEO_LUNG also truncates the airway trees at specified locations and assigns the boundary conditions at these airway truncation points a process required for fully-resolved 3-D CFD calculations of the upper lung. Specifically, for fluid flow calculations, the truncated trachea and the truncated “leaves” (bronchioles in the tree with no resolved daughters in a given segmentation) will be the pressure boundaries that drive the flow. The triangles defining these surfaces must be unambiguously distinguished from bronchi walls, and oriented normal to the flow, to assign the pressure boundary conditions. The truncation process in GEO_LUNG defines these boundaries automatically without any need of hand work.

Figure 2.17: Partitioning of airways by GEO_LUNG.
Truncation is based on the logic of intersection of a triangle with a plane in 3-D space. For the trachea intersection, surface triangles above the cutting plane are removed, (dashed lines in Figure 2.18). Those which intersect the cutting plane are re-created as also shown in Figure 2.18. The open tip of the trachea is closed with new surface triangles which have the same normal with the cutting plane. These new triangles define the inlet boundary condition. For the outlets, surface triangles below the cutting plane are removed and the same procedure is applied as shown in Figure 2.19. Note that the aspect ratio of these closure triangles can be quite large. In the present application, this is acceptable since they are exclusively used to define a closed, flat “cap” to the bronchiole which is interpreted as such by the 3D volume grid generator used for the CFD analysis. This is discussed further in Chapter 3.

Figure 2.18: Schematic representation of the truncation of surface triangles for inlet.
Figure 2.19: Truncated outlets closed with new surfaces.

Figure 2.20 shows elements of the partitioned and truncated rubber cast model.
Figure 2.20: Truncation of the rubber cast model.
2.4 Branching Algorithm / Filling the Lobes with Artificial Bronchioles

The final component of the lung geometric model is the statistical representation of the lower airways in the convective regime. A geometric description of each lobe and the skeleton corresponding to the resolved upper branches are used as input to a Fortran 90 code LBRANCH developed by Professor Daniel Haworth of The Pennsylvania State University (Dr. Haworth is a co-investigator in the NIH research grant that funded this work). The function of code LBRANCH is to fill in the geometric structure of the conducting airways from the truncated ends of the resolved upper branches down to the respiratory units.

A separate geometry input file is provided for each of the five lobes. Each lobe file contains a coarse tetrahedral volume mesh ($O(10^4)$ tetrahedral elements) that is generated from the triangulated surface mesh for that lobe. These surface meshes are obtained from AMIRA, shown in Figure 2.21. The upper-branches input skeleton file, shown in Figure 2.13 is also obtained from AMIRA. With current clinical resolution, there are $O(10^5)$ terminal branches in the input skeleton. From each terminal branch a subtree is propagated down to the respiratory units using a volume-filling statistical branching algorithm. The algorithm is scalable so that as more of the upper airways are resolved, more subtrees (each containing fewer branches) will be generated as necessary.

The branching algorithm reconstructs subtrees of the global tracheobronchial tree. The “root” of each subtree is a terminal branch of the input skeleton, and the “leaves” of each subtree correspond to respiratory units. At least one terminal branch feeds each lobe; the actual number increases with increasing resolution in the input skeleton. A volume-
filling algorithm has been devised such that the resulting spatial distribution of respiratory units is statistically uniform within each lobe. The method is similar to previously published lung branching algorithms, notably those of Tawhai et al. [42] and Kriete et al. [22]. Key branching model input parameters are an average respiratory unit volume (typically \( \sim 100 \text{ mm}^3 \)), an upper limit on parent-to-child branching angle (~60°), lower and upper limits on branch length (~1 mm, ~20 mm), and a branching fraction as defined in [42] (~0.4). A diameter is assigned to each branch based on its Weibel generation number [43]. A typical resulting global tracheobronchial tree (the union of the input skeleton and all subtrees) represents 22-23 generations of branching and contains 60,000-80,000 branches of which approximately half are terminal branches (respiratory units). Statistics of branching angles, branch lengths, branch ratios, and average number of generations from the trachea to the respiratory units are similar to those reported in [42], although the trees generated here tend to be somewhat more symmetric.

The output of the branching module is the global tracheobronchial tree. This consists of a list of bifurcation points, a list of branches, and connectivity information. In addition, each branch is assigned a diameter (as described above) and one or more generation numbers (e.g., Weibel generation number and Horsfield order) for postprocessing purposes. In Weibel generation numbering, branches are counted from the trachea. Each child branch is one order higher than its parent. On the other hand, Horsfield ordering classifies the last conducting airways (or first transitional bronchioles) as order 1, and each parent is one order higher than the child branch of highest order, moving back toward the trachea.
Typical tracheobronchial trees represent on the order of 17-19 generations of branching, and contain between 70,000 and 100,000 branches, approximately half of which are terminal branches that terminate in a respiratory unit. Figure 2.22 shows the reconstruction for the present subject.
Figure 2.22: Artificially generated bronchioles (after 5\textsuperscript{th} generation).
Chapter 3

GRID GENERATION FOR LUNG GEOMETRIES

3.1 Introduction

Gridding is the process of subdividing a region to be modeled into a set of small control volumes. For cell centered finite volume CFD schemes, as used in this thesis, each control volume is associated with the locally averaged values of the dependent flow variables, in this case, velocity, pressure, turbulent kinetic energy and turbulence dissipation rate.

The lung geometry introduced in the previous chapter is extremely complex. Accordingly, some grid generation approaches can be difficult if not near impossible to apply for this application. In this chapter, standard techniques for grid generation are summarized, and an automated unstructured technique is settled upon as having suitable geometric flexibility, and suitable ability to control cell size to return locally high resolution meshes. A unique approach for grid generation for the quasi-one-dimensional lower airway simulations is presented as well.
3.1.1 Structured Meshing

In structured meshing, the computational domain is subdivided into hexahedral elements that are ordered in a logically rectilinear fashion, with the connectivity between elements defined trivially by constant strides through the data. Structured meshes remain in widespread use for CFD due to the inherent structure efficiencies that can be exploited by the flow solver, and the high quality meshes that can be constructed (especially for high Reynolds number boundary layers.) Also, the now-widely-used extensions of multi-block and overset structured meshing, extend the application space of structured meshing to more complex geometries. However, there are some geometries that are so complex that construction of a good quality structured mesh would require an inordinate amount of user interaction, and in some cases be nearly impossible to obtain. Underhood engine compartment flows and some biological system flows are examples of such geometries.

3.1.2 Unstructured Grid Generation

In unstructured meshing, the computational domain is subdivided into arbitrarily shaped elements in an unordered fashion; any number of elements can meet at a single node and the connectivity between elements is no longer defined trivially. This generality affords the flexibility to mesh very complex geometries, and for this reason is employed for lung modeling in this work.

Tetrahedral grids, hybrid grids (composed of simplicial elements: tetrahedral, hexahedra, prisms, pyramids), “arbitrary polyhedral” meshes generated from the mesh dual of simplicial meshes, and octree based grids and are the most common unstructured
forms used in CFD today. Pure tetrahedral meshes are efficient to construct, but are associated with loss in numerical accuracy and cell count increase in regions with large gradients (i.e., boundary layers). This has led to approaches/software that employs prisms extruded from the surface triangulation. Such hybrid element approaches often lead to the existence of pyramids in regions where tetrahedral and prisms interface, and require CFD solver support for mixed element types. Such a hybrid approach is that used in earlier lung simulation work by the present Penn State/Drexel group, [12], [14]. Since the early 2000’s, arbitrary polyhedral meshes obtained from mesh duals of more conventional topologies have gained some notice. Such meshes can in principle improve the accuracy of a CFD simulation for a given cell/face count (i.e., net CPU time) since gradient computations have more compact support. However, grid quality can easily be worse than the generator mesh, adaptation is more complex and many CFD solvers do not support such meshes.

Of principal interest in this thesis is the octree technique. Octrees are recursive data structures where each tree-node may have eight geometrically-similar children. With this technique, the geometric model is inserted into a cube. Then cubes are recursively subdivided until the desired resolution is reached. Irregular cells are created where cubes intersect the surface, so surface intersection calculations are required. The resulting mesh is composed largely of hexahedra (indeed usually cubes), but the subdivision process produces unstructured adjacencies and elements that are non-simplicial. Specifically, general polyhedral elements can be generated in the internal mesh subdivision process and, adjacent to boundaries, in the surface intersection process. Cartesian elements with planar faces, many with “hanging nodes”, are the common element type in octree based
A hanging node occurs because a cell is subdivided while one of its neighbors is not, as shown in Figure 3.1. Accordingly, CFD solvers that employ octree meshes require arbitrary polyhedral support as well. Unlike the other techniques, the octree technique does not initialize from a surface mesh, but surface facets are formed when the internal octree structure intersects the surface boundary.

The principal advantages to octree approaches is that they execute very rapidly, and can be automated, even for very complex geometries, saving many hours of analyst time for each configuration, and rendering trivial the analysis of similar but non identical configurations (e.g., lungs of various patients). This ability to automate comes at the cost of increased cell/face count (i.e., net CPU time) for a given level of numerical accuracy. This can be somewhat compensated for by controlling the rate of mesh reduction, but care must be taken in maintaining accuracy especially within hanging node elements.

3.1.3 Gridding for the Lung Geometry

In this thesis, an octree unstructured method is employed. Specifically, a commercial software package, HARPOON is used for gridding of the upper branches. An in-house code described in Section 3.3 is used for lower branch gridding. The process is fully automated and gives rise to arbitrary polyhedral elements of several types as described below. As described in Chapter 4, the CFD solver used in this thesis, NPHASE-PSU solver supports such elements.
3.2 Octree Based Gridding of the Upper Airways

HARPOON is an octree-based fully automatic grid generator. It has a graphical user interface and it also works in batch mode. It is user friendly and very fast compared to unstructured grid generation software based on other techniques. By virtue of its ability to automatically generate grids of good quality for the upper airways, we have settled on this approach for the 3D gridding component of the overall end-to-end respiratory analysis package.

The hexahedral-dominant meshes characteristic of octree methods also locally return tetrahedras, wedges and pyramids adjacent to boundary intersections. Hanging nodes are generated at octree subdivision boundaries in as illustrated in Figure 3.1. A prism layer can be constructed along wall surfaces by applying HARPOON’s boundary layer option.

Figure 3.1: HARPOON mesh with hanging nodes
HARPOON uses different size hexas (octree levels) to control the detail of the final mesh. The base level 1 corresponds to the coarsest meshing in the model (center of largest bronchioles in the case of the lung). As the level increases, the size of the hexas decrease by a factor of two, that is the elements at level 2 are the half size of elements at level 1. A specific base cell size may be typed in to get the exact size required. Users assign different levels to different parts of the geometry. The user typically executes HARPOON interactively the first time a new class of geometry is considered. The surface geometry is input, all gridding attributes are assigned, the grid is generated and conditioned (e.g., repair, smoothing, prism layers) if necessary or desired, and then output to an appropriate format. These interactive steps are saved in an ASCII “configuration” file which can be modified by hand and executed later in batch mode. Figure 3.2 shows the configuration file used for the present upper airway gridding.

The imported .stl file for upper branches is generated by GEO_LUNG as described in Chapter 2. This file has been partitioned, truncated and had the wall and truncated pressure boundary names assigned. Regardless of the number of generations and pressure boundaries in the geometry, HARPOON recognizes the generation-numbered walls, the trachea and pressure boundaries as different parts (attributes inherited from the STL file produced by GEO_LUNG). The configuration file, HARPOON assigns the grid attributes to these different parts (e.g., smaller levels/cell-size for higher generations) and generates the mesh.

HARPOON exports the mesh in a face-based “Cobalt” format which is a format accepted by the front end to NPHASE-PSU, the CFD solver used in this thesis. GEO_LUNG outputs an additional file, cobalt.bc, that is used by the NPHASE-PSU front
end to assign flow-solver naming conventions to the numerical boundary specification attributes output by HARPOON.

HARPOON cannot create an internal mesh if the input geometry is not defined with an entirely closed surface. Despite the technology described in chapter 2, to prevent such occurrences, the STL geometry file output by GEO_LUNG may have small holes. The geometry must be repaired before mesh is generated. Therefore, at the beginning of the HARPOON configuration file, batch commands “findgeomhole” and “fillgeomhole” appear hole filling the surface geometry.

As mentioned above, HARPOON’s batch capability and its feature of assigning different mesh level to different parts of the geometry are two enablers for rapid automated human respiratory system modeling. Another important feature of HARPOON in this context is its speed. Although memory intensive for desktop PC applications (approximately 150MB per million cells), a one-million-cell-mesh can be generated in less than 60 seconds on a two-processor 2 GHz, 4GB RAM desktop PC.

Figures 3.3 and 3.4 show views of an automatically generated HARPOON mesh for the rubber cast upper airways geometry.

The aforementioned hexahedral dominant mesh method is used for the upper airways. The volume filling algorithm described in Chapter 2 requires a volume grid for the five lobes. The author supplies this mesh by using HARPOON as well. Specifically, the lobe STL files that are generated by AMIRA are input to HARPOON and a fairly coarse volume mesh is generated, again automatically. This mesh is not used for anything except volume containment in the lobe filling algorithm, so HARPOON’s full-tetrahedral meshing option is used. The lobe meshes are exported in abacus format for input to the
lobe filling code, LBRANCH. Figure 3.5 shows a view of the automatically generated HARPOON mesh for the Erna lobe geometry.

<table>
<thead>
<tr>
<th>Import</th>
<th>stl</th>
<th>plevel</th>
<th>pressure</th>
</tr>
</thead>
<tbody>
<tr>
<td>F:\a6_TUBE\ONEMLI\DOSYALAR\hexdom_wo_hgn\mart_red_per85fill.stl</td>
<td></td>
<td>plevel 25 -1 -1 -0</td>
<td>PRESSURE_18</td>
</tr>
<tr>
<td><strong>VERSION v2.4(c)</strong></td>
<td></td>
<td>plevel 26 -1 -1 -0</td>
<td>PRESSURE_19</td>
</tr>
<tr>
<td><strong>PREFERENCES USED</strong></td>
<td></td>
<td>plevel 27 -1 -1 -0</td>
<td>PRESSURE_20</td>
</tr>
<tr>
<td><strong>Max Skew 0.999500</strong></td>
<td></td>
<td>plevel 28 -1 -1 -0</td>
<td>PRESSURE_21</td>
</tr>
<tr>
<td><strong>Target Skew 0.980000</strong></td>
<td></td>
<td>plevel 29 -1 -1 -0</td>
<td>PRESSURE_22</td>
</tr>
<tr>
<td><strong>Max Face Warpage 40.000000</strong></td>
<td></td>
<td>plevel 30 -1 -1 -0</td>
<td>PRESSURE_23</td>
</tr>
<tr>
<td><strong>noreset</strong></td>
<td></td>
<td>plevel 31 -1 -1 -0</td>
<td>PRESSURE_24</td>
</tr>
<tr>
<td><strong>intersect</strong></td>
<td></td>
<td>plevel 32 -1 -1 -0</td>
<td>PRESSURE_25</td>
</tr>
<tr>
<td><strong>Separation Angle 40.0</strong></td>
<td></td>
<td>plevel 33 -1 -1 -0</td>
<td>PRESSURE_26</td>
</tr>
<tr>
<td><strong>Setting No. of Cells Between walls to 3</strong></td>
<td></td>
<td>plevel 34 -1 -1 -0</td>
<td>PRESSURE_27</td>
</tr>
<tr>
<td><strong>Setting BDF Exports to Short Format</strong></td>
<td></td>
<td>plevel 35 -1 -1 -0</td>
<td>PRESSURE_28</td>
</tr>
<tr>
<td><strong>Setting BDF Pyramid Treatment to use degenerate PENTA elements</strong></td>
<td></td>
<td>plevel 36 -1 -1 -0</td>
<td>PRESSURE_29</td>
</tr>
<tr>
<td><strong>Setting Max No. Separate Volumes to 10</strong></td>
<td></td>
<td>plevel 37 -1 -1 -0</td>
<td>PRESSURE_30</td>
</tr>
<tr>
<td><strong>Setting Part Description to use STL name</strong></td>
<td></td>
<td>plevel 38 -1 -1 -0</td>
<td>PRESSURE_31</td>
</tr>
<tr>
<td><strong>Setting Fluent Thin Wall Treatment to Double Sided</strong></td>
<td>plevel 39 -1 -1 -0</td>
<td>PRESSURE_32</td>
<td></td>
</tr>
<tr>
<td>farfield global</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>farfield xmin -87.452838</td>
<td></td>
<td>level 1 -1</td>
<td></td>
</tr>
<tr>
<td>farfield ymin -281.630183</td>
<td></td>
<td>level 2 -1</td>
<td></td>
</tr>
<tr>
<td>farfield zmin -632.163643</td>
<td></td>
<td>level 3 -1</td>
<td></td>
</tr>
<tr>
<td>farfield xmax 393.246401</td>
<td></td>
<td>level 4 -1</td>
<td></td>
</tr>
<tr>
<td>farfield ymax 199.069056</td>
<td></td>
<td>level 5 -1</td>
<td></td>
</tr>
<tr>
<td>farfield zmax -151.464422</td>
<td></td>
<td>level 6 -1</td>
<td></td>
</tr>
<tr>
<td>findgeomhole</td>
<td></td>
<td>level 7 -1</td>
<td></td>
</tr>
<tr>
<td>fillgeomhole</td>
<td></td>
<td>level 8 -1</td>
<td></td>
</tr>
<tr>
<td><strong>MESH METHODS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>type hex</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>expand slow</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mesh internal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>remove</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SINGLE LEVEL</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>level 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gminlev 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gmaxlev 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 1 2 2 0  'WALL_00'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 2 2 2 0  'WALL_01'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 3 2 2 0  'WALL_02'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 4 3 3 0  'WALL_03'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 5 3 3 0  'WALL_04'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 6 4 4 0  'WALL_05'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 7 -1 -1 -0  'PRESSURE_00'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 8 -1 -1 -0  'PRESSURE_01'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 9 -1 -1 -0  'PRESSURE_02'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 10 -1 -1 -0  'PRESSURE_03'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 11 -1 -1 -0  'PRESSURE_04'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 12 -1 -1 -0  'PRESSURE_05'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 13 -1 -1 -0  'PRESSURE_06'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 14 -1 -1 -0  'PRESSURE_07'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 15 -1 -1 -0  'PRESSURE_08'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 16 -1 -1 -0  'PRESSURE_09'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 17 -1 -1 -0  'PRESSURE_10'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 18 -1 -1 -0  'PRESSURE_11'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 19 -1 -1 -0  'PRESSURE_12'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 20 -1 -1 -0  'PRESSURE_13'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 21 -1 -1 -0  'PRESSURE_14'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 22 -1 -1 -0  'PRESSURE_15'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 23 -1 -1 -0  'PRESSURE_16'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>plevel 24 -1 -1 -0  'PRESSURE_17'</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.2: HARPOON’s configuration file for upper airways.
Figure 3.3: Octree based HARPOON upper airway grid. Surface geometry with six different parts and 32 outlets.
Figure 3.4: Octree based HARPOON upper airway grid. a) Cut plane xz. b) Cut plane xy. c) Cut plane yz.
3.3 Quasi-One-Dimensional Gridding of the Lower Branches

The upper airways constitute on the order of $10^3$-$10^4$ bronchioles depending on the resolution of the medical image and quality of the segmentation. As described in Chapter 2, however, the lower airways, i.e. those that extend from the resolved upper airways through the convective regime, constitute an additional order of $10^4$-$10^5$ bronchioles. The length, connectivity and diameter attributes of these lower airways are modeled using LBRANCH, summarized in chapter 2. In order to incorporate the flow through these lower branches in the CFD analysis of the entire lung, a computational mesh is constructed for these branches as well.
As described in Chapter 4, a quasi-one-dimensional CFD model is used for the lower airways. The approach taken is to use NPHASE-PSU to model the complete lung, using conventional 3D numerics for the upper airways, and simplified Q1D elements and modeling for the lower branches. In order to interface these two scales, a Fortran 90 code, TREE, was written by Dr. Robert Kunz (Dr. Kunz is a co-investigator in the NIH research grant that funded this work, as well as the author’s thesis adviser). TREE takes the output subtrees from LBRANCH and builds a pseudo-cylinder Q1D mesh for each branch, defined by a user specified $ns$ axial elements, as shown in Figure 3.6. Each of the $ns$ elements is itself an $nc+2$ sided polyhedron, where $nc$ is the user specified number of circumferential edges that define the pseudo-circular cross section of the branch.

![Figure 3.6: Q1D grid for a typical subtree section, ns=4, nc=12. Wall-less bifurcation element indicated by red circle.](image)

TREE builds the elements for each branch and defines the boundary pointers for each element. Each element has $nc$ quadrilateral wall surface faces and two $nc$ sided faces that are either internal faces that point to/from the cell to adjacent cells, or are pressure
boundary faces (last cell in a terminal bronchi). In addition, there are “wall-less” bifurcation elements defined between all parent and daughter (usually two, sometimes only one) bifurcations. Each of these bifurcation elements is bounded by two or three n-sided faces that are internal faces that point to/from the cells at the parent outlet and daughter inlet cells, as shown in Figure 3.6.

3.4 Interfacing the Upper and Lower Branch Grids

TREE also reads in the cobalt file output by HARPOON and appends the grid and connectivity information for the upper airways with the newly constructed lower airway mesh. A composite cobalt file is output by TREE that contains the grid and connectivity for the entire lung, ready for the CFD solver front-end. The elements that interface the terminal leaves from the upper airways and the top branches of each of the subtrees are also “wall-less” elements as illustrated in Figure 3.7.

Figure 3.7: Wall-less bifurcation element interfacing upper and lower airways. Element indicated by red circle.
The Figure 3.8 shows a view of the complete lung grid for the subject under consideration.

Figure 3.8: View of a complete lung grid. 207,903 elements in upper airways, 754,492 elements in lower airways.
4.1 Governing Equations

An ensemble-averaged single-pressure, n-fluid differential system is employed to solve the separate governing equations for particle/droplet and gas components. The averaged continuity and momentum equations in conservation law form are as follows:

\[
\frac{\partial \alpha_i^k \rho^k}{\partial t} + \frac{\partial \alpha_i^k \rho^k u_j^k}{\partial x_j} = \sum_{k \neq i} \left( \Gamma^{ik} - \Gamma^{ij} \right)
\]

\[
\frac{\partial \alpha_i^k \rho^k u_j^k}{\partial t} + \frac{\partial \alpha_i^k \rho^k u_j^k u_j^k}{\partial x_j} = -\alpha_i^k \frac{\partial \rho}{\partial x_i} + \frac{\partial}{\partial x_j} \left[ \alpha_i^k \mu_i \left( \frac{\partial u_j^k}{\partial x_j} + \frac{\partial u_j^k}{\partial x_i} \right) \right] + \rho_i^k g_i + \sum_{k \neq i} \left(D^{ik}[u_j^i - u_j^k] + \Gamma^{ik} u_j^i - \Gamma^{ij} u_j^k \right)
\]

(4.1)

The superscript \( k \) is the constituent or “field” designator. For mass transfer (\( \Gamma^{ij} \)) and drag (\( D^{ik} \)) and non-drag (\( M_i^{kj} \)) interfacial forces, superscripts \( k \) and \( l \) designate donor and receptor fields. In general each field, \( k \), will have a different density, volume fraction, velocity and viscosity. For single phase flow, these equations reduce to:

\[
\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_j}{\partial x_j} = 0
\]

\[
\frac{\partial \rho u_j}{\partial t} + \frac{\partial \rho u_j u_j}{\partial x_j} = -\frac{\partial \rho}{\partial x_j} + \frac{\partial}{\partial x_i} \left[ \mu_i \left( \frac{\partial u_j}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right] + \rho g_i
\]

(4.2)
4.2 Numerics

The CFD solver used in this thesis is the NPHASE-PSU code developed by Kunz et al. (2001), [14], [24], [33]. The code is unstructured, parallel and time-accurate. 2nd order viscous, 1st through 3rd order convection and 1st and 2nd order temporal discretizations are available. For single-phase flow, the algorithm follows established segregated pressure-based methodology. A collocated cell centered variable arrangement is used and a lagged coefficient linearization is applied (Clift 1994, for example, [34]). Continuity is introduced through a pressure correction equation, based on the SIMPLE-C algorithm (Van Doormal 1984,[35]). In constructing cell face fluxes, a momentum interpolation scheme (Rhie and Chow 1983, [36]) is employed, which introduces damping in the continuity equation. The discrete momentum equations are solved approximately every iteration, followed by a more exact solution of the pressure correction equation. Turbulence scalar and volume fraction equations are then solved in succession.

4.3 Data Structure

The hierarchal data structure employed is illustrated in Figure 4.1. The cell-centered finite volume flow solver accepts arbitrary polyhedral elements. The data structure is face-based, that is, subsequent to the assembly of geometric parameters in the front end, all inter-element connectivity is retained in face pointers to the two adjacent cells. This face-based data structure supports arbitrary polyhedra (as required for lung
The fundamental data structure member is the “fedge” (face edge) which points to nodes (vertices) and faces.

Figure 4.1: Hierarchal data structure in NPHASE-PSU

4.4 Discretization

The governing equations are discretized using a face-based, cell-centered, finite volume method applied to arbitrary polyhedral cell types. Inviscid and viscous fluxes are accumulated by sweeping through internal and boundary faces. For inviscid flux evaluation:

$$\int_A \rho \phi \vec{V} \cdot d\vec{A} \approx \sum_f C_f \phi_f$$  \hspace{1cm} (4.3)
Where \( C_f \) is face mass flux for field \( k \), and \( \phi \) is the value of general transport scalar \( \phi \) evaluated at face \( f \). The summation is taken over all faces bounding the element. \( C_f \) is evaluated based on field variables available prior to the solution of the transport equation for \( \phi \) (lagged coefficient linearization). Second-order accuracy is obtained by evaluating \( C_f \) using a central plus 4\(^{th}\) difference pressure artificial dissipation term due to Rhie and Chow (1983), [36].

\[
C_f = \rho_f \nabla \cdot \vec{A}_f + \rho_f \left[ \nabla \cdot \left( \vec{A}_f \cdot \nabla \phi - \Delta p \cdot |\vec{A}_f|^2 \right) \right] \quad (4.4)
\]

and by evaluating \( \phi_f \) from Lien (2000), [37]:

\[
\phi_f = \phi_{u} + \left( \nabla \phi \cdot d r_{u} \right) \quad (4.5)
\]

In Eq. 4.4, the overbar denotes a geometrically weighted mean at the face, i.e., referring to Figure 4.2.

---

Figure 4.2: Geometry nomenclature for cell face evaluations

---

\[
\overrightarrow{\nabla p_f} = (1-s)(\nabla p_1) + s(\nabla p_2) \quad (4.6)
\]

\[
s = \frac{\delta s_1}{(\delta s_1 + \delta s_2)}
\]
$\Delta$ designates a difference across the face (i.e., $\Delta p = p_2 - p_1$). In Eq. 4.5, the subscript $U$ designates the quantity associated with the element upwind of face $f$ (which can vary with field), and $d\mathbf{r}$ is the vector from the upwind cell center to the face center. The first-order contribution in Eq. 4.5, $\phi$, is treated implicitly, the second-order contribution, explicitly (Lien 2000, [37]).

Neglecting cross-diffusion and dilatation, the viscous flux in the momentum equations can be written for an element face as:

$$\int (\phi \cdot d\mathbf{A}) \equiv \mu (\nabla \cdot \mathbf{V})$$

(4.7)

Referring to Figure 4.2, the gradient of a scalar, $\phi$, on the face can be written as:

$$\nabla \phi = \nabla \phi - \left( \nabla \phi \cdot \frac{S_{12}}{S_{12}} \right) \hat{e}_{s_{12}} + \left( \nabla \phi \cdot \frac{S_{12}}{S_{12}} \right) \hat{e}_{s_{12}}$$

(4.8)

The terms labeled A represent components of the gradient that are orthogonal to $s_{12}$. These terms are generally small. (For hexahedral or prismatic elements extruded from geometric surfaces, neglecting them is nearly equivalent to the thin-layer assumption.) Their discrete form is treated explicitly in the solution of the momentum equations. The term labeled B represents components of the gradient that are parallel to $s_{12}$. These are discretized as:

$$\int_{f} (\mu \Delta \mathbf{V} \cdot d\mathbf{A}) = (\mu)_{f} \left( \Delta \mathbf{V} \cdot \frac{d\mathbf{r}}{d\mathbf{s}} \right) \left( \hat{e}_{s} \cdot d\mathbf{A} \right)$$

(4.9)
Gradients that appear in the flux calculations, and elsewhere, are computed using Gauss’ Law:

\[
\nabla \phi = \frac{1}{V} \sum_f \overline{A_f} \phi_f
\]

(4.10)

with internal face values of \( \phi \) computed from Eq. 4.4., and the summation taken over all faces bounding an element. Eq. 4.10 is computed by sweeping all internal and boundary faces, accumulating adjacent element contributions to \( V \) and \( \overline{A_f} \phi_f \) from the face.

### 4.5 Solution Procedure

A lagged-coefficient linearization is employed for non-linear convection terms. Invoking a dual-time formulation (with superscript \( m \) designating timestep), the discretized governing equations for transport scalar \( \phi \), can be written in \( \Delta \)-form as:

\[
\begin{bmatrix}
A_p + \frac{\rho V}{\Delta \tau} \\
\sum_{nb} A_{nb} \\
\end{bmatrix}
\Delta \phi_p + \sum_{nb} A_{nb} \Delta \phi_{nb} = \sum_{nb} \left( A_{nb} \left( \phi_{nb} \right)^m - \left( A_p \right) \left( \phi_p \right)^m \right) + S
\]

(4.11)

where \( A_p \) and \( A_{nb} \) represent cell and neighbor influence coefficients arising from convection, diffusion and implicitly treated source terms. \( S \) corresponds to explicitly treated source term, \( \Delta \phi \equiv \left( \phi \right)^{m+1} - \left( \phi \right)^m \). A standard under-relaxation procedure is
employed where an appropriate under-relaxation factor, $\omega$, is selected ($0.3 \leq \omega \leq 0.7$) and the pseudo-timestep is evaluated from:

$$\Delta \tau \equiv \frac{\omega}{1-\omega} \left( \frac{\rho^+ \alpha^+ V}{A^+_p + \sum_{k \neq i} b^{+i}} \right)$$  \hspace{1cm} (4.12)$$

4.6 Parallel Implementation

The code is parallelized based on domain decomposition using the message passing interface (MPI). Each block in the domain is distributed to a separate processor, resulting in decreased wall-clock solution time. Inter-partition boundaries are input to the flow code as all other boundaries, with an additional attribute being the neighbor partition processor number. Data is passed after each scalar is computed in the segregated procedure. For the point iterative solvers used for the scalar equations, $\Delta \phi$ is passed at every sweep of the linear solver, so that there is no degradation in convergence due to domain decomposition. For the pressure correction equation a global matrix is assembled (encompassing all partitions but with matrix component data residing locally) and solved, again ensuring that there is no degradation in convergence due to domain decomposition.

4.7 Formulation Details Specific to Respiratory Simulation

Prior to the present work, NPHASE-PSU has been applied to respiration simulations in two ways: 1) Two phase, steady inspiration model of upper airways with
particles, 2) Single phase, unsteady respiration model of upper airways with gas exchange. In this thesis the geometrical and physical modeling is extended to incorporate the lower airways.

Simulations of the first class were presented first in Kunz et al. 2003 [14], where details of the various multiphase models in Eq. 4.1 (i.e., drag, dispersion, deposition) were presented. Figure 4.3 includes two results from that work that demonstrate the ability of the steady state two-phase formulation in NPHASE-PSU to predict upper respiratory system particle deposition representative of pharmaceutical inhalation delivery.

Simulations of the second class were first presented in Leemhuis, 2004, [12], where improved segmentation of the rubber cast introduced in Chapter 2 led to 3D CFD resolution up to generation 10-12. A homogeneous multi-species gas transport formulation was employed with a very simple gas uptake model. An unsteady simulation with a prescribed sinusoidal trachea-leave pressure drop was performed. Figure 4.4 shows predicted surface oxygen content at a particular timestep during inspiration.
Figure 4.3: Predicted surface particle volume fraction and deposition efficiency vs. generation for a 5-7 generation CFD model. From Kunz et al. 2003 [14].
The geometric modeling details associated with incorporating the lower airways was presented in Chapter 3. A Quasi-one-dimensional model is employed. Accordingly, the CFD model must accommodate the Q1D geometric simplification. This involves the incorporation of two classes of body forces.

The first force, $F_p$, is that which supplants the inability of a Q1D method to support a transverse (to pipe axis) pressure gradient. Such a force is required to turn the flow towards the pipe axis as it enters a daughter branch from a parent branch. Without such a turning force, the flow exiting a parent branch does not align with the daughter branch axes if there is only one element at a given streamwise location (since a transverse pressure gradient cannot be supported). Consider a force that acts to oppose secondary components (perpendicular to branch axis) of the local velocity:

![Figure 4.4: Predicted surface oxygen content at a particular timestep during inspiration for a 10-12 generation CFD model. From Leemhuis 2004 [12].]
\[ \vec{F} = -\frac{1}{2} \dot{m} k_s \left( \vec{V} \cdot \hat{s} \right) \hat{k} \]  \hspace{1cm} (4.13)

where \( k_s \) is a positive constant. Here \( \vec{F} \) is formulated to be proportional to the local element mass through flow rate, \( \dot{m} \), directed towards the branch axis, \( \hat{s} \), and proportional to the degree to which the local velocity is misaligned with the branch axis, \( (\vec{V} \cdot \hat{s}) \). Eq. 4.13 can be written:

\[ \vec{F} = -\frac{1}{2} \dot{m} k_s \hat{s} \hat{s}^T \vec{V} \]  \hspace{1cm} (4.14)

Since \( \hat{s} \hat{s}^T \) is symmetric, the eigenvalues of the Jacobian, \( \partial \vec{F} / \partial \vec{V} \), are positive and therefore a purely implicit treatment of this force is unconditionally stable. Accordingly, the diagonal terms in Eq. 4.13 are treated implicitly and the off-diagonal terms explicitly in the segregated NPHASE-PSU solution procedure.

The second force that must be included at a minimum in the Q1D regions, \( F_v \), is that which accommodates viscous shear at the walls and its attendant effect on streamwise pressure gradient. Since the velocity profiles across the branch sections are not resolved, the wall shear must be modeled and incorporated into the discrete momentum equations.

The lower airway branches that are modeled with the Q1D method typically begin no lower than generation 5. One can estimate the maximum Reynolds number encountered in branch generation 5 and higher as follows: Table II of Yu and Diu, [16], provides an estimated diameter of Weibel generation number 5 as 3.0 mm. Taking the number branches at generation 5 as 32, and with an adult male inspiration flow of 16 liter/min [15] assumed distributed equally among these branches, yields a per branch
flow rate of approximately $8 \times 10^6$ m$^3$/s. This yields a bulk velocity estimate of 1 m/s and in turn a Reynolds number estimate of approximately 200 in generation 5. The Reynolds number decreases for all higher generations, so the laminar flow assumption is clearly valid in all of the Q1D modeled lower airways.

Consistent with the Q1D modeling and laminar flow regime of the lower airways, the streamwise body force, $F_v$, is modeled using the Poiseuille flow assumption. Specifically, the wall shear stress in Poiseuille flow is related to the average cross-sectional streamwise velocity (which is the velocity that is solved for in the Q1D problem) by:

$$\tau_w = -\frac{4\mu}{R} |V|$$

Eq. 4.15 is linear in $|V|$ and always represents a momentum sink. Therefore, $F_v$ is introduced for the lower branches by treating $\tau_w$ implicitly replacing $R$ by the distance from the wall face to the element centroid (exact as $n \to \infty$) and multiplying by the wall face area magnitude.

4.8 Software Infrastructure

With the exception of AMIRA, all of the software developed and/or used in this thesis is run in an automated fashion using a PYTHON master script, BREATHE, written by the author. Although the ultimate goal of the research is to fully automate the medical imaging processes as well, these remain semi-automatic (i.e. require hand work) per the discussions in Chapter 2. After semi-automatically obtaining the upper airway STL file,
the upper airway skeleton file and the lobe STL files from AMIRA, the other processes of the medical image through CFD package are executed automatically using BREATHE. Specifically, the script executes GEO_LUNG, HARPOON, LBRANCH, TREE and all of the CFD tools (pre-processing, flow-solver, post-processing) associated with NPHASE-PSU with one master script. The schematic representation of this automated software infrastructure is shown in Figure 4.5. Since BREATHE is a rapidly evolving and straightforward script, this schematic is provided in lieu of a listing.
Figure 4.5: Schematic representation of the software infrastructure
Chapter 5

DEVELOPMENT OF RESPIRATORY SYSTEM BOUNDARY CONDITIONS
BASED ON FULLY DEVELOPED PULSATILE FLOW

5.1 Introduction

Pulsatile flow is important for assessment of time discretization strategies in the
dynamic behavior of the human respiratory flow system. Eventually we seek to model the
respiratory system using a computational domain that extends from the near-head
environment (ambient atmospheric conditions) through the nose/mouth, oro-pharyngeal
cavity, upper-through-lower respiratory system, down to the respiratory units, terminating
at the alveolar sacks. This will not involve any flow through boundary conditions since
the flow will be driven primarily by the “piston-like” volume variation within the
respiratory units, with the farfield external boundary being at or near quiescent at
constant atmospheric pressure.

In the present work, we have truncated the domain at the trachea and the lower
airway leaves at the respiratory units. Accordingly, we need to drive the unsteady
simulation through modeled pressure-driven boundary conditions. Specifically, we seek
to set a periodic pressure drop across the domain corresponding to that which would
occur had the entire respiratory system been modeled. In addition to the challenge
associated with determining these pressures, which vary in space and time, this approach
introduces numerical discretization errors associated with extrapolation of the solution transport variables (most importantly velocity), while specifying the pressure at flow-through boundaries where flow may be coming into or exiting the domain locally.

Pulsatile flows are characterized by a lag between pressure and velocity, even for incompressible systems, and in order to verify that accurate phase relations are captured in the context of the discretization and boundary condition strategies employed in the CFD code, a canonical study is performed for pulsatile pipe/channel flows here.

In this chapter, the analytical solutions for fully developed pulsatile pipe and channel flow are derived, these derivations drawing on [31] and [32], respectively. These derivations are followed by CFD verification studies using NHASE-PSU.

5.2 Governing Equations for Unsteady Fully Developed Pipe Flow

The analysis of respiratory flow requires the full Navier Stokes equations. However, for the canonical pulsatile pipe and channel systems considered in this chapter, the governing equations can be simplified considerably:

i. The flow is incompressible.

ii. The flow is fully developed.

iii. The pipe has circular cross section.

iv. The pipe is rigid and symmetrical.

v. There are no external forces.

The axis of the pipe is in x coordinate, and r and θ are the polar coordinates. The velocity components of the flow are u, v and w in the x, r and θ directions, respectively.
For an incompressible Newtonian fluid, the continuity and three momentum equations in this coordinate system are as follows:

\[
\rho \left( \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial r} + w \frac{\partial u}{\partial \theta} \right) + \frac{\partial P}{\partial x} = \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial \theta^2} \right)
\]

\[
\rho \left( \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial r} + w \frac{\partial v}{\partial \theta} \right) + \frac{\partial P}{\partial r} = \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial r^2} + \frac{\partial^2 v}{\partial \theta^2} \right) - \frac{v}{r^2} \frac{\partial^2 v}{\partial \theta^2} + \frac{w}{r^2} \frac{\partial^2 v}{\partial \theta^2} + \frac{2}{r^2} \frac{\partial v}{\partial \theta}
\]

\[
\rho \left( \frac{\partial w}{\partial t} + u \frac{\partial w}{\partial x} + v \frac{\partial w}{\partial r} + w \frac{\partial w}{\partial \theta} + v \frac{\partial w}{\partial \theta} \right) + \frac{1}{r} \frac{\partial P}{\partial \theta} = \mu \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial r^2} + \frac{\partial^2 w}{\partial \theta^2} \right) - \frac{w}{r^2} \frac{\partial^2 w}{\partial \theta^2} + \frac{2}{r^2} \frac{\partial w}{\partial \theta}
\]

\[(5.1)\]

For the fully developed flow in a pipe, the derivatives in the x direction are zero.

\[
\frac{\partial}{\partial x} = 0
\]

\[(5.2)\]

By virtue of conditions iii, iv and v, the angular velocity and all derivatives in the \( \theta \) directions are zero:

\[
\frac{\partial}{\partial \theta} = 0
\]

\[(5.3)\]

When equations 5.2 and 5.3 are considered, along with the fact that the radial velocity must be zero at the wall, the continuity equation reduces to:

\[
\frac{\partial v}{\partial r} \frac{v}{r} = 0
\]

\[(5.4)\]
By considering the equations 5.2, 5.3 and 5.4, the radial and the axial component of the momentum conservation given in the Eq. 5.1 reduce respectively to:

\[ \frac{\partial (rv)}{\partial r} = 0 \]

\[ rv = \text{constant} \]

\[ r_{\text{wall}} \neq 0, \quad v_{\text{wall}} = 0; \quad \text{then,} \]

\[ v = 0 \]

By considering the equations 5.2, 5.3 and 5.4, the radial and the axial component of the momentum conservation given in the Eq. 5.1 reduce respectively to:

\[ \frac{\partial P}{\partial r} = 0 \quad (5.5) \]

\[ \rho \frac{\partial u}{\partial t} + \frac{\partial P}{\partial x} = \mu \left( \frac{\partial^2 u}{\partial r^2} + \frac{1}{r} \frac{\partial u}{\partial r} \right) \quad (5.6) \]

So under conditions, i, ii, iii, iv and v, the governing equations for unsteady fully developed pipe flow, are given by Eq. 5.6. The velocity is only a function of radial coordinate and time and the pressure is only the function of axial coordinate and time.

### 5.3 Analytical Solution for Pulsatile Flow in Pipes

The analytical solutions for Eq. 5.6 can be obtained for the case that the driving pressure is oscillatory in time. The pressure oscillates around a mean value as a sine or cosine function. The mean value is the steady state pressure varying in the x direction, \( P_{\text{steady}}(x) \). The oscillatory value changes both in x and time, \( P_{\text{oscillatory}}(x,t) \).

\[ u(r,t) = u_{\text{steady}}(r) + u_{\text{oscillatory}}(r,t) \]
\[ P(x,t) = P_{\text{steady}}(x) + P_{\text{oscillatory}}(x,t) \]

When these are inserted into the governing equation, terms which are only a function of \( x \) and the remaining terms which are the function of \( x \) and \( t \) can be grouped:

\[
\rho \frac{\partial u_{\text{oscillatory}}}{\partial t} + \frac{dP_{\text{steady}}}{dx} + \frac{\partial P_{\text{oscillatory}}}{\partial x} = 0
\]

\[
\mu \left( \frac{d^2 u_{\text{steady}}}{dr^2} + \frac{\partial^2 u_{\text{oscillatory}}}{\partial r \partial x} \right) + \frac{\mu}{r} \left( \frac{du_{\text{steady}}}{dr} + \frac{\partial u_{\text{oscillatory}}}{\partial r} \right) = 0
\]

(5.7)

(5.7.1) \[ \frac{dP_{\text{steady}}}{dx} - \mu \frac{d^2 u_{\text{steady}}}{dr^2} - \frac{\mu}{r} \frac{du_{\text{steady}}}{dr} = 0 \]

(5.7.2) \[ \rho \frac{\partial u_{\text{oscillatory}}}{\partial t} + \frac{\partial P_{\text{oscillatory}}}{\partial x} - \mu \frac{\partial^2 u_{\text{oscillatory}}}{\partial r^2} - \frac{\mu}{r} \frac{\partial u_{\text{oscillatory}}}{\partial r} = 0 \]

Eq. 5.7.1 is the steady state form of the Eq. 5.7. Eq. 5.7.2 is oscillatory part of Eq. 5.7.

The steady equation, Eq. 5.7.1 describes steady Poiseuille flow and its solution is as follows:

\[
\frac{dP_{\text{steady}}}{dx} = \mu \frac{d^2 u_{\text{steady}}}{dr^2} + \frac{\mu}{r} \frac{du_{\text{steady}}}{dr} = \text{constant} = k_{\text{steady}}
\]

\[
P_{\text{steady}}(x) = k_{\text{steady}} x + P_0
\]

\[
u_{\text{steady}}(r) = \frac{k_{\text{steady}} r^2}{4\mu} + A \ln r + B
\]

(5.8)

For a pipe, whose radius and length are \( a \) and \( L \), respectively, the no-slip boundary condition at \( r \) equals radius and a finite velocity boundary condition at \( r \) equals zero yields the classical Poiseuille flow solution, Eq. 5.9.
For the oscillatory part, Eq. 5.6.2, the pressure can be written in a manner similar to that of the steady state case:

\[ k_{\text{steady}} = \frac{P_{\text{steady}}(L) - P_{\text{steady}}(0)}{L} \]

Then the Eq. 5.6.2 can be written as follows:

\[ -\rho \cdot \frac{\partial u_{\text{oscillatory}}}{\partial t} + \mu \left( \frac{\partial^2 u_{\text{oscillatory}}}{\partial r^2} + \frac{1}{r} \cdot \frac{\partial u_{\text{oscillatory}}}{\partial r} \right) = k_{\text{oscillatory}} \]  

(5.10)

If the variation of pressure in time, \( k_{\text{oscillatory}} \) is known, then the equation for \( u_{\text{oscillatory}}(r,t) \) can be written as in Eq.5.11, and solved analytically. Additionally, if the oscillatory pressure gradient is expressed as an oscillatory function whose amplitude is \( k_{\text{steady}} \), then the peak value of oscillatory velocity profile can be compared with that of Poiseuille flow.

\[ -\rho \cdot \frac{\partial u_{\text{oscillatory}}}{\partial t} + \mu \left( \frac{\partial^2 u_{\text{oscillatory}}}{\partial r^2} + \frac{1}{r} \cdot \frac{\partial u_{\text{oscillatory}}}{\partial r} \right) = k_{\text{steady}} \left( \cos(\alpha) + i \sin(\alpha) \right) = k_{\text{steady}} e^{i\alpha} \]

(5.11)

This PDE can be reduced to an ODE by application of separation of variables. When the oscillatory velocity is separated into x-dependent and t-dependent components,
the part depending on must assume an exponential form to satisfy the right hand side of Eq. 5.11. Specifically, for \( u_{oscillatory}(r,t) = U(r)e^{i\omega t} \), Eq. 5.11 reduces Eq. 5.12.

\[
\frac{d^2U}{dr^2} + \frac{1}{r} \frac{dU}{dr} - i \rho U \omega = \frac{k_{steady}}{\mu} \tag{5.12}
\]

This ODE is the Bessel equation which has a known general solution, [26], [27]:

\[
U(r) = \frac{i \mu k_{steady}}{\rho \omega} + AJ_0(\zeta) + BY_0(\zeta) \tag{5.13}
\]

Where \( J_0 \) is the Bessel function of the first kind (Figure 5.1) and \( Y_0 \) is the Bessel function of the second kind, both of order zero. Both of them satisfy the standard Bessel equation:

\[
\frac{d^2J_0}{dx^2} + \frac{1}{x} \frac{dJ_0}{dx} + J_0 = 0 \tag{5.14}
\]

\[
J_0(x) = \sum_{m=1}^{\infty} \frac{(-1)^m}{(m!)^2} \left( \frac{x}{2} \right)^{2m}
\]

Figure 5.1: Bessel functions of the first kind at order zero, given in Eq. 5.14.
The independent variable, $\zeta$, in Eq. 5.13 is a complex number defined as:

$$\zeta(r) = \Lambda \frac{r}{a}$$

with $\Lambda = \frac{\sqrt{2}}{i} \Omega$, complex frequency parameter, \hfill (5.16)

and $\Omega = \sqrt{\frac{\mu \omega}{\rho}} a$, nondimensional frequency.

A and B are constants to be found by applying boundary conditions. The boundary conditions are finite velocity at the center of the pipe and zero velocity at the pipe wall:

**Boundary condition 1:** $r=0$, $u(0,t)=$finite, $U(0)=$finite, $\zeta=0$.

$$U(0) = \frac{i \mu k_{steady}}{\rho \omega} + AJ_0(0) + BY_0(0)$$

$Y_0(0) \to \infty$

$B = 0$

**Boundary condition 2:** $r=a$, $u(a,t)=0$, $U(a)=0$, $\zeta=\Lambda$. 

$$d^2Y_0 \over dx^2 + \frac{1}{x} dY_0 \over dx + Y_0 = 0$$

$$Y_0(x) = 2 \pi \left[ \ln \left( \frac{x}{2} \right) + \gamma \right] J_0(x) + \sum_{m=1}^{\infty} \left( -1 \right)^{m+1} H_m \left( \frac{x^2}{4} \right) (m!)^2$$

$$H_m = \sum_{k=1}^{m} \frac{1}{k}$$

$$\gamma = \lim_{n \to \infty} \left( H_n - \ln n \right)$$

\hfill (5.15)
\[
U(\Lambda) = \frac{ik_{\text{steady}}}{\rho \omega} + AJ_0(\Lambda) = 0
\]

\[
A = -\frac{ik_{\text{steady}}}{\rho \omega} \frac{1}{J_0(\Lambda)}
\]

Inserting the constants A and B into Eq. 5.13 yields the final result for \( U(r) \):

\[
U(r) = \frac{i\mu k_{\text{steady}}}{\rho \omega} \left( 1 - \frac{J_0(\xi)}{J_0(\Lambda)} \right) \cos(\omega t) + i \sin(\omega t) \tag{5.17}
\]

The oscillatory velocity is thereby obtained analytically in Eq. 5.18.

\[
u_{\text{oscillatory}}(r,t) = \frac{i\mu k_{\text{steady}}}{\rho \omega} \left( 1 - \frac{J_0(\xi)}{J_0(\Lambda)} \right) \left( \cos(\omega t) + i \sin(\omega t) \right) \tag{5.18}
\]

Finally the unsteady velocity for the fully developed flow in a symmetrical pipe of circular cross section is obtained analytically for the case of oscillatory pressure forcing in Eq. 5.19.

\[
u(r,t) = \frac{k_{\text{steady}}}{4 \mu} (r^2 - a^2) + \frac{i\mu k_{\text{steady}}}{\rho \omega} \left( 1 - \frac{J_0(\xi)}{J_0(\Lambda)} \right) \left( \cos(\omega t) + i \sin(\omega t) \right) \tag{5.19}
\]

Figure 5.2 shows the parabolic steady component of the Poiseuille velocity profile, nondimensionalized by its maximum, \( u_{\text{steady}}(0) \).
Figure 5.3 shows the oscillatory analytical flow profiles at different phase angles $\omega t$, nondimensionalized by $u_{\text{steady}}(0)$. Here, the phase angle ranges from 0 to 360 degrees by 90 degrees, and a nondimensional frequency, $\Omega$, of 3 is chosen. In Figure 5.4, the oscillatory analytical flow profiles are plotted at different phase angles, $\omega t$, also nondimensionalized by $u_{\text{steady}}(0)$. Here, the phase angle ranges from 0 to 360 degrees by 45 degrees, and a nondimensional frequency, $\Omega$, of 1 is chosen.
Figure 5.3: Oscillatory flow profiles at different phase angles $\omega t$, nondimensionalized by $u_{\text{steady}}(0)$. Phase angle ranges from 0 to 360 degrees by 45 degrees. Nondimensional frequency, $\Omega=3$. 
Figure 5.4: Oscillatory flow profiles at different phase angles ($\omega t$), nondimensionalized by $u_{\text{steady}}(0)$. Phase angle ranges from 0 to 360 degrees by 45 degrees. Nondimensional frequency, $\Omega = 1$. 
5.4 Governing Equation for Unsteady Fully Developed Flow in Rectangular Ducts

The result obtained in Section 5.3 is valid for fully developed pulsatile flow in a pipe of circular cross-section. Although the author could use that result directly to validate NPHASE-PSU, an equally relevant system of fully developed pulsatile flow through a duct of rectangular cross section is used in a further reduced form for CFD validation. The reason of this choice is a practical one; NPHASE-PSU does not have a cylindrical coordinate capability, so performing axisymmetric calculations can be fairly time consuming since a fully 3D grid must be used. However, the rectangular duct system/solution reduces readily to 2D, which NPHASE-PSU does support.

The flow of an incompressible Newtonian fluid in a straight rectangular duct is considered, as illustrated in Figure 5.5. \( u, v, \) and \( w \) are the velocity components in \( x, y \) and \( z \) directions, respectively.

\[
\rho \frac{\partial u}{\partial t} = -\frac{d}{dx} P(t) + \mu \left( \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)
\]

Figure 5.5: Straight rectangular duct

Making the same fully developed flow assumptions as above, and performing the corresponding similar analysis described in section 5.2, the Navier Stokes equations for this parallel flow reduce to the following form:

\[
\rho \frac{\partial u}{\partial t} = -\frac{d}{dx} P(t) + \mu \left( \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right)
\]
The velocity is a function of \( y, z \) and time, \( u(y,z,t) \) and the pressure is a function of \( x \) and time, \( P(x,t) \).

5.5 Analytical Solution for Pulsatile Flow in Rectangular Ducts

Again the periodic pulsatile flow is described sufficiently by the superposition of the steady and oscillatory flow solutions. Steady and oscillatory equations are entirely independent of each other.

The analytical solutions for Eq. 5.20 can be obtained for the case that the driving pressure is oscillatory in time. The pressure oscillates around a mean value as a sine or cosine function as in the pulsatile pipe flow:

\[
P(x,t) = P(x)(\cos(\alpha t) + i\sin(\alpha t)) = P(x)e^{i\omega t}
\]  \hspace{1cm} (5.21)

where \( L \) is the length of the channel. Again, velocity is also oscillating around a mean value:

\[
U(y,z,t) = U(y,z)e^{i\omega t}
\]  \hspace{1cm} (5.22)

Insert 5.21 and 5.22 into 5.20:

\[
\left( \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} - \frac{i\omega \rho}{\mu} \right) U(y,z) = \frac{1}{\mu} \frac{dP}{dx}
\]  \hspace{1cm} (5.23)

Now, introduce nondimensional parameters, \( Y \) and \( Z \) by normalizing the equation with \( b \), the smaller of channel dimensions, \( a \) and \( b \), Figure 5.5:

If \( y=Ya \) and \( z=Zb \), then Eq. 5.23 becomes:
\[
\left( \frac{\partial^2}{\partial Y^2} + \frac{\partial^2}{\partial Z^2} - \frac{i \omega \rho}{\mu} b^2 \right) U(Y, Z) = \frac{1}{\mu} \frac{dP}{dx} \cdot b^2 \tag{5.24}
\]

Right hand side of the equation is a nondimensional pressure gradient amplitude.

Nondimensional frequency also appears:

\[
\Omega = \sqrt{\frac{P \omega}{\rho b}}, \text{ nondimensional frequency}
\tag{5.25}
\]

\[
P' = \frac{b^2}{\mu} \frac{dP}{dx}, \text{ nondimensional pressure gradient amplitude}
\]

So Eq. 5.24 can be written:

\[
\left( \nabla^2 - i \Omega^2 \right) U(Y, Z) = P'
\tag{5.26}
\]

where \( \nabla^2 = \frac{\partial^2}{\partial Y^2} + \frac{\partial^2}{\partial Z^2} \). A particular solution of this PDE is as follows:

\[
U_p = \frac{P'}{\Omega^2} l
\tag{5.27}
\]

The homogenous equation \( \left( \nabla^2 - i \Omega^2 \right) U(Y, Z) = 0 \) is the Helmholtz equation. By separation of variables in \( Y \) and \( Z \), the homogenous solution for the Helmholtz equation can be obtained:

\[
U_h = \sum_{k=0}^{\infty} \alpha_k \cosh \left( \sqrt{(k^2 + i \Omega^2)} Y \right) \cos(kZ) + \sum_{q=0}^{\infty} \beta_q \cosh \left( \sqrt{(q^2 + i \Omega^2)} Z \right) \cos(qY) \tag{5.28}
\]

The boundary conditions are zero velocity at the channel walls:

\[
y=\pm a, \ Y=\pm a/b, \ U(a/b,Z)=0, \ U(-a/b,Z)=0.
\]

\[
z=\pm b, \ Z=\pm 1, \ U(Y,1)=0, \ U(Y,-1)=0.
\]
Upon application of boundary conditions, the following general solution is obtained:

\[
U(Y, Z) = iP' \Omega^2 \left( 1 - 2 \sum_{n=0}^{\infty} \left\{ \frac{(-1)^n}{k_n} \left[ \cosh(\gamma_nY) \cos(k_nZ) + \cos(q_nY) \cosh(\zeta_nZ) \right] \right\} \right)
\]

\[
k_n = \frac{2n + 1}{2} \pi
\]

\[
q_n = \frac{2n + 1}{2} \frac{b}{a}
\]

\[
\gamma_n = \sqrt{k_n^2 + i\Omega^2}
\]

\[
\zeta_n = \sqrt{q_n^2 + i\Omega^2}
\]

Result 5.29 can be reduced to 2D by considering its applications at Z=0 (centerline) and as the aspect ratio, a over b goes to infinity:

\[
U(Y, 0) = iP' \Omega^2 \left( 1 - \cosh(i\Omega Y) \cosh(Y) \right)
\]

This analytical solution is obtained by assuming the driving periodic pressure to be sum of both sine and cosine functions in exponential form. Accordingly, the corresponding velocity equation can be obtained for pure sine or pure cosine oscillatory pressure. If the driving pressure is in the form of cosine function, the resulting periodic velocity becomes:

\[
u(Y, Z) = Amp \cdot \cos(\omega t + \phi)
\]

\[
Amp = \sqrt{U_{\text{real}}^2 + U_{\text{imag}}^2}
\]

\[
\phi = a \tan \left( \frac{U_{\text{imag}}}{U_{\text{real}}} \right)
\]
5.6 Comparison of Numerical Results with Analytical Result

NPHASE-PSU, is the CFD solver used in this thesis. The author has explored a number of options for wall and pressure boundary conditions including 1st order and 2nd order, no-slip, Neumann and extrapolation conditions for velocity. It is stated here without presentation, that in order to return exact numerical results for steady pipe or channel Poiseuille flow, 2nd order accuracy is required for velocity at the wall, since this captures the analytical parabolic profile there exactly. Accordingly, this NPHASE-PSU option is employed in the following analyses. Also, in the pressure gradient computation in the momentum equations, pressure must be extrapolated to constant pressure boundaries in order to exactly capture the linear streamwise pressure gradient (i.e., \((\partial p/\partial n)_{\text{boundary face}} = (\partial p/\partial n)_{\text{adjacent cell}} \) \ not \ \((\partial p/\partial n)_{\text{boundary face}} = 0\).) This NPHASE-PSU option is also employed in the following analyses.

The Cartesian structured grid shown in Figure 5.6 was generated for a channel of 0.01m x 0.01m x 0.01m.

Figure 5.6: Structured grid for cubic rectangular duct.
Here, the x=0 and x=0.01 planes are flow-through pressure boundaries, the bounding xy planes are symmetry planes (NPHASE-PSU 2D approach, 1 cell in z-direction), and the bounding xz planes are walls. The streamwise direction is comprised of only 3 axial cells, which is consistent with the fully developed nature of the solution. A range of transverse grids was selected, the 20 cell mesh shown in Figure 5.6 is that for which results are presented below.

In order to validate the code at scales that are representative of the respiratory application, all dimensional scales are selected to correspond nominally to an upper airway branch. This ensures that the non-dimensional frequency is matched. (Note: non-dimensional frequency is the only dynamic similarity parameter in these systems, so the choice of lung-relevant dimensional scales is a matter of convenience [i.e., allows for specification of a physically intuitive channel \( \Delta p \)] not necessity.) These problem parameters are given in Table 5.1.

<table>
<thead>
<tr>
<th>Table 5.1: Flow parameters for the cubic channel oscillatory flow test case</th>
</tr>
</thead>
<tbody>
<tr>
<td>b (half height of channel)</td>
</tr>
<tr>
<td>L (length of the channel)</td>
</tr>
<tr>
<td>IBI (interbreath interval)</td>
</tr>
<tr>
<td>f (frequency of breathing)</td>
</tr>
<tr>
<td>( \omega ) (angular frequency)</td>
</tr>
<tr>
<td>( \rho ) (fluid density)</td>
</tr>
<tr>
<td>( \mu ) (dynamic viscosity of fluid)</td>
</tr>
<tr>
<td>( \Omega ) (nondimensional frequency)</td>
</tr>
</tbody>
</table>

\[
\frac{dP}{dX} = \frac{P(L) - P(0)}{L} = 1 \text{ (Pa/m)}
\]

P(L)-P(0) = 0.01 (Pa)
As mentioned in Chapter 1, the average human interbreath interval ranges from 3.49 to 3.61 seconds for women and 3.61 to 3.83 seconds in men, (young to old), [13]. Therefore, the interbreath interval is chosen to be 3.7 seconds. The physical time step is set to be 0.037, one percent of the interbreath internal.

The pressure boundary conditions are set to be zero at the inlet and \(- (P(L) - P(0)) \cos(\omega t)\) at the outlet.

The flow solver NPHASE-PSU is run for two full periods. The initial conditions, which are zero velocity and zero pressure, are set everywhere in the solution domain. A python code is added to the job script to calculate and compare the analytical solution profile for each numerically computed velocity profile. This code appears in Appendix C.

Figures 5.7, 5.8 and 5.9 include the velocity profiles at different time values in the first and the second periods. In these three figures, the velocity profiles having common line and symbol attributes belong to the same phase angles at different time steps. The Figure 5.7 shows velocity profiles for the first few timesteps in the first and second periods. It is clearly evident from this figure that the numerical values are different than the analytical values for the first period (as the initial condition is error is damped) while they are in very good agreement for the second period.
Figure 5.7: Velocity profiles for periodic channel flow. Phase angle ranges from 3.6° to 39.6°.
Figure 5.7 also shows that the disagreement between numerical and analytical solutions decreases with timestep during the first period. Figure 5.8 shows that agreement continues to improve during the first period and for the second period, the NPHASE-PSU results agree almost perfectly with the analytical solution. These findings suggest that for unsteady respiration flow simulation, the second period is likely to return close to the statistically converged periodic solution.

Figures 5.8 and 5.9 show that the oscillatory velocity profile exhibits maximum and minimum values at t=4.07s and 5.92s, corresponding to phase angles of 36° and 217° respectively. By comparison, the streamwise Δp extrema occur at phase angles of 0° and 180° (by prescription). This phase lag is due to the inertia of the fluid and it is a physical characteristic of pulsating flow. If the timescale of the pressure variation is high (i.e., slow), as represented by the nondimensional frequency, \( \Omega \), then the attendant velocity will be almost in phase with the pressure. If the change in pressure is rapid, the velocity field phase lag increases. Because of phase lag, the maximum velocity reached in each cycle is less than it would be observed in steady Poiseuille flow under constant Δp equal to the peak of the oscillatory Δp.

The oscillating velocity profiles exhibit an interesting character when they transition from inflow to outflow. Specifically, Figure 5.10 shows that in this transition region, the flow near the walls can be oppositely directed from the flow near the center of the channel.

In summary, it is clear that for non-dimensional frequencies of relevance to respiratory simulation, \( \Omega = O(1) \), phase lag is quite important. The numerical
discretization strategies (i.e., transverse grid size and boundary conditions summarized above) employed in NPHASE-PSU are sufficient to capture the richness associated with laminar pulsatile flow for the non-dimensional frequencies of relevance to respiration.
Figure 5.8: Velocity profiles for periodic channel flow. Phase angle ranges from $36^\circ$ to $180^\circ$. 
Figure 5.9: Velocity profiles for periodic channel flow. Phase angle ranges from 216° to 360°.
Figure 5.10: Velocity profiles for periodic channel flow. A closer look at the transition from inflow to outflow.
Chapter 6

RESPIRATORY SYSTEM SIMULATION

6.1 Introduction

As indicated in Chapter 4, NPHASE-PSU has been applied in the past to respiration simulations in two ways: 1) Two phase, steady inspiration model of upper airways with particles, 2) Single phase, unsteady respiration model of upper airways with gas exchange. In this thesis, the geometrical and physical modeling is extended to incorporate the lower airways. In this Chapter, three simulations are presented illustrating the contribution of the elements developed in this thesis: 1) Single phase, steady 3D expiration simulation of an upper airway model derived from bronchiole CT data of a living human subject. 2) Single phase, steady Q1D expiration simulation of a lower airway model derived from lobe CT data of a living human subject and the volume filling and branch gridding methods. 3) Single phase, unsteady 3D simulation of a complete respiratory cycle of a lower airway model.

6.2 3D Steady Expiration Simulation of Upper Airways

The flow through the upper airways of the “Erna” model, described in Chapters 2-4, was first modeled. Specifically, The AMIRA generated STL file of the first 4-6 generations was processed using GEO_LUNG, and the partitioned truncated STL output of this code was input to the grid generator, from which a 197,202 cell mesh was constructed. The trachea truncation boundary was set to a pressure of -1.0 Pa, and each of
the truncated leave boundaries were set to a pressure of 0.0 Pa. This pressure difference drives the expiration. Figure 6.1 shows predicted contours of pressure on the branch walls and the velocity at the trachea and leaf branch cross-sections, directed out of the lung. Several interesting features of the simulation are observed, including the localized acceleration/deceleration features at the cartilages (rings) of the trachea, the relatively larger streamwise pressure gradients in the flow in the narrower branches near the leaves, and none of the stagnation regions characteristic of bifurcation during inspiration, Ref. [12] and [14].

Figure 6.1: Erna upper airways pressure distribution and velocity
6.3 Q1D Steady Expiration Simulation of Lower Airways

The second simulation presented is also of the Erna model, but using only the Q1D processes developed in this thesis to study steady expiration in the convective regime bronchioles. The 32 terminal leaves arising from the Erna model upper branch segmentation were specified as geometric and flow boundary conditions. The 5 Erna lobe segmentations were used as boundary conditions for the branching/lobe filling process described in Section 2.4. The difference between the present Q1D representation and that presented in Section 2.4 is that here the algorithm was applied to the Erna configuration not the rubber model. The Q1D grid generation scheme developed in Section 3.3 was applied.

Table 6.1 lists the three different simulations that were run. The only difference between these runs is the total depth of penetration of the Q1D branching algorithm. This was parameterized to explore the effect of this “Maximum # of generations” parameter on the predictions. Figures 6.2 through 6.4 show predicted pressure distributions for the same $\Delta p$ applied across the entire convective regime.

<table>
<thead>
<tr>
<th>Case</th>
<th>Max. generations #</th>
<th># of Airways</th>
<th># of Subtrees</th>
<th># of Elements</th>
<th># of Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>444</td>
<td>32</td>
<td>3958</td>
<td>49780</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>3503</td>
<td>32</td>
<td>36080</td>
<td>453948</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>75710</td>
<td>32</td>
<td>794336</td>
<td>9909626</td>
</tr>
</tbody>
</table>

Table 6.1: Simulated Q1D models
Figure 6.2: Pressure distribution of Q1D lower airways of Erna; maximum generation number is 10.

Figure 6.3: Pressure distribution of Q1D lower airways of Erna; maximum generation number is 13.
Clearly the geometric pedigree of the convective regime representation improves dramatically as the number of represented airway generations increases. It is difficult to derive any flow accuracy conclusions from this simple study however, since each simulation represents significantly different expiration volume flow rates.

**Figure 6.4:** Pressure distribution of Q1D lower airways of Erna; maximum generation number is 19.
6.4 3D Unsteady Respiration Simulation of a Complete Respiratory Cycle of Lower Airways

The final simulation presented is also of the Erna model, using only the Q1D processes developed, but running NPHASE-PSU in an unsteady fashion in order to represent complete breathing cycles. For this simulation, a simple sinusoidal wave form with a 4 second period was applied to the 32 upper branch leaves, and a zero reference pressure to the terminal bronchiole leaves. Better approximations are available for both the wave form and pressure differences, but in this preliminary work, the method was applied to develop some understanding of the response of the model to the low Weibel number driving oscillatory pressure differences. The complete 19 bronchiole generation model was used. The model was run in NPHASE-PSU for 3 full boundary condition periods at which point the simulation had become periodic. Figure 6.5 shows predicted local pressure distributions at four times during a periodic cycle; maximum inspiration, full inhale, maximum expiration and full exhale. Note that blue is high pressure and red is low pressure in this figure. Several interesting features of simulation are observed including the relative uniformity of pressure within the lung at each time, and the persistence of pressure gradients in the flow, i.e, local velocities non-zero, at the full inhale and full exhale “dwells” where no net flow is occurring.
Figure 6.5: Predicted local pressure distributions at four times during a periodic cycle; maximum inspiration, full inhale, maximum expiration and full exhale.
Chapter 7
CONCLUSION

In this work, some geometric modeling and manipulation algorithms for human lung have been developed. Medical image data from in-vivo subjects has been semi-automatically processed. Upper airway geometry, the skeleton and lobe geometries were derived. A partitioning and truncation algorithm was developed and applied to the conducting airways of a dead subject. 3D unstructured gridding of trachea through generation 5 to 8 was automated. Interfacing this upper bronchi and lobe geometry with a volume filling algorithm for the subresolved bronchi was presented.

Importance of quality of CT scans was discussed as a limitation to gain the capability of fully automated image processing. Although CT is more widely available and cost less, in the future, instead of CT scans, MRI images may be used to increase the medical image quality. MRI images may provide better resolution and contrast than CT medical images.

The partitioning and truncation algorithm developed in this thesis can be modified to apply for other body systems, like bones or nerves.

In addition to these geometric modeling contributions, unsteady pressure-forced boundary conditions within a CFD solver were validated in this work by comparison of unsteady 2D/3D CFD simulations with known analytical solutions to the incompressible Navier-Stokes equations for non-dimensional frequencies and Reynolds numbers of
relevance to respiration. Importance of specification of well posed and accurate streamwise boundary conditions were discussed. Solver gave excellent comparison with analytical solution. CFD simulation of respiration was also presented.

The contributions of the presented work are: 1) the application of modern medical imaging technology and commercial image processing tools to segment lung images, within the AMIRA [23] software package, that are suitable for CFD analysis, 2) the development of grid generation schemes based on the octree method, within the HARPOON [29] software package, for discretization of the macro-scales, 3) the development of software written for quasi-one-dimensional (Q1D) geometric, grid generation and flow modeling approaches for the unresolved micro-scale convective regime, 4) the development of a multi-disciplinary, script-based simulation framework for macro-through-micro scale respiration modeling, 5) the assessment of different boundary condition approaches for the pulsatile flows/scales of interest in respiration, and, 6) the application of the overall simulation scheme to study human respiration.

The ultimate goal of this research is to automate the process from image processing through CFD analysis so that the physician will have additional information for making diagnostic, dosage or surgical option decisions.

For the next steps, adequacy of the automated grid generation and the accuracy of the semi-automated CFD calculations should be discussed. For forming new routes for future studies, discussing more closely how automated or semi-automated mathematical model of the macro to micro scale CFD could serve as a diagnostic tool for lung diseases with medical lung research centers would be beneficial. Disease specific improvements to the developed CFD tool may be necessary. Especially not for long term lung disease
treatments but for emergency problem recognitions, developed CFD tool could be used to obtain a human respiratory simulator and designed to work along with this simulator. For this, for emergency clinical lung problems, CFD tool could be run for many cases and the results could be saved as data sheets to form a simulator which requires indeed empirical data. Of course the human body is an incredibly complicated system, but still a hand made lung simulator can help to find at least the basic problems in the system beforehand. This tool could also be improved and used for education purposes.
References


[23] AMIRA 4.1 User’s Guide


Appendix A

AMIRA Skeleton Script

# Amira skeleton script file to obtain the skeleton and its connectivity information of upper airways generated by Amira

```plaintext
$this proc constructor {} {
    $this newPortFilename filename
    $this newPortText fileLocation
    $this fileLocation setValue "f:\F:\a11_TIFF_FILES\skeleton"
    $this fileLocation setLabel "Location of output files:"
    $this newPortRadioBox distChoice 2
    $this distChoice setLabel "Distance map"
    $this distChoice setLabel 0 "regular"
    $this distChoice setLabel 1 "enhanced"
    $this newPortButtonList Action 1
    $this Action setLabel Action:
    $this Action setLabel 0 "Generate iskelet"
}
$this proc destructor {} {
}
$this proc compute {} {
    if { [$this Action isNew ] } {
        set fileDir [$this fileLocation getValue]
        set filename [$this filename getValue ]
        #We create Mosaic object and convert to large disk format.
        set mosaic [create HxMosaic]
        $mosaic addBrick $filename
        set convertToLargeData [create HxMosaicToDiskData]
        $convertToLargeData data connect $mosaic
        $convertToLargeData fire
        $convertToLargeData filename setFilename "$fileDir/mosaic.am"
        $convertToLargeData doIt setValue 0 1
        $convertToLargeData fire
        set mosaicLDD [$convertToLargeData getResult]
        #Compute the distance map, either regular or enhanced
        if { [$this distChoice getValue] == 0 } {
            set distanceMap [create HxExtDistanceMap]
            $distanceMap data connect $mosaicLDD
            $distanceMap doIt setValue 0 1
        } else {
            set distanceMap [create HxLDAChamferCalc]
            $distanceMap data connect $mosaicLDD
        }
    }
}
```
$distanceMap action setValue 0 1
}
$distanceMap fire
set distanceMapResult [$distanceMap getResult]

# Perform thinning
set thinner [create HxExtThinner]
$thinner data connect $mosaicLDD
$thinner distmap connect $distanceMapResult
$thinner action setValue 0 1
$thinner fire
set thinnedResult [$thinner getResult]

# Generate skeleton
set tracer [create HxExtTraceLines]
$tracer data connect $thinnedResult
$tracer options setValue 1 1
$tracer doIt setValue 0 1
$tracer fire
set line [$tracer getResult]

# Do a little cleanup on the skeleton before saving/smoothing it
$line cleanup
set smoother [create HxSmoothLine]
$smoother lineSet connect $line
$smoother doIt setValue 0 1
$smoother fire
set smoothLine [$smoother getResult]

# Map the distance map to the lineset
set evalOnLine [create HxExtEvalOnLines]
$evalOnLine data connect $smoothLine
$evalOnLine field connect $distanceMapResult
$evalOnLine doIt setValue 0 1
$evalOnLine fire
set lineView [create HxDisplayLineSet]
$lineView data connect $smoothLine
$lineView shape setValue 6
$lineView scaleMode setValue 0 1
$lineView scaleFactor setValue 1.2
$lineView colorMode setValue 1
$lineView colormap connect physics.icol

# Compute actual min max...
$lineView colormap setMinMax 0 5
$lineView fire

# Save the lineset using Micro Visu3D file format.
$smoothLine save "MicroVisu3D ASCII" $fileDir/result.mv3d
}
/*
GEOTOOL of Lung Project
Aerospace Engineering / Penn State University
Written by: Gulkiz Dogan
November 2006
INPUT files
1. STL file = Surface geometry file = input.stl
2. Topology file = Cast.txt (skeleton file)
OUTPUT files
1. centers.dat = centers of triangles
2. edgeinfo.dat = EdgeID | Generation# | Length | Volume | Radius
3. nodecor.dat = Tecplot file (node coordinates and connectivity)
4. output.stl -- > which is truncated and assigned boundary condition
5. edgeid_outlet.dat -- > Pressure Outlet Edge ID List For Daniel Haworth’s Code
6. cobalt.bc -- > Boundary condition file for NPHASE

NOTES
edgeinfo[NN][1] -- > stores generation number of each edge, integer
edgeinfo[NN][3] -- > stores length | volume | diameter of each edge, float

NB -- > # of branches which will remain after translation
   0 corresponds to trachea.

For new triangles following applies:
   gn# = 100 -- > all triangles have this as gn at the beginning.
   If a triangle does not belong to any of the branches.
   gn# = 101 -- > originally at NB branch but above the plane; remove
   gn# = 200 -- > outlet, new triangle
   gn# = 101 -- > originally at the 0th branch but above the plane; remove
   gn# = 201 -- > inlet, new triangle

imax -- > # of triangles in input STL
jmax -- > # of triangles after newly created triangles at outlets
jmax2 -- > # of triangles after newly created triangles at inlet
   i and j must be seperated

tf, tA, tB -- > Parametric representation of a line; t is the parameter.

ke = # of triangle at the last branch

end[]][ -- > last generation[triangle id][edge id]
outletid[] -- > edge id of pressure outlets, or leaves
noutlet -- > # of pressure outlets, or leaves

Segmentation Fault -- > Look to line "read this line if STL file"
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define SIZE 1000000 //size of triangle arrays (guess)
#define NN 1434  //number of nodes as assigned in Cast.txt
#define pi 3.141563 //pi number
//#define NB 5  //number of branches wanted
//#define per 0.85    //cut percentage from bottom

int main()
{
    float normal[SIZE][3], vertex[SIZE][3][3], center[SIZE][3];
    float a, b, c, e, aver, sum[3];
    float node[NN][3], edgeinfo[NN][3];
    float point[1][3], pn[1][4]; //cut point of each edge, normal of cut plane
    float d, d1, d2, d3, tA, tB, radius, x0, y0, x1, y1, z1, x2, y2, z2, xn, yn, zn, per;
    float tf, xf, yf, zf, gamma, distance, distance_new; //footpoint and angle
    int edgenum1, edgenum2, edgenum3, edg e[NN][2], edgecounter[NN];
    int edgeinfo1[NN][1], trach[SIZE], lungr[SIZE], lungl[SIZE], leaf[9000];
    int end[SIZE][2], endinfo[100000][1], outletid[NN], number_tri;,
    int i, j, k, l, kt, kr, kl, ke, imax, jmax, jmax2, ltmax, klmax, krmax, ktmax, lmax, kemax;
    int NB, gn[SIZE], leafnumber[SIZE];
    char STL, solid[100], *line, yazi[10], dummy[10];
    FILE *input, *output;

    printf("enter the name of input stl file, like biglung.stl\n");
    scanf("%s","STL");
    printf("enter stl parameter 0-with leaf numbers \n");
    scanf("%d","k");
    printf("enter number of branches - trachea is 0 : \n");
    scanf("%d","NB");
    printf("enter cut percentage from bottom:\n");
    scanf("%f","per");
    printf("%d %f\n",NB, per);

    // Read Normals and Vertexes of surface triangles from STL file
    input = fopen(&STL,"r");
    line=fgets(solid,100,input);
    sscanf(line,"%s",yazi);//yazi represents important strings, dummy represents unimportant strings
    i=0;
    while (strcmp(yazi,"endsolid")!=0){
        line=fgets(solid,100,input); //define the line
        if(k==0) sscanf(line,"%s %s %e %e %e",yazi,dummy, &normal[i][0], &normal[i][1], &normal[i][2]);
        else sscanf(line,"%s %s %f %f %f",yazi,dummy, &normal[i][0], &normal[i][1], &normal[i][2]);
if (strcmp(yazi,"endsolid")==0) break;
if(k==0) line=fgets(solid,100,input);            // read this line if STL file
// sscanf(line, "%s %d", dummy, &leafnumber[i]);   // includes leafnumbers, otherwise erase
line=fgets(solid,100,input);
line=fgets(solid,100,input);
sscanf(line, "%s %e %e %e", dummy, &vertex[i][0][0], &vertex[i][0][1], &vertex[i][0][2]);
line=fgets(solid,100,input);
sscanf(line, "%s %e %e %e", dummy, &vertex[i][1][0], &vertex[i][1][1], &vertex[i][1][2]);
line=fgets(solid,100,input);
sscanf(line, "%s %e %e %e", dummy, &vertex[i][2][0], &vertex[i][2][1], &vertex[i][2][2]);
i++;  
}
imax=i; //imax is equal to number of triangles in stl file

// Calculate Incenter of Surface Triangles
for (i=0; i<imax; i++)
{aver = pow((vertex[i][1][0]-vertex[i][2][0]),2) +
 pow((vertex[i][1][1]-vertex[i][2][1]),2) +
 pow((vertex[i][1][2]-vertex[i][2][2]),2);
an=sqrt(aver);
acer = pow((vertex[i][2][0]-vertex[i][0][0]),2) +
 pow((vertex[i][2][1]-vertex[i][0][1]),2) +
 pow((vertex[i][2][2]-vertex[i][0][2]),2);
b=sqrt(acer);
ccer = pow((vertex[i][0][0]-vertex[i][1][0]),2) +
 pow((vertex[i][0][1]-vertex[i][1][1]),2) +
 pow((vertex[i][0][2]-vertex[i][1][2]),2);
c=sqrt(ccer);
acer=an+bc;
center[i][0] = ( a*vertex[i][0][0]+b*vertex[i][1][0]+c*vertex[i][2][0] ) / acer;
center[i][1] = ( a*vertex[i][0][1]+b*vertex[i][1][1]+c*vertex[i][2][1] ) / acer;
center[i][2] = ( a*vertex[i][0][2]+b*vertex[i][1][2]+c*vertex[i][2][2] ) / acer;
}

// Read nodes and connectivity information from Cast.txt
for (i=0; i<NN; i++) edgecounter[i]=0;
input = fopen("Cast.txt","r");
line=fgets(solid,100,input); // read unnecessary 6 lines
line=fgets(solid,100,input); // at the top of the file
line=fgets(solid,100,input); // # of edge is 1 less
line=fgets(solid,100,input); // than # of nodes
line=fgets(solid,100,input); // read node coordinates
line=fgets(solid,100,input);
line=fgets(solid,100,input);
line=fgets(solid,100,input);
sscanf(line, "\%s \%d", dummy, &k); // read number of edge assigned to this node
line=fgets(solid,100,input);
if(k==1){
    sscanf(line, "\%s \%d", dummy, &edgenum1);
    edge[edgenum1][edgecounter[edgenum1]]=i;
    edgecounter[edgenum1]++;
}
else{
    sscanf(line, "\%s \%d \%d \%d", dummy, &edgenum1, &edgenum2, &edgenum3);
    edge[edgenum1][edgecounter[edgenum1]]=i;
    edgecounter[edgenum1]++;
    edge[edgenum2][edgecounter[edgenum2]]=i;
    edgecounter[edgenum2]++;
    edge[edgenum3][edgecounter[edgenum3]]=i;
    edgecounter[edgenum3]++;
}
line=fgets(solid,100,input);
}

//Assign generation number, length and volume to each edge from Cast.txt
i=0;
while (1!=2){
    line=fgets(solid,100,input);
    sscanf(line,"\%s",yazi);
    if (strcmp(yazi,"EdgeID:")==0){
        line=fgets(solid,100,input); //2
        line=fgets(solid,100,input); //3
        line=fgets(solid,100,input); //4
        line=fgets(solid,100,input); //5
        sscanf(line,"\%s \%d",dummy, &edgeinfo1[i][0]); // generation number
        line=fgets(solid,100,input); //6
        sscanf(line,"\%s \%f",dummy, &edgeinfo[i][0]); // length
        line=fgets(solid,100,input); //7
        sscanf(line,"\%s \%f",dummy, &edgeinfo[i][1]); // volume
        i++;
    }
    if(i==NN-1) break;
}

//Calculate approx. radius of each edge
for (i=0; i<NN; i++){
    edgeinfo[i][2] = sqrt(edgeinfo[i][1] / (pi * edgeinfo[i][0]));
}

//Make the transformation for Cast.txt for skeleton and STL match
for(i=0; i<NN; i++){ // ST
    node[i][0]=node[i][0] + 11.0; // ST
    node[i][1]=node[i][1] - 133.0; // ST
    node[i][2]=node[i][2] - 529.0; // ST
}

//don't search for 0th generation
a = node[0][0] + 1.5*edgeinfo[0][2];  // a, b, c and d are dummy variables
b = node[1][0] - 1.5*(edgeinfo[0][2]);
c = node[0][1] - 1.5*(edgeinfo[0][2]);
\[ d = \text{node}[1][1] + 1.5 \times (\text{edgeinfo}[0][2]); \quad // \text{d is not distance} \]

\[ \text{kt} = 0; \text{kr} = 0; \text{kl} = 0; \]

\[ \text{for}(i = 0; i < \text{imax}; i++) \{
    \text{if(}\text{center}[i][0] < a \quad \&\& \quad \text{center}[i][0] > b \quad \&\& \quad \text{center}[i][1] > c \quad \&\& \quad \text{center}[i][1] < d \quad \&\& \quad \text{center}[i][2] = \text{node}[1][2])\{
        \text{gn}[i] = 0; \text{trach}[\text{kt}] = i; \text{kt}++;
    \}
    \text{else}\{
        \text{if(}\text{center}[i][0] > \text{node}[1][0])\{
            \text{lungr}[\text{kr}] = i; \text{kr}++;
        \}
        \text{else}\{
            \text{lungr}[\text{kl}] = i; \text{kl}++;
        \}
    \}
\}\]

\[ \text{ktmax} = \text{kt}; \text{klmax} = \text{kl}; \text{krmax} = \text{kr}; \]

\[ // \text{ktmax} = 18283 \text{ klmax} = 275928 \text{ krmax} = 223883 \text{ imax} = 562798 \text{ leaf} = 44704 \]

\[ \text{//, ..................................trachea.................................................}\]
\[ x1 = \text{node}[0][0]; \quad // \text{First node of the edge, 1} \]
\[ y1 = \text{node}[0][1]; \]
\[ z1 = \text{node}[0][2]; \]
\[ x2 = \text{node}[1][0]; \quad // \text{Second node of the edge, 2} \]
\[ y2 = \text{node}[1][1]; \]
\[ z2 = \text{node}[1][2]; \]
\[ \text{xn} = (x2-x1) / \text{edgeinfo}[0][0]; \quad // \text{normal is towards node1; up} \]
\[ \text{yn} = (y2-y1) / \text{edgeinfo}[0][0]; \]
\[ \text{zn} = (z2-z1) / \text{edgeinfo}[0][0]; \]
\[ \text{for(kt} = 0; \text{kt} < \text{ktmax}; \text{kt}++) \{
    \text{x0} = \text{center}[\text{trach}[\text{kt}]][0]; \]
\[ \text{y0} = \text{center}[\text{trach}[\text{kt}]][1]; \]
\[ \text{z0} = \text{center}[\text{trach}[\text{kt}]][2]; \]
\[ \text{d} = \text{xn}*x0 + \text{yn}*y0 + \text{zn}*z0 - (\text{xn}*x2 + \text{yn}*y2 + \text{zn}*z2); \quad //\text{distance between point and plane} \]
\[ \text{if(}d <= 0.0) \text{gn}[\text{trach}[\text{kt}]] = 0; \]
\[ \text{else gn}[\text{trach}[\text{kt}]] = 1; \]
\]

\[ \text{//, ..................................right lobe.............................................}\]
\[ \text{ke} = 0; \quad //\text{end edge counter} \]
\[ \text{for(kr} = 0; \text{kr} < \text{krmax}; \text{kr}++) \{ //1 \]
\[ \text{x0} = \text{center}[\text{lungr}[\text{kr}]][0]; \]
\[ \text{y0} = \text{center}[\text{lungr}[\text{kr}]][1]; \]
\[ \text{z0} = \text{center}[\text{lungr}[\text{kr}]][2]; \]
\[ \text{gn}[\text{lungr}[\text{kr}]] = 100; \text{j} = 0; \text{distance} = 1000.0; \]}
while(j<NN-1){//2
    j++;
    x1 = node[edge[j][0]][0];       // First node of the edge, 1
    y1 = node[edge[j][0]][1];
    z1 = node[edge[j][0]][2];
    x2 = node[edge[j][1]][0];       // Second node of the edge, 2
    y2 = node[edge[j][1]][1];
    z2 = node[edge[j][1]][2];

    tf =   ((x2-x1)*(x0-x1) + (y2-y1)*(y0-y1) + (z2-z1)*(z0-z1)) / 
      ( ((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) + (z2-z1)*(z2-z1)) + 
        ((x2-x1)*(x0-x1) + (y2-y1)*(y0-y1) + (z2-z1)*(z0-z1)) );
    xf = x1 + (x2-x1)*tf;
    yf = y1 + (y2-y1)*tf;
    zf = z1 + (z2-z1)*tf;

    distance_new = sqrt((xf-x0)*(xf-x0) + (yf-y0)*(yf-y0) + (zf-z0)*(zf-z0)); // DISTANCE between footpoint and center

    gamma = (180./pi)*acos( (normal[lungr][kr][0]*(x0-xf)+normal[lungr][kr][1]*(y0-yf)+normal[lungr][kr][2]*(z0-zf) )/distance_new);

    if(distance_new<distance) {//5
        distance = distance_new;
        gn[lungr]=edgeinfo1[j][0]; //triangle id
        end[ke][0]=lungr; end[ke][1]=j;
    }
};//5
};//2

if(gn[lungr][kr]==NB) ke++;
};//1

//...........................................left lobe............................................
for(kl=0; kl<klmax; kl++)//1
x0 = center[lungl[kl]][0];
y0 = center[lungl[kl]][1];
z0 = center[lungl[kl]][2];

gn[lungl[kl]]=100; j=0; distance=1000.0;

while(j<801)//2
j++;

x1 = node[edge[j][0]][0];       // First node of the edge, 1
y1 = node[edge[j][0]][1];
z1 = node[edge[j][0]][2];
x2 = node[edge[j][1]][0];  // Second node of the edge, 2
y2 = node[edge[j][1]][1];
z2 = node[edge[j][1]][2];

xn = (x2-x1) / edgeinfo[j][0];  // normal is towards node2; down
yn = (y2-y1) / edgeinfo[j][0];
z2 = (z2-z1) / edgeinfo[j][0];

d= xn*x0 + yn*y0 + zn*z0 - ( xn*x2 + yn*y2 + zn*z2 ); //distance between point and plane 2

if(d<=0.0 && abs(d)<=1.05*edgeinfo[j][0]){ //3 point is between plane 1 and plane 2
  a = (y2-y0)*(z1-z0)-(y1-y0)*(z2-z0);
b = -(x2-x0)*(z1-z0) + (z2-z0)*(x1-x0);
c = (x2-x0)*(y1-y0) -(y2-y0)*(x1-x0);

  radius = sqrt(a*a + b*b + c*c) /edgeinfo[j][0];

  if(radius<(5.0*edgeinfo[j][2]){//4
    tf =   (((x2-x1)*(x0-x1) + (y2-y1)*(y0-y1) + (z2-z1)*(z0-z1)) /
     ( (((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1) + (z2-z1)*(z2-z1)) +
       ((x2-x1)*(x0-x1) + (y2-y1)*(y0-y1) + (z2-z1)*(z0-z1)) ) ;

    xf = x1 + (x2-x1)*tf;
yf = y1 + (y2-y1)*tf;
zf = z1 + (z2-z1)*tf;

    distance_new = sqrt((xf-x0)*(xf-x0) + (yf-y0)*(yf-y0) +zf-z0)*(zf-z0)); // DISTANCE between footpoint and center

    if(distance_new<distance) {//5
      distance = distance_new;
gn[lungr[kr]]=edgeinfo1[j][0];
    }
  }
}
}
}
if(gn[lungr[kr]]==NB){//6
  end[ke][0]=lungr[kr]; //triangle id
  end[ke][1]=j;
}//6
}//5
}//4
}//3
}//2
if(gn[lungr[kr]]==NB) ke++;

kemax=ke; //# of triangle at the last branch

.isTrue\......SHIFT................................................
for(k=0; k<3; k++) sum[k]=0.;
number_tri=0;
for(ke=0; ke<kemax; ke++) {
    i = end[ke][0]; // triangle id
    x1 = node[edge[end[ke][1]][0]][0]; // First node of the edge, 1
    y1 = node[edge[end[ke][1]][0]][1];
    z1 = node[edge[end[ke][1]][0]][2];
    x2 = node[edge[end[ke][1]][1]][0]; // Second node of the edge, 2
    y2 = node[edge[end[ke][1]][1]][1];
    z2 = node[edge[end[ke][1]][1]][2];

    // Plane Normal
    pn[0][0] = (x2 - x1) / edgeinfo[end[ke][1]][0]; // normal is towards node 1; up
    pn[0][1] = (y2 - y1) / edgeinfo[end[ke][1]][0];
    pn[0][2] = (z2 - z1) / edgeinfo[end[ke][1]][0];
    point[0][0] = x1 + 0.95 * (x2 - x1); // point coordinates
    point[0][1] = y1 + 0.95 * (y2 - y1);
    point[0][2] = z1 + 0.95 * (z2 - z1);
    pn[0][3] = - (pn[0][0] * point[0][0] + pn[0][1] * point[0][1] + pn[0][2] * point[0][2]);
    // fourth term of plane equation, constant d = -(nxP x + nyPy + nzPz)

    // NOW WE HAVE THE EQUATION OF THE PLANE
    // Test the Triangle
    d1 = pn[0][0]*vertex[i][0][0] + pn[0][1]*vertex[i][0][1] + pn[0][2]*vertex[i][0][2] +
         pn[0][3];
    d2 = pn[0][0]*vertex[i][1][0] + pn[0][1]*vertex[i][1][1] + pn[0][2]*vertex[i][1][2] +
         pn[0][3];
    d3 = pn[0][0]*vertex[i][2][0] + pn[0][1]*vertex[i][2][1] + pn[0][2]*vertex[i][2][2] +
         pn[0][3];

    // duzlemle kesismeyenler
    if(d1*d2>0 && d1*d3>0) { // no intersection // 0th if starts
        gn[i]=NB;
    }
    else { // 0th else starts
        for(k=0; k<3; k++) sum[k] = center[i][k]; // sum[3] float
        number_tri=number_tri+1; // int number_tri
    }

    // orta noktayi yeniden hesapla
    for(k=0; k<3; k++) node[edge[end[ke][1]][0]][0][k] = sum[k] / number_tri;
} /* // .................. SHIFT ....................... */

// .......... TRUNCATION ...............

j = imax;
for(ke=0; ke<kemax; ke++) {
    i = end[ke][0]; // triangle id
    x1 = node[edge[end[ke][1]][0]][0]; // First node of the edge, 1
    y1 = node[edge[end[ke][1]][0]][1];

\[ z_1 = \text{node}[\text{edge}[\text{end}[\text{ke}][1]][0]][2]; \]
\[ x_2 = \text{node}[\text{edge}[\text{end}[\text{ke}][1]][1]][0]; \quad \text{// Second node of the edge, 2} \]
\[ y_2 = \text{node}[\text{edge}[\text{end}[\text{ke}][1]][1]][1]; \]
\[ z_2 = \text{node}[\text{edge}[\text{end}[\text{ke}][1]][1]][2]; \]

// Plane Normal
\[ \text{pn}[0][0] = (x_2 - x_1) / \text{edgeinfo}[\text{end}[\text{ke}][1]][0]; \quad \text{// normal is towards node 1; up} \]
\[ \text{pn}[0][1] = (y_2 - y_1) / \text{edgeinfo}[\text{end}[\text{ke}][1]][0]; \]
\[ \text{pn}[0][2] = (z_2 - z_1) / \text{edgeinfo}[\text{end}[\text{ke}][1]][0]; \]
\[ \text{point}[0][0] = x_1 + \text{per} * (x_2 - x_1); \quad \text{// point coordinates} \]
\[ \text{point}[0][1] = y_1 + \text{per} * (y_2 - y_1); \]
\[ \text{point}[0][2] = z_1 + \text{per} * (z_2 - z_1); \]
\[ \text{pn}[0][3] = - (\text{pn}[0][0]*\text{point}[0][0] + \text{pn}[0][1]*\text{point}[0][1] + \text{pn}[0][2]*\text{point}[0][2]); \quad \text{// fourth term of plane equation, constant d = -(nxPx + nyPy + nzPz)} \]

// NOW WE HAVE THE EQUATION OF THE PLANE

// Test the Triangle
\[ d_1 = \text{pn}[0][0]*\text{vertex}[i][0][0] + \text{pn}[0][1]*\text{vertex}[i][0][1] + \text{pn}[0][2]*\text{vertex}[i][0][2] + \text{pn}[0][3]; \]
\[ d_2 = \text{pn}[0][0]*\text{vertex}[i][1][0] + \text{pn}[0][1]*\text{vertex}[i][1][1] + \text{pn}[0][2]*\text{vertex}[i][1][2] + \text{pn}[0][3]; \]
\[ d_3 = \text{pn}[0][0]*\text{vertex}[i][2][0] + \text{pn}[0][1]*\text{vertex}[i][2][1] + \text{pn}[0][2]*\text{vertex}[i][2][2] + \text{pn}[0][3]; \]

\[ \text{if}(d_1*d_2>0 \&\& d_1*d_3>0 \&\& d_2*d_3>0) \{ \quad \text{// no intersection} \quad \text{// 0th if starts} \]
\[ \text{if}(d_1<0) \quad \text{// This triangle is under the plane, store it} \quad \text{// 1st if starts} \]
\[ \text{gn}[i]=\text{NB}; \]
\[ \text{else} \quad \text{gn}[i]=101; \]
\[ \} \quad \text{// 0th else starts} \]
\[ \} \quad \text{// 0th else ends} \]
\[ \} \quad \text{// 2nd if ends} \]
\[ \} \quad \text{// 3rd if ends} \]
\[ \text{if}(d_2*d_3>0) \{ \quad \text{// 2nd else starts} \]
\[ \text{if}(d_1<0) \quad \text{// 3rd else starts} \]
\[ \text{tA} = d_1 / (d_1 - d_2); \]
\[ \text{tB} = d_1 / (d_1 - d_3); \]
\[ \text{for}(k=0; k<3; k++) \{ \]
\[ \quad \text{vertex}[i][1][k] = \text{vertex}[i][0][k] + (\text{vertex}[i][1][k] - \text{vertex}[i][0][k]) * \text{tA}; \]
\[ \quad \text{vertex}[i][2][k] = \text{vertex}[i][0][k] + (\text{vertex}[i][2][k] - \text{vertex}[i][0][k]) * \text{tB}; \]
\[ \quad \text{vertex}[j][1][k] = \text{vertex}[j][0][k]; \quad \text{// new triangle} \]
\[ \quad \text{vertex}[j][2][k] = \text{vertex}[j][1][k]; \]
\[ \quad \text{vertex}[j][0][k] = \text{point}[0][k]; \]
\[ \quad \text{normal}[j][k] = \text{normal}[i][k]; \]
\[ \quad \text{gn}[j]=200; \quad \text{endinfo}[j][0]=\text{end}[\text{ke}][1]; \]
\[ \quad j++; \]
\[ \} \quad \text{// 3rd if ends} \]
\[ \} \quad \text{// 3rd else starts} \]
\[ \} \quad \text{// new triangle} \]
\[ \text{tB} = d_1 / (d_1 - d_3); \]
\[ \text{tA} = d_1 / (d_1 - d_2); \]
\[ \text{for}(k=0; k<3; k++) \{ \]
\[ \quad \text{vertex}[j][1][k] = \text{vertex}[i][0][k] + (\text{vertex}[i][2][k] - \text{vertex}[i][0][k]) * \text{tB}; \]
\[ \quad \text{vertex}[j][2][k] = \text{vertex}[i][1][k]; \]
\[ \quad \text{vertex}[j][0][k] = \text{vertex}[i][0][k]; \]
\[ \quad \text{normal}[j][k] = \text{normal}[i][k]; \]
\[ \quad \text{gn}[j]=\text{NB}; \]
\[ \quad j++; \]
\[ \} \quad \text{// new triangle} \]
for(k=0; k<3; k++)
{
    vertex[j][0][k] = vertex[j-1][0][k];
    vertex[j][1][k] = vertex[j-1][1][k];
    vertex[j][2][k] = point[0][k];
    normal[j][k] = pn[0][k];
    gn[j]=200; endinfo[j][0]=end[ke][1];
    j++;
}       //3rd else ends
}
}      //2nd if end
if(d1*d2>0){    //4th if starts
    if(d3<0){    //5th if starts
        tA = d1 / (d1 - d3);
        tB = d2 / (d2 - d3);
        for(k=0; k<3; k++){
            vertex[i][0][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k] )*tA;
            vertex[i][1][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
            vertex[j][0][k] = vertex[i][0][k]; //new triangle
            vertex[j][1][k] = vertex[i][1][k];
            vertex[j][2][k] = point[0][k];
            normal[j][k] = pn[0][k];
            gn[j]=200; endinfo[j][0]=end[ke][1];
            j++;
        }       //5th if ends
    }       //5th else starts
    else{      //5th else starts
        //new triangle
        tA = d1 / (d1 - d3);
        tB = d2 / (d2 - d3);
        for(k=0; k<3; k++){
            vertex[i][0][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
            vertex[i][2][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k])*tA;
            vertex[j][0][k] = vertex[i][0][k];
            vertex[j][1][k] = vertex[i][1][k];
            vertex[j][2][k] = vertex[i][2][k];
            normal[j][k] = normal[i][k];
            gn[j]=NB;
            j++;
        }       //5th else ends
    }
}      //4th if end

if(d1*d3>0){    //6th if starts
    if(d2<0){     //7th if starts
        tA = d1 / (d1 - d2);
        tB = d2 / (d2 - d3);
        for(k=0; k<3; k++){
            vertex[i][0][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tA;
            vertex[i][2][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
            vertex[j][0][k] = vertex[i][0][k]; //new triangle
            vertex[j][2][k] = vertex[i][2][k];
            }        //5th else ends
        }
}      //6th if start
vertex[j][1][k] = point[0][k];
normal[j][k] = pn[0][k];
gn[j]=200; endinfo[j][0]=end[ke][1];
j++;
}   //7th if ends
else{   //7th else starts

//new triangle
tA = d1 / (d1 - d2);
tB = d2 / (d2 - d3);
for(k=0; k<3; k++){
    vertex[j][0][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
    vertex[i][1][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tA;
    vertex[j][1][k] = vertex[i][1][k];
    vertex[j][2][k] = vertex[i][2][k];
    normal[j][k] = normal[i][k];
}
gn[j]=NB;
j++;
//new triangle
for(k=0; k<3; k++){
    vertex[j][1][k] = vertex[j-1][1][k];
    vertex[j][0][k] = vertex[j-1][0][k];
    vertex[j][2][k] = point[0][k];
    normal[j][k] = pn[0][k];
}
gn[j]=200; endinfo[j][0]=end[ke][1];
j++;
}   //7th else ends
}   //6th if ends

jmax=j; //new number of triangles by newly created triangles at outlets

//Truncation for inlet
for(kt=0; kt<ktmax; kt++){
i=trach[kt]; //triangle id
if(gn[i]==0){
x1 = node[0][0];       // First node of the edge, 1
y1 = node[0][1];
z1 = node[0][2];
x2 = node[1][0];       // Second node of the edge, 2
y2 = node[1][1];
z2 = node[1][2];

//Plane Normal
pn[0][0]= (x1-x2) / edgeinfo[0][0]; //normal is towards 2. node; down
pn[0][1]= (y1-y2) / edgeinfo[0][0];
pn[0][2]= (z1-z2) / edgeinfo[0][0];
point[0][0]= x1 + (0.0) * ( x2 - x1 ); //point coordinates
point[0][1]= y1 + (0.0) * ( y2 - y1 );
point[0][2]= z1 + (0.0) * ( z2 - z1 );
pn[0][3]= - ( pn[0][0]*point[0][0] + pn[0][1]*point[0][1] + pn[0][2]*point[0][2] ); //fourth term of plane equation, constant d = -(nxPx + nyPy +nzPz)

//NOW WE HAVE THE EQUATION OF THE PLANE
// Test the Triangle

d1 = pn[0][0]*vertex[i][0][0] + pn[0][1]*vertex[i][0][1] + pn[0][2]*vertex[i][0][2] + pn[0][3];
d2 = pn[0][0]*vertex[i][1][0] + pn[0][1]*vertex[i][1][1] + pn[0][2]*vertex[i][1][2] + pn[0][3];
d3 = pn[0][0]*vertex[i][2][0] + pn[0][1]*vertex[i][2][1] + pn[0][2]*vertex[i][2][2] + pn[0][3];

if(d1*d2>0 && d1*d3>0 && d2*d3>0) { // no intersection
  if(d1<0) // This triangle is under the plane, store it
    gn[i]=0;
  else gn[i]=101;
} else {
  //0th else starts
  if(d2*d3>0) {
    if(d1<0) //3rd if starts
      tA = d1 / (d1 - d2);
tB = d1 / (d1 - d3);
    for(k=0; k<3; k++)
      vertex[i][1][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tA;
      vertex[i][2][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k])*tB;
    //new triangle
    vertex[j][1][k] = vertex[i][1][k];
    vertex[j][2][k] = vertex[i][2][k];
    normal[j][k] = pn[0][k];
    gn[j]=201;
    j++;
  } else {
    //3rd else starts
    //new triangle
    tB = d1 / (d1 - d3);
tA = d1 / (d1 - d2);
    for(k=0; k<3; k++)
      vertex[i][1][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k])*tA;
      vertex[i][2][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tB;
    //new triangle
    vertex[j][1][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tA;
      vertex[j][2][k] = vertex[i][1][k];
    normal[j][k] = normal[i][k];
    gn[j]=0;
    j++;
  }
} else {
  //3rd else starts
  //new triangle
  tB = d1 / (d1 - d3);
tA = d1 / (d1 - d2);
  for(k=0; k<3; k++)
    vertex[j][0][k] = vertex[j-1][0][k];
    vertex[j][1][k] = vertex[j-1][1][k];
      vertex[j][2][k] = point[0][k];
  normal[j][k] = normal[i][k];
  gn[j]=201;
  j++;
}
else {
  //3rd else ends
  //2nd if end
  if(d1*d2>0) { //4th if starts
    if(d3<0) //5th if starts
      tA = d1 / (d1 - d2);
tB = d1 / (d1 - d3);
    for(k=0; k<3; k++)
      vertex[i][0][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k])*tA;
      vertex[i][1][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
  } else {

  }
}
vertex[j][0][k] = vertex[i][0][k]; //new triangle
vertex[j][1][k] = vertex[i][1][k];
vertex[j][2][k] = point[0][k];
    normal[j][k] = pn[0][k];
    gn[j]=201;
    j++;
}       //5th if ends
else{      //5th else starts
    //new triangle
    tA = d1 / (d1 - d3);
    tB = d2 / (d2 - d3);
    for(k=0; k<3; k++){
        vertex[j][0][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
        vertex[i][2][k] = vertex[i][0][k] + (vertex[i][2][k] - vertex[i][0][k])*tA;
        vertex[j][2][k] = vertex[i][2][k];
        vertex[j][1][k] = vertex[i][1][k];
        normal[j][k] = normal[i][k];
    }
    gn[j]=0;
    j++; //5th else ends
//new triangle
for(k=0; k<3; k++){
    vertex[j][0][k] = vertex[j-1][0][k];
    vertex[j][2][k] = vertex[j-1][2][k];
    vertex[j][1][k] = point[0][k];
    normal[j][k] = pn[0][k];
    gn[j]=201;
    j++;
}       //5th else ends
}

if(d1*d3>0){    //6th if starts
    if(d2<0){     //7th if starts
        tA = d1 / (d1 - d2);
        tB = d2 / (d2 - d3);
        for(k=0; k<3; k++){
            vertex[i][0][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tA;
            vertex[i][2][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
            vertex[j][0][k] = vertex[i][0][k]; //new triangle
            vertex[j][2][k] = vertex[i][2][k];
            vertex[j][1][k] = point[0][k];
            normal[j][k] = pn[0][k];
            gn[j]=201;
            j++;
        }       //7th if ends
    } else{      //7th else starts
        //new triangle
        tA = d1 / (d1 - d2);
        tB = d2 / (d2 - d3);
        for(k=0; k<3; k++){
            vertex[i][0][k] = vertex[i][1][k] + (vertex[i][2][k] - vertex[i][1][k])*tB;
            vertex[i][1][k] = vertex[i][0][k] + (vertex[i][1][k] - vertex[i][0][k])*tA;
            vertex[j][1][k] = vertex[i][1][k];
            vertex[j][2][k] = vertex[i][2][k];
            normal[j][k] = normal[i][k];
            gn[j]=0;
        }       //7th else ends
    } //6th if ends
}
j++;  //new triangle
for(k=0; k<3; k++){
    vertex[j][1][k] = vertex[j-1][1][k];
    vertex[j][0][k] = vertex[j-1][0][k];
    vertex[j][2][k] = point[0][k];
    normal[j][k] = pn[0][k];
    gn[j]=201;
}       //7th else ends
j++;
}      //6th if ends

jmax2=j; //max number of triangles with newly created ones at inlet, jmax2>jmax>imax

//........OUTPUTS......................
output = fopen("mart_red.stl","w");
//bu sirayi sakin bozma, harpoon dosyasinda da sira bozuluyor
//generation 0.........................
fprintf(output, "solid WALL_00
"); for(j=jmax; j<jmax2; j++){ if(gn[j]==0){ fprintf(output," facet normal ");    fprintf(output,"%e %e %e
",normal[j][0], normal[j][1], normal[j][2]); fprintf(output," outer loop\n");    for(k=0; k<3; k++){ fprintf(output, " vertex ");    fprintf(output,"%e %e %e
",vertex[j][k][0], vertex[j][k][1], vertex[j][k][2]); fprintf(output," endloop\n endfacet\n");    } } for(i=0; i<imax; i++){ if(gn[i]==0){ fprintf(output," facet normal ");    fprintf(output,"%e %e %e\n",normal[i][0], normal[i][1], normal[i][2]); fprintf(output," outer loop\n");    for(k=0; k<3; k++){ fprintf(output, " vertex ");    fprintf(output,"%e %e %e\n",vertex[i][k][0], vertex[i][k][1], vertex[i][k][2]); fprintf(output," endloop\n endfacet\n");    } } fprintf(output, "endsolid\n"); //........................................
// generations 1-4.....................
for(l=1; l<NB; l++){ fprintf(output, "solid WALL_0%d\n",l); for(i=0; i<imax; i++){ if(gn[i]==l){

}}
fprintf(output," facet normal ");
    fprintf(output," %e %e %e\n",normal[i][0], normal[i][1], normal[i][2]);
fprintf(output," outer loop\n");
for(k=0; k<3; k++){
    fprintf(output," vertex ");
    fprintf(output," %e %e %e\n",vertex[i][k][0], vertex[i][k][1], vertex[i][k][2]);
    fprintf(output," endloop\n endfacet\n");
    fprintf(output," endsolid\n"); //.................................
}

//........................................
fprintf(output, "solid WALL_0%d\n",NB);
for(i=0; i<jmax; i++){
    if(gn[i]==100 || gn[i]==NB ){
        fprintf(output," facet normal ");
            fprintf(output," %e %e %e\n",normal[i][0], normal[i][1], normal[i][2]);
            fprintf(output," outer loop\n");
            for(k=0; k<3; k++){
                fprintf(output," vertex ");
                    fprintf(output," %e %e %e\n",vertex[i][k][0], vertex[i][k][1], vertex[i][k][2]);
                    fprintf(output," endloop\nendfacet\n");
            }
        fprintf(output," endsolid\n"); //.................................
    }
if(noutlet<10){
    fprintf(output," solid PRESSURE_0%d\n",noutlet);
    //........................................
    fprintf(output, "solid WALL_0%d\n",NB);
    for(i=0; i<jmax; i++){
        if(gn[i]==100 || gn[i]==NB ){
            fprintf(output," facet normal ");
                fprintf(output," %e %e %e\n",normal[i][0], normal[i][1], normal[i][2]);
                fprintf(output," outer loop\n");
                for(k=0; k<3; k++){
                    fprintf(output," vertex ");
                        fprintf(output," %e %e %e\n",vertex[i][k][0], vertex[i][k][1], vertex[i][k][2]);
                        fprintf(output," endloop\nendfacet\n");
               }
            fprintf(output," endsolid\n"); //.................................
        }
    }
    //........................................
    fprintf(output," solid PRESSURE_00\n");
    for(j=jmax; j<jmax2; j++){
        if(gn[j]==201){
            fprintf(output," facet normal ");
                fprintf(output," %e %e %e\n",normal[j][0], normal[j][1], normal[j][2]);
                fprintf(output," outer loop\n");
                for(k=0; k<3; k++){
                    fprintf(output," vertex ");
                        fprintf(output," %e %e %e\n",vertex[j][k][0], vertex[j][k][1], vertex[j][k][2]);
                        fprintf(output," endloop\nendfacet\n");
               }
            fprintf(output," endsolid\n"); //.................................
        }
    }
    //........................................
    fprintf(output," solid PRESSURE_0%d\n",noutlet);
    //........................................
    fprintf(output, "solid WALL_0%d\n",NB);
    for(i=0; i<jmax; i++){
        if(gn[i]==100 || gn[i]==NB ){
            fprintf(output," facet normal ");
                fprintf(output," %e %e %e\n",normal[i][0], normal[i][1], normal[i][2]);
                fprintf(output," outer loop\n");
                for(k=0; k<3; k++){
                    fprintf(output," vertex ");
                        fprintf(output," %e %e %e\n",vertex[i][k][0], vertex[i][k][1], vertex[i][k][2]);
                        fprintf(output," endloop\nendfacet\n");
               }
            fprintf(output," endsolid\n"); //.................................
        }
    }
    //........................................
    fprintf(output," solid PRESSURE_00\n");
    for(j=jmax; j<jmax2; j++){
        if(gn[j]==201){
            fprintf(output," facet normal ");
                fprintf(output," %e %e %e\n",normal[j][0], normal[j][1], normal[j][2]);
                fprintf(output," outer loop\n");
                for(k=0; k<3; k++){
                    fprintf(output," vertex ");
                        fprintf(output," %e %e %e\n",vertex[j][k][0], vertex[j][k][1], vertex[j][k][2]);
                        fprintf(output," endloop\nendfacet\n");
               }
            fprintf(output," endsolid\n"); //.................................
        }
    }
    //........................................
    fprintf(output," solid PRESSURE_0%d\n",noutlet);
    //........................................
}
else{
    fprintf(output, "solid PRESSURE_%d",noutlet);
}

outletid[noutlet-1] = i;

for(j=imax; j<jmax; j++){
    if(gn[j]==200 & & endinfo[j][0]==i){ //.*.
        fprintf(output, " facet normal ");
        fprintf(output,"%e %e %e\n",normal[j][0], normal[j][1], normal[j][2]);
        fprintf(output, " outer loop\n"); for(k=0; k<3; k++){
            fprintf(output, " vertex ");
            fprintf(output,"%e %e %e\n",vertex[k][0], vertex[k][1], vertex[k][2]);
        } //.*.
        fprintf(output, "endsolid\n");
}
}

//..............................

output = fopen("edgeid_outlet.dat","w");
for(i=0;i<noutlet;i++) fprintf(output, "%d\n",outletid[i]);

//.......area calculation

} //end of main
#!/usr/bin/env python

# PYTHON SCRIPT OF LUNG PROJECT
# Developed by Gulkiz Dogan
# January 2007
# To run this script, you need followings in the directory of script:

# 1. Directory GULKIZ
# GEO_LUNG will run here
# - input files: 1) airway_rubber.stl --> rough stl of airways from Amira
#                 2) airway_skeleton_rubber.txt --> airway skeleton from Amira
#                 3) lung_conf.txt --> HARPOON configuration file
# - output file: 1) upper_rubber.stl --> it is truncated, has boundary conditions
#                 and it has holes

# 2. Directory HAWORTH
# - Harpoon and D. Haworth’s branch will run here
# - 5 Lobe stl & skeleton --> stl of lobes and
# - 5 * lobe_conf.txt --> HARPOON configuration file

# 3. Directory NPHASE

import commands, math, os, time

# ..........UPPER BRANCH GEOMETRY Geolung..........  
# change directory
os.chdir("GULKIZ")
os.system('gcc ./geotool.c -o ./geo -lm')
os.system('./geo')

# copy list of edge ids of pressure outlet for DH’s code
os.system('cp edgeid_outlet.dat ../HAWORTH/.

# run harpoon to obtain octree based mesh with hanging nodes in upper branches
os.system('harpoon -batch ./lung5bc_conf.txt')

# send the associated cobalt file to NPHASE directory
os.system('cp cobalt.inp ../NPHASE/.


Appendix C

Python Script: breath.py
#send the associated cobalt.bc file to NPHASE directory
os.system('cp cobalt.bc ../NPHASE/.

# come up back
os.chdir(os.pardir)

#.........LOWER BRANCH GEOMETRY Daniel Haworth's code ..........
# change directory
os.chdir("HAWORTH")
# run Harpoon for each lobe geometries
# and name output tetrahedral mesh geometries according to DHs code
for i in range(5):
    os.system('harpoon -batch ./lobe_conf'+str(i)+'.txt')
    os.system('mv ./abacus ./Cast.input_lob'+str(i))

os.system('./makeit')
os.system('./runit')

#There are tecplot output files in output directory
#but only one output is necessary for NPHASE
#send lower branch geometry file to NPHASE
os.system('cp Cast.output_gulkiz ../NPHASE/.

# come up back
os.chdir(os.pardir)

#.........NPHASE......................................
# change directory
os.chdir("NPHASE")

#number of processor & number of iterations
nproc = int(raw_input('enter number of processors'))
numit = int(raw_input('enter number of iterations'))
walltime = int(

# Runs fump with specified number of processors
input = file('./fump.input','w')
input.write(str(nproc) + '1')
input.close()
os.system('fump < fump.input')
os.system('rm ./fump.input')
# Deletes EnSight files (if they exist) in order to avoid errors
if os.path.exists('./en6.case'):
    os.system('rm ./en6.*')
if os.path.exists('./nphase_has_completed'):
    os.system('rm ./nphase_has*')

# Creates temporary backup file for nphase.dat
input = file('./nphase.dat','r')
output = file('./nphase.dat.temp','w')
for line in input:
    output.write(line)
input.close()
output.close()

# Edit run.nphase to incorporate correct number of processors
# Creates temporary backup file for run.nphase
input = file('./run.nphase','r')
output = file('./run.nphase.temp','w')
for line in input:
    output.write(line)
input.close()
output.close()

# Modifies temporary file and write to actual run.nphase file
input = file('./run.nphase.temp','r')
output = file('./run.nphase','w')
first_line = input.readline()
input.readline()
output.write(first_line + '#PBS -l nodes=' + str(numpro) + ':ppn=1
')
for line in input:
    output.write(line)
input.close()
output.close()

# Deletes temporary file run.nphase.temp
os.system('rm ./run.nphase.temp')

# Runs nphase
if numpro == 1:
    os.system('nphase')
else:
    os.system('submit run.nphase')
# Loop
while not os.path.exists('./nphase_has_completed'):
    if os.path.exists('./nphase_has_bombed'):
        print 'nphase has bombed!'
        break
    break

# Concatenates all resid.print files to master file in order to test for convergence
# Creates temporary backup file for original master file
input = file('./resid.print.master','r')
output = file('./resid.print.master.temp','w')
for line in input:
    output.write(line)
input.close()
output.close()

# Modifies temporary file and write to actual master file
input1 = file('./resid.print.master.temp','r')
input2 = file('./resid.print','r')
output = file('./resid.print.master','w')
for line in input1:
    output.write(line)
for line in input2:
    output.write(line)
input1.close()
input2.close()
output.close()

# Deletes temporary master file
os.system('rm ./resid.print.master.temp')

#time.sleep(10)

    # Runs emerge or egoldmerge with specified number of processors and fields
    input = file('./emerge.input','w')
    input.write(str(numpro) + '
1')
    input.close()
    os.system('egoldmerge < egoldmerge.input')
    os.system('rm ./emerge.input')

    if os.path.exists('./ensightfiles' + str(i) + '.tar.gz'):
        os.system('rm ./ensightfiles' + str(i) + '.tar.gz')
int(i)
# Creates files for transient analysis
if i < 10:
    os.system('cp ./en6.geo ./en6.geo.00000'+ str(i))
    os.system('cp ./en6.p00.Esca ./en6.p00.Esca.00000'+ str(i))
    os.system('cp ./en6.u00.Esca ./en6.u00.Esca.00000'+ str(i))
    os.system('cp ./en6.v00.Esca ./en6.v00.Esca.00000'+ str(i))
    os.system('cp ./en6.w00.Esca ./en6.w00.Esca.00000'+ str(i))
    os.system('cp ./en6.uvw00.Evec ./en6.uvw00.Evec.00000'+ str(i))
if (i >= 10) and (i < 100):
    os.system('cp ./en6.geo ./en6.geo.0000'+ str(i))
    os.system('cp ./en6.p00.Esca ./en6.p00.Esca.0000'+ str(i))
    os.system('cp ./en6.u00.Esca ./en6.u00.Esca.0000'+ str(i))
    os.system('cp ./en6.v00.Esca ./en6.v00.Esca.0000'+ str(i))
    os.system('cp ./en6.w00.Esca ./en6.w00.Esca.0000'+ str(i))
    os.system('cp ./en6.uvw00.Evec ./en6.uvw00.Evec.0000'+ str(i))
if (i >= 100) and (i < 1000):
    os.system('cp ./en6.geo ./en6.geo.000'+ str(i))
    os.system('cp ./en6.p00.Esca ./en6.p00.Esca.000'+ str(i))
    os.system('cp ./en6.u00.Esca ./en6.u00.Esca.000'+ str(i))
    os.system('cp ./en6.v00.Esca ./en6.v00.Esca.000'+ str(i))
    os.system('cp ./en6.w00.Esca ./en6.w00.Esca.000'+ str(i))
    os.system('cp ./en6.uvw00.Evec ./en6.uvw00.Evec.000'+ str(i))
if (i >= 1000) and (i < 10000):
    os.system('cp ./en6.geo ./en6.geo.00'+ str(i))
    os.system('cp ./en6.p00.Esca ./en6.p00.Esca.00'+ str(i))
    os.system('cp ./en6.u00.Esca ./en6.u00.Esca.00'+ str(i))
    os.system('cp ./en6.v00.Esca ./en6.v00.Esca.00'+ str(i))
    os.system('cp ./en6.w00.Esca ./en6.w00.Esca.00'+ str(i))
    os.system('cp ./en6.uvw00.Evec ./en6.uvw00.Evec.00'+ str(i))

# Creates zip file for transient analysis
int(i)
if os.path.exists('./ensightfiles_transient.tar.gz'):
    os.system('rm ./ensightfiles_transient.tar')
if os.path.exists('./ensightfiles_transient.tar.gz'):
    os.system('rm ./ensightfiles_transient.tar.gz')
o.system('tar -czvf ./ensightfiles_transient.tar en6.case en6.geo.??????
/en6.v00.Esca.?????? en6.w00.Esca.??????)
    # os.system('tar -czvf ./ensightfiles_transient.tar engold.case engold.geo.??????
/engold.p00. Esca.?????? engold.u00. Esca.?????? engold.uvw00. Esce.??????
/engold.v00. Esca.?????? engold.w00. Esca.??????)
    os.system('gzip ./ensightfiles_transient.tar')
#!/usr/bin/env python
import commands, math, cmath, string

# PYTHON code to calculate the analytical result for 2-D channel oscillating flow
# Written by Gulkiz Dogan
# Input = comp_num.dat comes from NPHASE run.
# Output = tecplot1.dat & tecplot2.dat --> Multi-zone xy-plot
# Add this lines to job script: chmod +x compna.py
#

#VARIABLES
pi = 3.141593
b, dpdz, rou, mu = 0.005, 1.0, 1.119, 1.9*10**-.5
omega = 2*pi/3.7
comega = ( rou*omega/mu )**0.5 * b
Pstar = ( b**2 / mu ) * dpdz
W1 = ( Pstar / comega**2 ) * 1j
W0 = 0.5 * Pstar

input = file('./comp_num.dat','r')
output1 = file('./tecplot1.dat','w')
output2 = file('./tecplot2.dat','w')
output1.write('TITLE = "Comparison of Numerical with Analytical, Multi-Zone XY Plot"\n')
output1.write('VARIABLES = "numerical", "analytical","Nondimensional Channel Height"\n')
output2.write('TITLE = "Comparison of Numerical with Analytical, Multi-Zone XY Plot"\n')
output2.write('VARIABLES = "numerical", "analytical","Nondimensional Channel Height"\n')
while 1:
    line = input.readline()
    (sil1, sil2, sil3) = string.split(line)
    if sil1 == 'EOF':
        break
    elif sil1 == 'time':
        t = float(sil3)
        if t<3.71:
output1.write('0 0 1'+'
')
output1.write('ZONE T="t='+str(t)+'s. ", I=22, f=point'+"'"+
')
output1.write('0 0 -1 '+'
')
else:
    output2.write('0 0 1'+'
')
    output2.write('ZONE T="t='+str(t)+'s. ", I=22, f=point'+"'"+
')
    output2.write('0 0 -1 '+'
')
    line = input.readline()
else:
    y = float(sil2)/b - 1.0
    U=W1*( 1.0 - cmath.cosh(1j**0.5 * comega*y) / cmath.cosh(1j**0.5 * comega) )
    fi=math.atan(U.imag/U.real)
    Amp=(U.real**2+U.imag**2)**0.5
    u_anly=Amp*math.cos(omega*t + fi)
    if t<3.71:
        output1.write(sil3 + ' ' + str(u_anly)+ ' ' + str(y) +'
')
    else:
        output2.write(sil3 + ' ' + str(u_anly)+ ' ' + str(y) +'
')
output1.write('0 0 1 '+'
')
output2.write('0 0 1 '+'
')
inpu.close()
output1.close()
output2.close()