UNVEILING ADVANCED MECHANISMS OF INHALABLE DRUG AEROSOL DYNAMICS USING COMPUTATIONAL FLUID DYNAMICS AND DISCRETE ELEMENT METHOD

By

JIANAN ZHAO

B.E. in Thermodynamics and Power Engineering
Southeast University
Nanjing, Jiangsu, China
2013

M.S. in Mechanical Engineering
University of Southern California
Los Angeles, CA, USA
2015

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
July, 2021

UNVEILING ADVANCED MECHANISMS OF

INHALABLE DRUG AEROSOL DYNAMICS USING

COMPUTATIONAL FLUID DYNAMICS AND DISCRETE

ELEMENT METHOD

Dissertation Approved:

Dr. Yu Feng

Dissertation Adviser

Dr. Jindal Shah

Dr. Ashlee N. Ford Versypt

Dr. Arvind Santhanakrishnan

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep gratitude for the effort and time that my advisor Dr. Y. Feng and committee members Drs. A. Ford Versypt, J. Shah, and A. Santhanakrishnan exerted for my research. Their guidance, ideas, and encouragement have been an invaluable source of inspiration and direction throughout this study. They shared their experiences and accomplishments with the perspective not attainable by a younger person like me. Thank you for all those candid, relaxed, and beneficial instructions and discussions not only on my Ph.D. research but also on my life planning. I would also like to thank other professors in Chemical Engineering and Mechanical Engineering who have offered me superb classes, exerted time and energy throughout my research ventures. I sincerely appreciate all the advice and encouragement that they provided.

My colleagues and previous colleagues in CBBL have provided invaluable suggestions, comments, and discussions throughout my research. I would like to acknowledge these individuals as Dr. Hang Yi, Mr. Ted Sperry, Miss Hamideh Hayati, and Mr. Ahmad Haghnegahdar.

Also, my Ph.D. research has richly benefited from multiple collaborations and supports of many different sources. I am grateful for the source code provided by Dr. Kenichiro Koshiyama's team for the construction of the virtual acinar model.

The research was made possible by funding through the award for project number HR19-106, from the Oklahoma Center for the Advancement of Science and Technology and the award provided by Cipla Inc. (Mumbai, India). The use of Rocky DEM (ESSS, Woburn, MA) as part of the ESSS-CBBL academic partnership is gratefully acknowledged (Dr. Rahul Bharadwaj). The use of Ansys software (Ansys Inc., Canonsburg, PA) as part of the Ansys-CBBL academic partnership is also gratefully acknowledged (Dr. Thierry Marchal). This material is based upon work supported by the National Science Foundation under grant no. CBET 2120688. This work used resources services and support provided via the COVID-19 HPC Consortium (https://covid19-hpc-consortium.org/), a unique private-public effort to bring together government, industry, and academic leaders who are volunteering free compute time and resources in support of COVID-19 research. Some computational jobs were performed at the High-Performance Computing Center, Oklahoma State University, supported, in part, through the National Science Foundation (Grant No. OAC-1531128).

Last but not least, I am indebted to my parents for providing me the opportunities and experiences that have made me who I am. I would like to thank my parents for their selfless love and encouragement.

Acknowledgements reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: JIANAN ZHAO

Date of Degree: JULY, 2021

Title of Study: UNVEILING ADVANCED MECHANISMS OF INHALABLE DRUG AEROSOL DYNAMICS USING COMPUTATIONAL FLUID DYNAMICS AND DISCRETE ELEMENT METHOD

Major Field: CHEMICAL ENGINEERING

Abstract: Capsule-based dry powder inhalers (DPIs) are widely used to treat chronic obstructive pulmonary disease (COPD) by delivering active pharmaceutical ingredients (APIs) *via* inhalation into human respiratory systems. Previous research has shown that the actuation flow rate, aerodynamic particle size distribution (APSD), and particle shape of lactose carriers are factors that can influence the particle deposition patterns in human respiratory systems. Understanding the dynamics of APIs transport in DPIs and airways can provide significant value for the design optimization of DPIs and particle shapes to enhance the delivery of APIs to the designated lung sites, i.e., small airways. Thus, it is necessary to investigate how to modulate the above-mentioned factors to increase the delivery efficacy to small airways and enhance the therapeutic effect to treat COPD. Compared with *in vitro* and *in vivo* methods, computational fluid-particle dynamics (CFPD) models allow researchers to study the transport dynamics of inhalable therapeutic dry powders in both DPI and human respiratory systems. However, existing CFPD models neglect particle-particle interactions, and most existing airway models lack peripheral lung airway and neglect the airway deformation kinematics. Such deficiencies can lead to inaccurate predictions of particle transport and deposition. This study developed a one-way coupled computational fluid dynamics (CFD) and discrete element method (DEM) model to simulate the particle-particle and particle-device interactions, and the transport of API-carrier dry powder mixtures with different shapes of carriers in a DPI flow channel. The influence of actuation flow rate (30 to 90 L/min) and particle shape (aspect ratio equals 1, 5, and 10) on lactose carrier dynamics in a representative DPI, i.e., Spiriva^{TM} Handihaler^{TM}, has been investigated. Subsequently, an elastic truncated whole-lung model has also been developed to predict particle transport and deposition from mouth to alveoli, with disease-specific airway deformation kinematics. Numerical results indicate that 90 L/min actuation flow rate generates the highest delivery efficiency of Handihaler, as approximately 26% API reaches the deep lung region. The elastic truncated whole-lung modeling results show that noticeable differences of predictions between static and elastic lung models can be found, which demonstrates the necessity to model airway deformation kinematics in virtual lung models.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Nomenclature

Acronyms	
ANDA	Abbreviated new drug application
API	Active pharmaceutical ingredient
APSD	Aerodynamic particle size distribution
BE	Bioequivalence
CFD	Computational fluid dynamics
DEM	Discrete element method
COPD	Chronic obstructive pulmonary disease
DE	Delivery efficiency
DF	Deposition fraction
DPI	Dry powder inhaler
FDA	Food and drug administration
GDUFA	Generic drug user fee amendments
H-M	Hertz-Mindlin
JKR	Johnson, Kendall, and Roberts
OIDP	Orally inhaled drug product
ROA	Route of administration
Symbols	
a	Radius of contact
A'	Projected particle area in the flow direction
C_D	Drag coefficient
C_H	Normal damping coefficient
d_p	Particle (volume equivalent) diameter

E^*	Effective Young’s Modulus
E	Young’s Modulus
$\vec{F}_c$	Contact force
$\vec{F}_{cn}$	Normal contact force
$\vec{F}_{cnadh}$	Adhesion force modeled by JKR cohesion
$\vec{F}_{cnd}$	Normal viscous damping force
$\vec{F}_{cne}$	Normal elastic force
$\vec{F}_{ct}$	Tangential contact force
$\vec{F}_{ct}$	Tangential force
$\vec{F}_{ctd}$	Tangential viscous damping force
$\vec{F}_{cte}$	Tangential spring force
$\vec{F}_{ctf}$	Frictional force
$\vec{F}_D$	Drag force
$\vec{F}_{fp}$	Forces acting on particles by fluid flow
$\vec{F}_g$	Gravitational force
$\vec{F}_L$	Lift force
$\vec{F}_{pf}$	Forces acting on fluid flow by particles
$\vec{F}_{VM}$	Added (virtual) mass force
$\vec{F}_{\nabla p}$	Pressure gradient force
$\vec{g}$	Gravitational acceleration
I	Moment of inertia tensor
K_n	Normal contact stiffness

$\vec{M}_c$	Torque generated by tangential forces
$\vec{M}_{fp}$	Torque generated by flow velocity gradient
m^*	Effective mass
m_p	Mass of particle
$\vec{n}$	Unit normal vector
p	Pressure
R^*	Effective radius
s_n	Normal contact overlap
$\dot{s}_n$	Time derivative of s_n
$\vec{s}_\tau$	Tangential relative displacement at the contact
$\dot{\vec{s}}_n$	Tangential component of the relative velocity at the contact
$s_{\tau,max}$	Threshold value of relative tangential displacement
t	Physical time
$\vec{u}_f$	Velocity of airflow
$\vec{u}_p$	Velocity of particle
V_c	Mesh cell volume
Greek symbols	
α_f	Volume fraction of air
Γ	Surface energy
ε	Coefficient of restitution
η	Damping ratio for the linear spring-dashpot model
η_H	Normal damping ratio for the Hertzian model
η_τ	Tangential damping ratio

μ_d	Dynamic friction coefficient
μ_p	Friction coefficient
μ_s	Static friction coefficient
ν_f	Kinematic viscosity of air
ρ_f	Density of air
ρ_p	Density of particle
σ	Poisson’s ratio
$\vec{\tau}_f$	Local stress tensor
$\overrightarrow{\omega}_p$	Angular velocity vector

CHAPTER I

INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

The administration of medicines via inhalation is one of the most popular treatments of pulmonary diseases, such as irreversible asthma and chronic obstructive pulmonary disease (COPD) (GBD, 2017). Capsule-based dry powder inhalers (DPIs) are widely used to deliver active pharmaceutical ingredients (APIs) attached to micron-sized carrier particles into human respiratory systems. However, such methods are ineffective since only a small fraction of the medication, i.e., less than 30%, can reach the designated small airways (G8 to alveoli) (Islam & Cleary, 2012; Kolanjiyil & Kleinstreuer, 2019). Understanding the dynamics of APIs transport in DPIs and airways can provide significant value for the design optimization of DPIs, and dry powder particle shapes and surface characteristics to enhance the delivery of APIs to the designated lung sites. Compared with *in vitro* and *in vivo* methods, computational models allow researchers to investigate the complex process of the transport and interactions of aerosolized dry powder medications with its unique capability to unveil the underlying physics with high-resolution “x-ray” vision inside the medical device and human respiratory systems. Specifically, computational fluid-particle dynamics (CFPD) can predict the airflow fields and the trajectories of embedded particles based on first principles. Compared with *in vitro* tests, numerical simulation is less expensive and time-consuming while providing accurate results. This allows researchers/engineers to reduce the experiment cost and development cycle duration of DPI product innovations.

Conventional computational fluid-particle dynamics (CFPD) models have been employed for decades to investigate particle-laden airflow transport phenomena in medical devices and human respiratory systems (Chen et al., 2017; Feng et al., 2016; Feng et al., 2015; Feng et al., 2017; Haghnegahdar et al., 2019; Koullapis et al., 2018; Koullapis et al., 2016; Koullapis et al., 2018). However, three major limitations exist for the conventional Euler-Lagrange models employed in the previous CFPD models:

(1) Particle shape factors are considered but simplified using equivalent diameter concepts. In the Euler-Lagrange model, the discrete particulate phase is assumed to be spherical point-mass particles with equivalent volume or Stokes diameters (Kleinstreuer & Feng, 2013). Such a simplification can lead to inaccurate particle trajectory predictions, especially when the particles are in anisotropic shapes, e.g., elongated fibers. Although shape factors are introduced to compensate for the drawback of the CFPD model mentioned above, the accuracy of the modified CFPD model is still questionable since the transient rotational motions of particles with anisotropic shapes are not predicted, which is a key mechanism to influence the magnitudes of drag and lift forces acting on particles, as well as the particle trajectories.

(2) Particle-particle interactions are neglected. During the drug powder release, transport, and emission inside DPIs, the particle volume and mass fractions are large in the airflow. Accordingly, particle-particle interaction forces, i.e., contact forces, the van der Waals, and electrostatic forces, play vital roles in the agglomeration/de-agglomeration between particles, the resultant emitted aerodynamic particle size distributions (APSDs) and emitted total particle mass entering the human mouth. The emitted APSDs and total particle mass can significantly impact delivered dose to the designated lung sites of lung disease treatment, i.e., small airways (G8-alveoli) for COPD treatment. Neglecting particle-particle interactions in the DPI flow channel may result in inaccurate predictions of the emitted APSDs and particle mass entering the human mouth, leading to the wrong prediction of localized drug deposition in the lung and the effectiveness of the inhalation therapy.

(3) Disease-specific airway deformation kinematics has not been extensively studied and integrated into whole-lung modeling. Existing modeling strategies for lung aerosol dynamics assume the airways are static, neglecting the dynamic rhythmical expansion and contraction movement of the airways during breathing. However, the airflow pattern and drug powder transport may be significantly altered by moving airway wall boundaries, which is pulmonary disease-specific. Thus, neglecting airway deformation kinematics limit the capability of existing CFD-based lung models of investigating the underlying aerosol dynamics of pulmonary-related problems.

To address the above-mentioned modeling efficiencies and knowledge gaps, the research objective of this study is to extend the CFPD model to explicitly consider the modeling of irregular-shaped particles, particle-particle interactions, as well as the physiologically realistic airway deformation kinematics in a whole-lung model. By employing the novel elastic whole-lung model and CFD-DEM model, this study investigated how particle shape, breathing pattern, and disease condition can influence the transport and deposition of therapeutic dry powders in a representative DPI and the whole lung from mouth to alveoli. The CFD-DEM based elastic whole-lung modeling framework will enable the capability of virtual lung models to simulate the pulmonary air-particle flow dynamics on a device-specific and disease-specific level.

1.2 Specific Aims

The research objective has been achieved by pursuing the following specific aims (see Figure 1):

Aim I: Quantify the particle shape effects on their agglomeration/de-agglomeration and their emitted APSDs from a representative DPI using CFD-DEM

Task 1.1: Develop, calibrate, and validate the CFD-DEM model for spherical and sphero-cylindrical particle-particle interactions.

Task 1.2: Determine the impacts of particle shape and actuation flow rate on particle-particle and particle-device interactions, as well as the API and lactose particle transport and deposition in a DPI flow channel and a human respiratory system.

Aim II: Determine the lung deposition of inhaled therapeutic particles in a newly developed elastic truncated whole-lung model with the physiologically realistic airway deformation

Task 2.1: Develop, calibrate, and validate a 3D elastic TWL model with disease-specific airway deformation kinematics.

Task 2.2: Determine how disease-specific airway deformation kinematics, breathing pattern, and particle size can influence the particle delivery efficiencies to small airways (G8-alveoli) for optimal COPD treatment.

Specifically, Aim I focuses on developing a CFD-DEM model to accurately simulate inter-particulate agglomeration and de-agglomeration between API and carrier (lactose) particles during the emission and transport of dry powder particles in a representative DPI, i.e., Spiriva^{TM} Handihaler^{TM} (SH). Our central hypothesis is that particle-particle interactions play a key role in the dynamics of particle transport, which can significantly influence the agglomeration/de-agglomeration process among particles. Hence, the APSD emitted from the DPI mouthpiece. Based on an extensive literature review, the working rationale is that the coupled CFD-DEM approach is a promising alternative to the CFPD model for modeling granular-fluid systems since it can capture the discrete nature of the particle phase while maintaining the computational tractability. This is accomplished by solving the airflow field at the mesh cell level instead of at the detailed particle level while tracing individual particles by solving Newton's second law. Instead of treating particles as spherical mass points in the CFPD model, the CFD-DEM approach considers each particle as an entity with a specific volume, which enables the detection of particle-particle contact (Hertz, 1882). Therefore, CFD-DEM can improve the understanding of the complex aerosol dynamics (i.e., dense particle suspensions) inside the flow channel by explicitly modeling the particle-particle interactions, especially for particles with irregular shapes. By considering the effect of actuation flow rate and inter-particulate agglomeration/de-agglomeration, APSD at the mouthpiece (mouth front) of the DPI can be obtained and used as the accurate and realistic inlet boundary conditions for lung aerosol dynamics simulations (see Task 1.2).

Aim II focuses on developing an innovative elastic truncated whole-lung (TWL) model, covering the entire conductive and respiratory zones of the human pulmonary system, including acinar. Here, the acinar refers to the gas-exchanging unit of the lung and is defined as that portion of the lung distal to the terminal bronchiole, which is composed of the respiratory bronchioles, alveolar ducts, alveolar sacs, and alveoli (McLoud & Aquino, 2010). Using the elastic TWL model with physiologically realistic anisotropic airway deformation, we numerically quantified the drug delivery efficacy in small airways by applying a clinically validated Euler-Lagrange model with a dynamic mesh method. The rationale is that the delivery efficacy calculated through numerical simulation using the elastic TWL model will be more physiologically realistic and disease-specific than existing lung models, which can assist in determining and optimizing the drug delivery strategy in an accurate, cost-effective, time-saving, and noninvasive manner.

1.3 Significance and Innovation

This study is significant and innovative since it can:

(1) Advance the scientific knowledge of pharmaceutical drug particle dynamics by enhancing the understanding of the underlying relationships between particle shape and size factors, inter-particulate forces that dominate particle fluidization, agglomeration, and dispersion;

(2) Identify the critical particle shape factors that can be optimized to increase DPI drug delivery efficacy to the small airways (G8 to alveoli);

(3) Be the first computational lung aerosol dynamics model predicting irregular-shaped particle interactions, transport behaviors, and deposition considering realistic lung deformations *via* computational fluid dynamics-discrete element method (CFD-DEM) coupled with elastic lung modeling method realized by one-way coupled fluid-structure interaction (FSI) techniques (i.e., dynamic mesh method); and

(4) Enlighten the understanding of the transport and interaction among drug particles in diseased lung airways leading to a more effective strategy of targeted drug delivery to treat pulmonary diseases.

1.4 Dry Powder Inhaler

Chronic lung diseases, such as asthma and chronic obstructive pulmonary disease (COPD), are causing a terrible toll on the health of individuals and becoming a growing concern globally (Lee et al., 2015). To treat chronic lung diseases, orally inhaled drug products (OIDPs) are widely used. As a primary type of medical device delivering OIDPs, DPIs deliver active pharmaceutical ingredients (APIs) via the inhalation route to treat asthma and/or COPD. Among different DPIs, carrier-based DPIs, e.g., SH and Advair^{TM} Diskus^{TM}, are preferable compared with other pulmonary drug delivery devices because of several advantages, such as high compliance with the patient, propellant-free actuation, formulation stability, and low risk of chemical degradation (Lewis et al., 2017). The DPI delivers an efficacious dose of API nanoparticles to designated sites of the lung, i.e., small airways with diameters less than 2 mm to treat COPD. Upon actuation via patient inhalation, a dosage of dry powder is entrained and dispersed in the inspiratory airflow by a variety of fluidization and dispersion mechanisms specific to each device. In addition, dry powders contain micron carrier particles (e.g., lactose) to increase API particle dispersion, which can benefit the drug delivery efficiency (DE) to peripheral lung.

As a result, the development and production of OIDPs are drawing more attention, especially in the generic area, as patients would benefit from appropriate and affordable medications/products. However, it is challenging to develop generic devices for OIDP delivery because of the complex design of the DPI and the challenge of establishing bioequivalence (BE) for a drug product that needs to be delivered to the site of action area in the lung (Lee et al., 2015). In 2017, the US Food and Drug Administration (FDA) published the Generic Drug User Fee Amendments (GDUFA) to enable reviewers to evaluate abbreviated new drug applications (ANDAs) more efficiently with an emphasis on regulatory science enhancements of complex drug products, including OIDPs (*Generic Drug User Fee Amendments (GDUFA)*, 2017). For most orally administered drugs that reach their sites of action through the systemic circulation, BE is demonstrated based on drug concentration in a relevant biologic fluid (e.g., plasma or blood). However, this approach is currently considered inadequate in the United States to establish BE of inhalation products intended for local action, such as pressurized metered dose inhalers (pMDIs) and DPIs that are used to treat lung diseases (e.g., asthma and COPD), since their drug delivery and intended action in the lung does not rely on the systemic circulation. Therefore, the demonstration of BE for these locally acting drug products is challenging.

Furthermore, effective inhalation therapy using DPIs depends highly on the emitted APSDs and total mass of the API from the DPI mouthpieces, with a preferable diameter between 1 and 5 μm (Lewis et al., 2017). Thus, providing accurate predictions of emitted APSDs from DPIs and the resultant lung deposition of OIDPs is essential as the first step to demonstrating the equivalence between different designs of DPIs. However, achieving equivalence in emitted APSDs and lung depositions is also challenging since they are highly related to DPI performance, which is a function of interactions between the patient, device, and drug particle characteristics. Specifically, the de-agglomeration and agglomeration between APIs and carriers need to be well understood and controlled, since they are the key mechanisms to determine the emitted APSD. Therefore, new insights for DPI product developments and BE assessments are critically needed, which requires support from high-resolution particle dynamics data provided by reliable numerical models in a cost-effective and time-saving manner. In addition, the difficulty in acquiring data for BE demonstration using *in vitro* and *in vivo* methods triggers the need to develop a reliable computational model to provide high-resolution *in silico* evidence of air-particle flow dynamics both in the DPI flow channel and in virtual human respiratory systems.

1.5 Role of Numerical Modeling in Therapeutic Particle Transport Dynamics

Fulfilling such knowledge gaps of BE assessments is difficult if using experiments only. This is due to the limits in operational flexibility, imaging resolution, and research expenditure. As an alternative, computational fluid dynamics (CFD) coupled with discrete element method (DEM) has been employed, which can capture the interactions between particles and explicitly track the dry powder particle trajectories in the DPI and the human respiratory systems (Benque & Khinast, 2021; Mitani et al., 2020; Ponzini et al., 2021; Tong et al., 2017; Tong et al., 2016; Yang et al., 2015). Specifically, Tong et al. (2016) employed the CFD-DEM approach to investigate the underlying mechanisms of loose drug agglomeration using a T-shape pipe. Tong et al. (2017) also studied the aerosolization mechanism of carrier-based formulations using the CFD-DEM approach. However, both studies used oversimplified drug-device geometry, which does not consider the device design effect on the particle interaction mechanisms. Mitani et al. (2020) investigated the drug particle adhesion mechanism based on a combined CFD-DEM approach but only focused on the capsule region (design) of a DPI. Ponzini et al. (2021) discussed the possibility of validating a coupled CFD-DEM model for the NextHaler® DPI device by comparing it with experimental data. However, the APSD data is lacking from the study. Benque and Khinast used CFD-DEM approach to estimate the flow-induced effect on APSD dispersion in a specific DPI device, i.e., Aerolizer®, but the simulated particle number was reduced to 10,000 from the real-world total particle number due to the insufficient computational cost. As a result, the effect of APSD on the drug delivery efficiency (DE) in the lung was not investigated. Although a few studies attempted to develop CFD-DEM models to predict agglomeration and de-agglomeration of carriers and APIs in other DPIs (Yang et al., 2015), there are no DEM studies on SH. There is still a lack of an "all-in-one" numerical modeling framework that can predict the emitted APSDs and the resultant lung deposition. Therefore, the objective of Chapter 2 is to provide the "all-in-one" modeling tool to unveil the connections between DPI design, drug particle characteristics (shape and size effects), human factors, and the drug delivery efficiency to specific lung regions, i.e., from generation 8 (G8) to alveoli.

Another reason for using the CFD-DEM model is to study the particle shape effect on the drug powder DE of DPIs in human lung, since the particle shape engineering approach is a promising way to enhance the DE of APIs and the DEM model has the advantages of capturing the particle shape effect on particle-particle interaction over other models, such as discrete phase model (DPM). Previous studies have demonstrated that elongated particles are more likely to reach lower airways (Feng & Kleinstreuer, 2013). Thus, controlling particle shape is a feasible way to increase small-airway depositions of orally administrated medications. Also, with the progress in particle shape engineering for pulmonary drugs, customized shapes can now be produced to meet medication customization demands (Chow et al., 2007; Zellnitz et al., 2019). Therefore, it is beneficial to investigate how particle shape factors can be used to control particle aerodynamics and achieve targeted delivery to small airways.

Among various DPIs, we choose to model SH (Tiotropium Bromide), considering its representativeness in design and the time limit of the project duration. Specifically, SH is a typical design of single-dose and capsule-based devices, which cover a broad range of flow resistance (Delvadia et al., 2016). Chapter 2 focused on predicting and comparing the emitted APSDs and resultant lung deposition patterns using SH with different breathing patterns and particle shapes. An experimentally calibrated and validated CFD-DEM method was employed to predict agglomeration and de-agglomeration of lactose carriers and APIs. CFD-DEM simulations were performed at steady actuation flow rates of 30, 39, 60, and 90 L/min, which are relevant to drug delivery applications (CDER, 2017). Other than particles in spherical shape, simulations were also performed for sphero-cylindrical particles with an aspect ratio (AR) of 2, 5, and 10. This study provides an enhanced understanding of the fundamental carrier-API interactions in DPIs, and the effect of drug particle characteristics and DPI flow channel designs on the drug delivery efficiency to the G8-to-alveoli region. The "all-in-one" modeling framework developed in this study has the potential to numerically generate the *in vitro*-lung deposition correlations, reduce the cost of generic product innovations, and accelerate the generic product review and approval.

1.6 Drug Particle Transport and Deposition Simulations in Whole-lung Models

OIDPs are frequently used in inhalation therapy to treat lung diseases such as asthma, COPD, and cystic fibrosis (Labiris & Dolovich, 2003). The goal of inhalation therapy is to deliver sufficient medication dose to designated lung sites, which are mostly small airways (i.e., airways with diameters less than 2mm). Such respiratory system configurations that cover mouth/nose to alveoli are also called whole-lung airway models. As a result, to accurately predict the absorption and translocation of the drug particles, regional deposition data must be precisely measured. Hence, it is crucial to determine drug aerosol deposition patterns in respiratory systems containing small airways from G8 to alveoli. Furthermore, quantifying the regional deposition of drug particles in designated lung sites can not only provide evidence of delivery dosage of the drugs and the treatment effectiveness, but also provide accurate regional deposition data as input for physiological based pharmacokinetic (PBPK) models (Haghnegahdar et al., 2019). However, measuring the regional deposition in the human whole-lung using *in vitro* and/or *in vivo* studies is challenging due to the limitations of imaging resolutions, resolutions of 3D airway replica printing, operational flexibilities, and possible radiation exposure hazard to human volunteers (Martin et al., 2018; Walenga et al., 2019).

As an alternative, *in silico* models can simulate and visualize particle aerodynamic behavior in human respiratory systems in cost-effective and time-saving manners (Feng et al., 2016; Feng et al., 2017). Compared with *in vitro* and *in vivo* methods, computational models allow researchers to investigate the complex process of the transport and deposition of respirable aerosols with its unique capability to unveil the underlying physics with high-resolution "x-ray" vision inside the human respiratory systems. Due to the geometrical complexity and high Reynolds numbers (Re), especially at flow rates relevant to drug delivery via oral administration, airflow in the upper airways usually transitions to turbulence. The most common numerical approach, i.e., computational fluid-particle dynamics (CFPD) models, can capture the laminar-to-turbulence pulmonary airflows accurately and predict the trajectories of embedded particles based on the first (Feng et al., 2018; Zhao et al., 2020).

Although CFPD models have been extensively validated and widely used for decades to investigate particle-laden airflow transport phenomena in human respiratory systems (Feng et al., 2016; Feng et al., 2015; Feng et al., 2018; Koullapis et al., 2018; Koullapis et al., 2018), only a few studies simulated the transport and deposition using whole-lung models. This lack of study is mainly due to two reasons: (i) Reconstruction of small airway geometries is difficult due to the insufficient resolutions of medical images (Conway et al., 2013; Newman et al., 2012); (ii) The computational cost would increase exponentially as the deep lung region (up to alveoli) needs to be simulated. Several recent studies were carried out to address these limitations by developing whole-lung models (Cui et al., 2020; Hasler et al., 2019; Kolanjiyil & Kleinstreuer, 2016; Kolanjiyil & Kleinstreuer, 2017; Koullapis et al., 2020; Longest et al., 2016; Poorbahrami et al., 2021; Taulbee & P., 1975; Tawhai et al., 2000; Tena et al., 2017; Yin et al., 2013; Zhang et al., 2008). Existing whole-lung research efforts include: (1) using trumpet geometry to represent the whole TB tree (Cui et al., 2020; Hasler et al., 2019; Kolanjiyil & Kleinstreuer, 2016; Poorbahrami et al., 2021; Taulbee & P., 1975; Tawhai et al., 2000; Yin et al., 2013; Zhang et al., 2008), and (2) extending 3D upper airways to deep lungs and conducting CFD modelings by truncating airways and applying advanced coupled boundary conditions (Kolanjiyil &

Kleinstreuer, 2017; Koullapis et al., 2020; Longest et al., 2016; Tena et al., 2017). Specifically, Kolanjiyil and Kleinstreuer (2016) developed a whole-lung airway model by combining a basic 3D mouth-to-trachea geometry with an exponentially expanding 1D conduit, i.e., trumpet model. The different inhalation/exhalation profiles can be achieved by controlling the displacement of the bottom wall of the model. They also demonstrated the accuracy of their model for predicting the total and regional particle deposition results. Poorbahrami et al. (2021) established a whole-lung model to estimate age-dependent particle dosimetry by coupling CFD upper airway model with an adapted one-dimensional (1D) model based on the trumpet model. In the 1D regions, the diffusive and advective terms are used to consider the distal airway branching structure and inhaled air convection effect, respectively. Such a model enables the prediction of regional particle fate in the lungs by using a multi-domain method. Although a multi-scale model requires relatively low computational cost, the details of small airways structure including alveoli, are neglected. With the capabilities of CFD computing evolved over time, developing a 3D whole-lung airway model becomes feasible. In 2016, Longest et al. (2016) extended airways to lobar bronchi coupled with Stochastic Individual Path (SIP) approximations of bronchioles to predict the deposition of aerosol emitted from the dry powder inhaler. Koullapis et al. (2020) constructed a 3D deep lung model covering the 15 most distal lung generations to study the airflow patterns at quiet/deep breathing and the gravity effect on regional deposition.

1.7 Anisotropic and Disease-specific Airway Deformation Kinematics

Another major limitation in most numerical research regarding pulmonary simulation (Koullapis et al., 2018; Koullapis et al., 2016; Koullapis et al., 2018) is that airway deformation kinematics is not considered. Most existing virtual lung models assume the airways are “rigid” (Chen et al., 2017; Feng & Kleinstreuer, 2013; Feng & Kleinstreuer, 2013; Feng & Kleinstreuer, 2014; Feng et al., 2015; Feng et al., 2016; Tian et al., 2016), which is not physiologically realistic. Indeed, during the inhalation-exhalation cycles, glottis is opening and closing while the tracheobronchial (TB) tree is expanding and contracting (Zhao et al., 2020). Moreover, lung diseases can also alter airway deformation kinematics. For example, losses of lung expansion and contraction capability are commonly diagnosed in multiple obstructive lung diseases such as COPD (Dutta, 2016; Xi et al., 2018). Such losses significantly limit the delivery of inhaled therapeutic particles in nanoscale and microscale to distal airways as the designated sites for treatment (De Boer et al., 2017). Therefore, neglecting the physiologically realistic airway deformation in the previous research can lead to errors in the predictions of air-particle transport phenomena and disable the capability to predict the influence of disease-specific airway deformation kinematics on pulmonary air-particle flow structures. Recovering the real-time disease-specific anisotropic lung deformation in a whole-lung model from mouth to alveoli is necessary to reflect the physiologically realistic disease lung conditions and its effect on the inhaled particle transport and deposition. Research efforts have been made to address the modeling deficiencies by developing models to capture the deformation in certain regions (Aghasafari & Pidaparti, 2018; Comerford et al., 2010; Heravi et al., 2016; Hofemeier & Sznitman, 2016; Kolanjiyil & Kleinstreuer, 2017; Malvè et al., 2011; Seyfi et al., 2016; Seyfi Noferest et al., 2018; Subramaniam et al., 2017; Sul et al., 2019; Talaat & Xi, 2017; Wall & Rabczuk, 2008; Wang et al., 2019; Werner et al., 2008; Xi et al., 2018; Xi et al., 2018; Xia et al., 2010), i.e., alveolar movement, bronchioles movement, trachea-to-bronchi movement, and uvula motion. However, there is no existing research that modeled the particle deposition in the whole lung (from mouth/nose to alveoli) with anisotropic deformation kinematics representing disease-specific airway opening and stiffness.

To overcome such a drawback and enable simulating physiologically realistic lung deformation, Chapter 3 developed a truncated whole-lung airway model to enable the simulation of the inhaled particle transport simulation simultaneously with the transient anisotropic airway expansion and contraction in the entire TB tree. Numerical investigations are carried out on the particle size and biomechanical properties of the airways effects on the regional deposition in a truncated whole-lung airway model. Airways at three deformed levels representing healthy lung and diseased lung at two different COPD stages were simulated. Specifically, mild COPD represents GOLD I stage, and severe COPD represents GOLD III stage. Particles with aerodynamic diameters of 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, and 10.0 μm are modeled to reflect the typical range of OIDP APSDs from 100 nm to 100 μm (Chandel et al., 2019). The objective of Chapter 3 is to (i) quantify how the changes in real-time airway deformation kinematics alter the pulmonary airflow features (i.e., laminar-to-turbulence transition, and relaminarization) and particle distribution, thereby influencing the particle trajectories and deposition sites in the lung; and (ii) evaluate the modulated particle size to overcome the significant drug loss due to the upper airway deposition caused by turbulence dispersion, inertial impaction, and interception with the moving airway boundaries, thereby enhancing the particle delivery efficiency to distal airways that have undergone the loss of lung expansion and contraction capability. The key contributions of Chapter 3 are (i) the development of an innovative elastic truncated whole-lung (TWL) model, covering the entire conductive and respiratory zones of the human pulmonary system, including heterogeneous alveolar structure, (ii) the calibration and validation of the elastic TWL model to capture the airway deformation kinematics for the different COPD lung conditions, (iii) the quantitative determination of how disease-specific airway deformation kinematics influence the pulmonary air-particle transport and deposition patterns. The results of the delivery efficacy calculated through numerical simulation will assist the determination and optimization of the drug delivery strategy and particle engineering process.

CHAPTER II

PARTICLE TRANSPORT DYNAMICS IN DPI AND AIRWAYS

2.1 Theory

2.1.1 Geometry and mesh

The flow channel of the SH was reconstructed with capsule and grid included (see Fig. 1). The flow channel consists of the following: (i) a circular air inlet with 3.4 mm diameter for SH; (ii) capsule chamber with a diameter of 7.5 mm and a length of 17.8 mm along the flow direction; (iii) a grid designed for separating the particle bulk flow (see Fig. 2.1 for geometry details); (iv) an extending tube with a diameter of 5.4 mm; and (v) an elliptic-shaped orifice as outlet connecting to oral cavity with an area of 76.3 mm^2. The length (from the actuation air inlet to the orifice connecting to the mouth) of SH DPI is 57.5 mm. A 5 mm diameter and 15 mm length capsule was positioned at the center of the capsule chamber.

The 3D airways geometry employed in this study was modified based on the geometries used in Zhang et al. (2012). A detailed description of the revised geometry and the corresponding mesh can be found in a previous study by Hayati et al. (2021). An overview of the lung geometry and corresponding mesh is shown in Fig. 2.3. CFD meshes were generated for the DPI flow channels using Ansys Fluent Meshing 2020 R2 (Ansys Inc., Canonsburg, PA). Meshes consist of polyhedral cells with prism layers on walls to ensure $y^+ < 1$. The final mesh of the DPI flow channel (see Fig. 2.1) has a total of 3,732,269 cells, based on the mesh independence test.

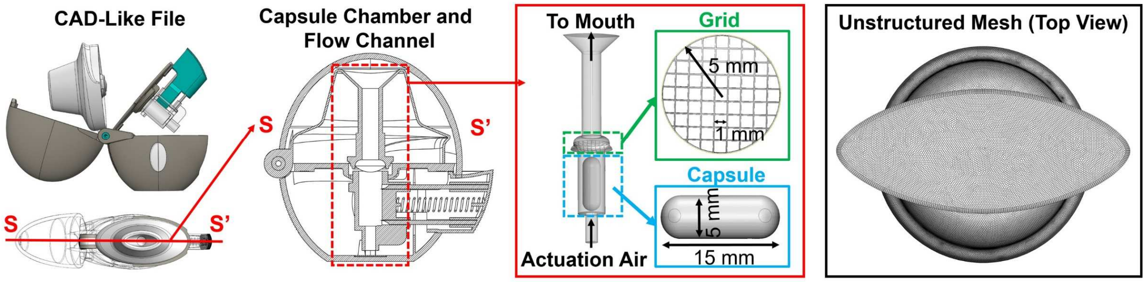

Figure 2.1: Reconstructed Spiriva^{TM} Handihaler^{TM} (SH) dry powder inhaler (DPI) device geometry and the hybrid polyhedral mesh, including the flow channel, grid, and capsule

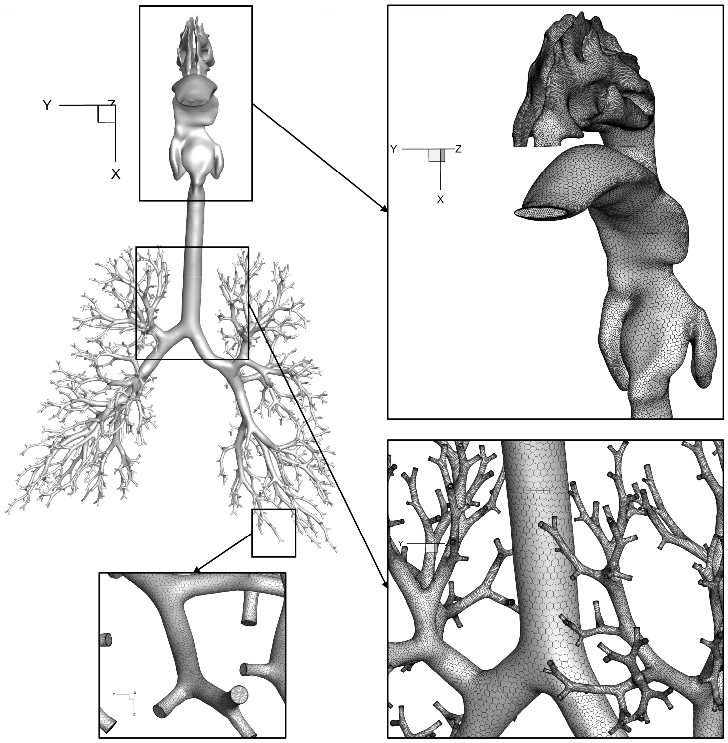

Figure 2.2: Geometry and polyhedral mesh with prism layers of the human respiratory system employed in Chapter 2

2.1.2 Governing Equations

Accurately predicting the emitted APSDs from the inhalers requires the knowledge of the particle-particle and particle-wall interactions during the drug transport simulations (Feng, 2013; Feng & Kleinstreuer, 2013; Feng & Kleinstreuer, 2014). To address such modeling demands, a generalized one-way coupled CFD-DEM method (Feng & Kleinstreuer, 2013; Feng & Kleinstreuer, 2014; Zhao, 2020) with the Hertz-Mindlin (H-M) JKR cohesion model (Jiang et al., 2018; Johnson et al., 1971) has been calibrated and validated. Compared with other numerical methods, the validated CFD-DEM method can accurately predict the particle agglomeration and de-agglomeration by considering particle-particle interactions, and the resultant emitted APSDs. In terms of air-particle flow dynamics modeling in the respiratory system, considering its particle-laden characteristic and the turbulence induced by high flow rates relevant to DPI application, the previous validated Euler-Lagrange model, i.e., the computational fluid dynamics-discrete phase model (CFD-DPM) (Feng et al., 2018), is employed to simulate the particle transport in the human respiratory system. Specifically, the turbulent airflow is simulated using the Reynolds averaged Navier-Stokes (RANS) approach considering its computational efficiency and reasonable accuracy when compared with the large eddy simulation (LES) approach. In terms of particle phase, each particle is tracked using the Lagrangian particle tracking method (Feng et al., 2018). Basically, the particle trajectory and velocity are calculated by solving Newton’s second law (Feng et al., 2018). Forces acting on the particles considered in this study are drag force, gravitational force, Brownian motion induced force (Feng et al., 2018; Longest et al., 2019). The validation of the application of our customized Lagrangian particle-tracking model in the oral/nasal cavities and TB tree has been well documented in peer-reviewed papers (Feng & Kleinstreuer, 2013; Feng et al., 2016; Feng et al., 2015; Feng et al., 2020; Feng et al.; Haghnegahdar et al., 2018; Haghnegahdar et al., 2019; Zhao et al., 2019; Zhao et al., 2020; Zhao, 2020).

2.1.2.1 Continuous Phase

The governing equations are summarized as follows. In the Euler-Lagrange approach, the air is treated as a continuous phase by solving continuity and Navier-Stokes equations simultaneously, while the dispersed phase is solved by tracking a large number of particles through the flow field. Each particle or group of particles is individually tracked along with the fluid phase by the result of forces acting on them by numerically integrating Newton's equations that govern the translation and rotation of the particles (Drew, 1983). Governing equations for the air-particle mixture are summarized in Eqs. (2.1) and (2.2),

$$\frac{\partial\left(\alpha_f \rho_f\right)}{\partial t} + \nabla \cdot \left(\alpha_f \rho_f \vec{u}_f\right) = 0 \tag{2.1}$$

$$\frac{\partial\left(\alpha_f \rho_f \vec{u}_f\right)}{\partial t} + \nabla \cdot \left(\alpha_f \rho_f \vec{u}_f \cdot \vec{u}_f\right) = -\alpha_f \nabla p + \nabla \cdot \left(\alpha_f \vec{\tau}_f\right) + \alpha_f \rho_f \vec{g} + \vec{F}_{pf} \tag{2.2}$$

where α_f is the air volume fraction, $\vec{\tau}_f$ is the local stress tensor, and $\vec{F}_{pf}$ represents the volumetric source term of momentum from interaction with the particulate phase, calculated according to the expression:

$$\vec{F}_{pf} = -\frac{\sum_{p=1}^{N} \vec{F}_{fp}}{V_c} \tag{2.3}$$

where, V_c is the computational cell volume, N is the number of particles inside the computational cell, and $\vec{F}_{fp}$ accounts for the forces generated by the fluid on the particles, such as drag force $\vec{F}_D$, pressure gradient force $\vec{F}_{\nabla p}$, added (virtual) mass force $\vec{F}_{VM}$, lift force $\vec{F}_L$, etc:

$$\vec{F}_{fp} = \vec{F}_D + \vec{F}_{\nabla p} + \vec{F}_{VM} + \vec{F}_L + \vec{F}_{others}. \tag{2.4}$$

The majority of these forces in Eq. (2.4) can be ignored. Specifically, since the density between fluid and particles is high ($\rho_f \gg \rho_p$), $\vec{F}_{VM}$ and $\vec{F}_L$ can be neglected. In addition, since the particle size is much smaller than the mesh cell size, $\vec{F}_{\nabla p}$ is negligible.

For solid spherical particles, the drag force, $\vec{F}_D$, acting on the particle is calculated using the definition of the drag coefficient C_D (Pritchard & Mitchell, 2011):

$$\vec{F}_D = \frac{1}{2}\frac{C_D}{C_{slip}}\rho_f A' \left|\vec{u}_f - \vec{u}_{p,j}\right|\left(\vec{u}_f - \vec{u}_{p,j}\right) \tag{2.5}$$

where $(\vec{u}_f - \vec{u}_{p,j})$ is the relative velocity between particle and fluid, C_{slip} is the Cunningham correction factor (Clift et al., 2005) to account for non-continuum effects for drag force on small particles with the increase in Knudsen number (*Kn*), A' is the projected particle area in the flow direction, and C_D is the particle drag coefficient (Clift et al., 2005). For spherical particles, C_D can be estimated according to the Schiller & Naumann drag correlation for spherical particles (Clift & Gauvin, 1971)

$$C_D = \begin{cases} 24(1 + 0.15Re_p^{0.687})/Re_p & for\ \ 0.0 < Re_p \leq 1000 \\ 0.44 & for\ \ 1000 < Re_p \end{cases} \tag{2.6 a \& b}$$

where $Re_p = \left|\vec{u}_f - \vec{u}_p\right| d_p/\nu_f$ is the particle Reynolds number.

For particles with sphero-cylinder shape, Marheineke & Wegener drag law (Marheineke & Wegener, 2011) is applied. Among existing drag laws for elongated particles (Ganser, 1993; Haider & Levenspiel, 1989; Marheineke & Wegener, 2011), Marheineke & Wegener's drag law is not based on Eq. (2.5), since it considers the normal and tangential components of the drag separately in relation to the cylinder axis. Marheineke & Wegener drag law is derived based on slender body theory instead of solely correlation. Therefore, the derived aerodynamic force concept for a general drag model is valid for all Reynolds number regimes and incident flow directions. The formula for calculating the drag force over a sphero-cylinder (see Fig. 2.3) is

$$\vec{F}_{D,s} = \rho_f d_p u_n^{rel}\left(C_{D,n} u_n^{rel}\hat{n} - C_{D,\tau} u_\tau^{rel}\hat{\tau}\right) \tag{2.7}$$

where $\hat{n}$ is the unit vector perpendicular to the sphero-cylinder axis and $\hat{\tau}$ is the unit vector parallel to the sphero-cylinder axis. $C_{D,n}$ and $C_{D,\tau}$ are the drag coefficients for the normal and tangential directions, respectively. Both coefficients are functions of the Reynolds number based on the normal component of the velocity, defined as

$$Re_n = u_n^{rel} d_p / \nu_f \tag{2.8}$$

The relative velocity $\vec{u}^{rel}$ is defined as

$$\vec{u}^{rel} = \vec{u}_f - \vec{v}_p \tag{2.9}$$

where $\vec{v}_p$ is the velocity of the particle geometric center. As shown in Fig. 2.3, the tangential and normal components of $\vec{u}^{rel}$ can be calculated by

$$u_\tau^{rel} = \vec{u}^{rel} \cdot \hat{\tau} \tag{2.10}$$

$$u_n^{rel} = \|\vec{u}^{rel} - \vec{u}_\tau^{rel}\| \tag{2.11}$$

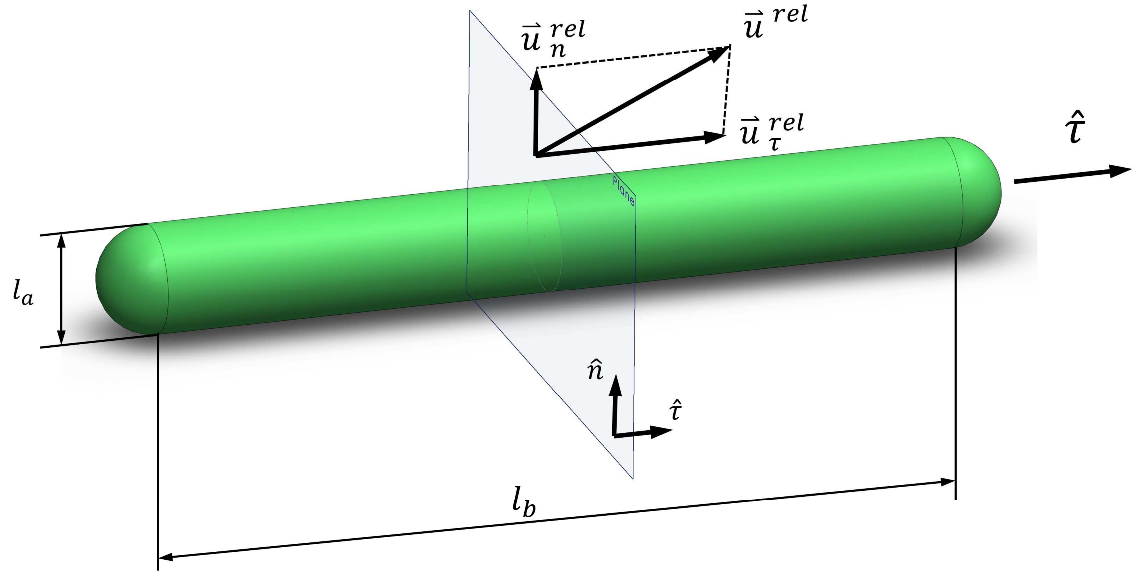

Figure2.3: Schematic depiction of parameters considered in the Marheineke & Wegener drag law

For different flow regimes (different Re_n), coefficients $C_{D,n}$ and $C_{D,\tau}$ are calculated using either analytical methods or numerical simulations. To unify the expressions of drag force, resistance coefficients are defined as

$$r_{D,\gamma} = Re_n C_{D,\gamma} \qquad \gamma = n, \tau \tag{2.12}$$

Accordingly, Eq. (2.8) can be reduced to:

$$\vec{F}'_{D,s} = \mu_f \left(r_{D,n} u_n^{rel} \hat{n} - r_{D,\tau} u_\tau^{rel} \hat{\tau} \right) \tag{2.13}$$

The correlations provided by Marheineke & Wegener (see Eq. (2.8)) can be then rewritten as

$$r_{D,\gamma} = \begin{cases} \sum_{k=0}^{3} S_{\gamma,k} Re_n^k & Re_n \leq a_0 \\ \Psi_\gamma(Re_n) & a_0 < Re_n \leq 0.1 \\ Re_n exp\left(\sum_{k=0}^{3} a_{\gamma,k} \ln^k Re_n\right) & 0.1 < Re_n \leq 100 \\ \Theta_\gamma(Re_n) & Re_n \geq 100 \end{cases} \tag{2.14 a, b, c & d}$$

where a_0 is determined by constructing Eqs. (2.14 a) and (2.14 b) continuously differentiable.

Table 2.1: Coefficient values for Eq. (2.14 c).

k	$a_{n,k}$	$a_{\tau,k}$
0	1.6911	1.1552
1	-6.7222×10^{-1}	-6.8479×10^{-1}
2	3.3287×10^{-2}	1.84884×10^{-2}
3	3.5015×10^{-3}	7.4966×10^{-4}

The functions Ψ_γ and Θ_γ are:

$$\Psi_\gamma(Re_n) = \begin{cases} \frac{4\pi}{S}\left(1 - \frac{S^2 - \frac{1}{2}S - \frac{5}{16}}{32S} Re_n^2\right) & \gamma = n \\ \frac{4\pi}{2S-1}\left(1 - \frac{sS^2 - 2S + 1}{16(2S-1)} Re_n^2\right) & \gamma = \tau \end{cases} \tag{2.15 a \& b}$$

$$\Theta_\gamma(Re_n) = \begin{cases} Re_n\left(\frac{2}{\sqrt{Re_n}} + 0.5\right) & \gamma = n \\ 2\sqrt{Re_n} & \gamma = \tau \end{cases} \tag{2.16 a \& b}$$

where parameter S is defined as

$$S = 2.0022 - \ln Re_n \tag{2.17}$$

Coefficients in the correlations that valid for 0.1< Re_n ≤100 are in Table 2.1. The coefficients for the correlations valid for $Re_n \leq a_0$ are defined in a way that the resistance coefficients converge asymptotically to the values predicted by the Stokes theory for slender bodies, as $Re_n \rightarrow 0$. Those values are given by

$$r_{n,St} = \frac{4\pi}{ln(4/\delta)} - \frac{\pi}{ln^2(4/\delta)} \tag{2.18}$$

$$r_{\tau,St} = \frac{2\pi}{ln(4/\delta)} - \frac{\pi/2}{ln^2(4/\delta)} \tag{2.19}$$

where δ is the aspect ratio, defined as $\delta = l_a/l_b$ (see Fig. 2.3 for the definitions of l_a and l_a).

The coefficients $S_{\gamma,k}$ that achieve a smooth transition between the regimes valid on $a_0 < Re_n \leq 0.1$ and the asymptotic values for $Re_n \rightarrow 0$ are expressed as

$$S_{\gamma,k} = \begin{cases} r_{\gamma,S} & k = 0 \\ 0 & k = 1 \\ \frac{3\Psi_{\gamma}(a_0) - a_0\Psi'_{\gamma}(a_0) - 3r_{\gamma,S}}{a_0^2} & k = 2 \\ \frac{2\Psi_{\gamma}(a_0) + a_0\Psi'_{\gamma}(a_0) + 2r_{\gamma,S}}{a_0^3} & k = 3 \end{cases} \quad \text{(2.20 a, b, c, \& d)}$$

The magnitude of a_0 also depends on the aspect ratio (AR) through the value of $r_{n,St}$:

$$a_0 = 2exp\left(2.0022 - \frac{4\pi}{r_{n,St}}\right) \quad (2.21)$$

2.1.2.2 Discrete Particle Phase

In this study, translations, rotations, and interactions of APIs and carrier particles were considered and calculated using the following equations for particle j (see Fig. 2.4), combining the Lagrange method and DEM based on our calibrated H-M model with the JKR cohesion model (Feng & Kleinstreuer, 2013; Feng & Kleinstreuer, 2014; Johnson, 2016; Mindlin, 1953; Zhao, 2020) (see the particle-particle interaction forces acting on particle j in Fig. 2.4)

$$m_{p,j}\frac{d\vec{u}_{p,j}}{dt} = \sum_i \vec{F}_{c,ji} + \vec{F}_{fp,j} + \vec{F}_{g,j} \quad (2.22)$$

$$I_j\frac{d\vec{\omega}_{p,j}}{dt} = \sum_i \vec{M}_{c,ji} + \vec{M}_{fp,j} \quad (2.23)$$

where $m_{p,j}$ is the particle mass, $\vec{F}_{c,ji}$ is the contact force that accounts for particle-particle and particle-wall interactions, $\vec{F}_{fp,j}$ is the force acting on the particle due to the fluid phase, I_j is the moment of inertia tensor, $\vec{\omega}_{p,j}$ is the angular velocity vector, $\vec{M}_{c,ji}$ is the torque generated by tangential forces that causes the rotation of the particle, and $\vec{M}_{fp,j}$ is the torque due to the fluid phase velocity gradient, which was neglected since the particle size is much smaller than the mesh cell size.

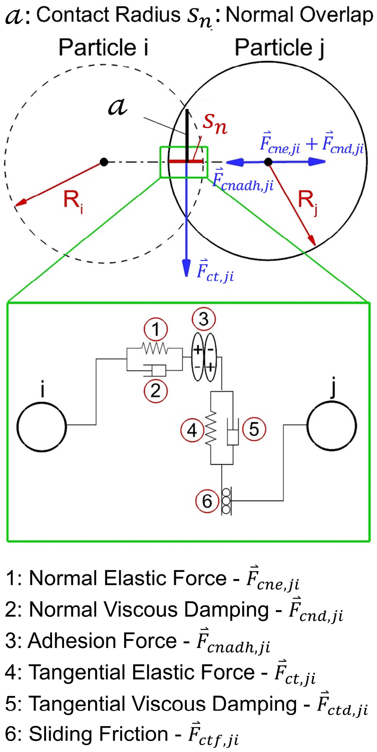
a: Contact Radius s_n: Normal Overlap
Particle i
Particle j
a
s_n
$\vec{F}_{cne,ji} + \vec{F}_{cnd,ji}$
$\vec{F}_{cnadh,ji}$
R_i
R_j
$\vec{F}_{ct,ji}$
1
2
3
4
5
6
i
j
1: Normal Elastic Force - $\vec{F}_{cne,ji}$
2: Normal Viscous Damping - $\vec{F}_{cnd,ji}$
3: Adhesion Force - $\vec{F}_{cnadh,ji}$
4: Tangential Elastic Force - $\vec{F}_{ct,ji}$
5: Tangential Viscous Damping - $\vec{F}_{ctd,ji}$
6: Sliding Friction - $\vec{F}_{ctf,ji}$

Figure 2.4: Schematic of particle-particle interaction forces in the Hertz-Mindlin based adhesion models in DEM

To accurately model the de-agglomeration and agglomeration behaviors among different APIs and lactose carriers with diameters from 1 to 200 μm (Calvert et al., 2011; Kinnunen et al., 2014), the dominant adhesive forces, i.e., van der Waals force and electrostatic force, must be integrated into the DEM contact force model. Among several available adhesion DEM models, the H-M model with Johnson-Kendall-Roberts (JKR) Cohesion (Johnson et al., 1971) has been evaluated and validated on its capability to simulate micro-/nano- scale powder packing process with the surface energy varying between 0.1 to 25 J/m^2 (Morrissey, 2013). The H-M JKR model can describe the adhesion resulted from the short-range surface forces for studies of agglomeration at micro-/nano-scale (Carrillo et al., 2010; Horabik & Molenda, 2016; Jiang et al., 2018; Modenese et al., 2012; Moreno-Atanasio, 2012; Wang et al., 2017). Therefore, we will initially employ the H-M model with JKR Cohesion to account for the adhesive behaviors between fine particles in this study and introduce a cutoff value for the inter-particulate distance to avoid the numerical singularity at particle contact.

The H-M model with JKR cohesion is a contact model that allows the modeling of de-agglomeration and agglomeration behaviors of fine powders. Specifically, the adhesive contact force was modeled based on the balance between the stored elastic energy (i.e., normal and tangential elastic forces) and the loss in the surface energy (adhesion force). It has been assumed that the adhesion originated from van der Waals force and electrostatic force. Specifically, the H-M model with JKR cohesion describes particle contacts as normal and tangential damped harmonic oscillators with tangential friction $\vec{F}_{cn,ji}$ and an adhesion force $\vec{F}_{ct,ji}$. Specifically, the JKR model includes the effect of elastic deformation, treats the effect of adhesion as surface energy only, and neglects adhesive stresses in the separation zone. Accordingly, inter-particle forces acting on particle j from particle i (see Fig. 2.4) can be separated into two forces in normal and tangential directions and expressed as

$$\vec{F}_{c,ji} = \vec{F}_{cn,ji} + \vec{F}_{ct,ji} \tag{2.24}$$

where $\vec{F}_{cn,ji}$ and $\vec{F}_{ct,ji}$ are normal and tangential contact forces, which can be written as:

$$\vec{F}_{cn,ji} = \vec{F}_{cne,ji} + \vec{F}_{cnd,ji} + \vec{F}_{cnadh,ji} \tag{2.25}$$

$$\vec{F}_{ct,ji} = \vec{F}_{cte,ji} + \vec{F}_{ctd,ji} + \vec{F}_{ctf,ji} \tag{2.26}$$

In Eq. (2.26), $\vec{F}_{cne,ji}$ is the normal elastic force, $\vec{F}_{cnd,ji}$ is the normal viscous damping force, and $\vec{F}_{cnadh,ji}$ is the adhesion force modeled by the JKR cohesion (Johnson et al., 1971). Specifically, using the Hertz spring dashpot model with JKR cohesion, the forces can be expressed as

$$\vec{F}_{cne,ji} = \left(K_n s_n^{\frac{3}{2}}\right)\vec{n}_{ij} = \left(\frac{4}{3}E^*\sqrt{R^*}s_n^{\frac{3}{2}}\right)\vec{n}_{ij} \tag{2.27}$$

$$\vec{F}_{cnd,ji} = \left(C_H s_n^{\frac{1}{4}}\dot{s}_n\right)\vec{n}_{ij} = \left(2\eta_H\sqrt{m^* K_n} s_n^{\frac{1}{4}}\dot{s}_n\right)\vec{n}_{ij} \tag{2.28}$$

$$\vec{F}_{cnadh,ji} = \sqrt{8\pi\Gamma E^* a^3}\vec{n}_{ij} \tag{2.29}$$

where K_n is the normal contact stiffness, E^* is the effective Young's Modulus, R^* is the effective radius, C_H is the normal damping coefficient, m^* is the effective mass, η_H is the normal damping ratio for the Hertzian model, defined by Eqs. (2.30) to (2.33), s_n is the normal contact overlap (see Fig. 2.4), $\dot{s}_n$ is the time derivative of s_n, $\vec{n}_{ji}$ is the unit normal vector, and a is the radius of contact between particles or between a particle and a boundary (see Fig. 2.4), whose value can be calculated by solving Eq. (2.35).

$$\frac{1}{E^*} = \frac{1-\sigma_1^2}{E_1} + \frac{1-\sigma_2^2}{E_2} \tag{2.30}$$

$$\frac{1}{R^*} = \begin{cases} \dfrac{2}{d_{p,j}} + \dfrac{2}{d_{p,i}} & \text{particle-particle contact} \\ \dfrac{2}{d_{p,j}} & \text{particle-boundary contact} \end{cases} \tag{2.31}$$

$$\frac{1}{m^*} = \begin{cases} \dfrac{1}{m_{p1}} + \dfrac{1}{m_{p2}} & \text{particle-particle contact} \\ \dfrac{1}{m_p} & \text{particle-boundary contact} \end{cases} \tag{2.32}$$

$$\eta_H = \frac{\sqrt{5}}{2}\eta \tag{2.33}$$

$$\varepsilon = \begin{cases} exp\left[-\dfrac{\eta}{\sqrt{1-\eta^2}}\left(\pi - \tan^{-1}\dfrac{2\eta\sqrt{1-\eta^2}}{1-2\eta^2}\right)\right] & 0 \le \eta < \dfrac{\sqrt{2}}{2} \\ exp\left(-\dfrac{\eta}{\sqrt{1-\eta^2}}\tan^{-1}\dfrac{2\eta\sqrt{1-\eta^2}}{2\eta^2-1}\right) & \dfrac{\sqrt{2}}{2} \le \eta < 1 \\ exp\left(-\dfrac{\eta}{\sqrt{\eta^2-1}}\ln\dfrac{\eta+\sqrt{\eta^2-1}}{\eta-\sqrt{\eta^2-1}}\right) & 1 < \eta \end{cases} \tag{2.34}$$

$$s_n = \frac{a^2}{R^*} - \left(\frac{2\pi\Gamma a^2}{E^*}\right)^{\frac{1}{2}} \tag{2.35}$$

In Eq. (2.30), E_1 and E_2 are Young's modulus of the two contacting particles or the particle and the boundary. In Eq. (2.31), $d_{p,j}$ and $d_{p,j}$ are the sizes of the contacting particles. In Eq. (2.32), m_{p1} and m_{p2} are the mass of the contacting particles, and m_p is the mass of the particle in contact with the boundary. In Eq. (2.33), η is the damping ratio, a dimensionless parameter whose value is related to the restitution coefficient ε, defined in Eq. (2.34). The restitution coefficient ε is a user input t the particle-particle interaction or particle-boundary interaction according to the case. In Eq. (2.35), Γ is the surface energy.

The tangential force $\vec{F}_{ct,ji}$ (see Eq. (2.24)) consists of the tangential spring force $\vec{F}_{cte,ji}$, the tangential viscous damping force $\vec{F}_{ctd,ji}$, and the frictional force $\vec{F}_{ctf,ji}$. $\vec{F}_{ct,ji}$ is calculated using the Mindlin-Deresiewicz model, which can be expressed as

$$\vec{F}_{t,ji} = -\mu_p \left\|\vec{F}_{cn,ji}\right\| \left(1 - \lambda^{\frac{3}{2}}\right) \frac{\vec{s}_\tau}{\|\vec{s}_\tau\|} + \eta_\tau \left(\frac{6\mu_p m^* \left\|\vec{F}_{cn,ji}\right\|}{s_{\tau,max}}\right)^{\frac{1}{2}} \lambda^{\frac{1}{4}} \dot{\vec{s}}_n \tag{2.36}$$

$$\lambda = \begin{cases} 1 - \dfrac{\min\left(\|\vec{s}_\tau\|, s_{\tau,max}\right)}{s_{\tau,max}} & |\vec{s}_\tau| \le s_{\tau,max} \\ 0 & |\vec{s}_\tau| > s_{\tau,max} \end{cases} \tag{2.37}$$

where μ_p is the friction coefficient defined in Eq. (2.38), η_τ is the tangential damping ratio estimated using Eq. (2.39), $\vec{s}_\tau$ is the tangential relative displacement at the contact, $\dot{\vec{s}}_n$ is the tangential component of the relative velocity at the contact and $s_{\tau,max}$ is the maximum relative tangential displacement at which particles begin to slide (see Eq. (2.40)).

$$\mu_p = \begin{cases} \mu_s & \text{no sliding at the contact} \\ \mu_d & \text{sliding at the contact} \end{cases} \tag{2.38}$$

in which μ_s and μ_d are the static and dynamic friction coefficients, respectively.

$$\eta_\tau = \frac{\ln \varepsilon}{\sqrt{\ln^2 \varepsilon + \pi^2}} \tag{2.39}$$

The value of the maximum relative tangential displacement $s_{\tau,max}$ is determined by

$$s_{\tau,max} = \mu_p \left(\frac{1-\sigma_1}{2-\sigma_1} + \frac{1-\sigma_2}{2-\sigma_2}\right)^{-1} s_n \tag{2.40}$$

where σ_1 and σ_2 are the Poisson’s ratios of the two particles or the particle and the boundary.

Moreover, the regional deposition of particles in airways is quantified using deposition fraction (DF), which is defined as the mass of deposited particles in a specific region in the lung divided by the total mass of particles entering the mouth (Feng et al., 2018; Tian et al., 2015).

2.1.3 Particle Material Properties

The API and excipient employed in this work are Tiotropium and lactose (monohydrate, α-). The material properties of these two ingredients are summarized in Table 2.2. Specifically, the Poisson's Ratio and Young's Modulus of the API, as well as the cohesion/adhesion forces between API-API, lactose-lactose, and API-lactose, were measured using a commercial scanning probe microscopy (SPM) system (Asylum Research MFP-3D, Oxford Instruments company). Then, the surface energies between drug particles, carriers, and drug-carrier are calculated based on the JKR model (Walton, 2008) (see Table 2.3 for the computed results). The rest of the data were obtained from the open literature (see Table 2.2).

Table 2.2: Material properties of particles.

Particles Properties	Tiotropium	Lactose (monohydrate, α-) [b]
Particle density [kg/m3]	1.53 [a]	1.52
Poisson's Ratio	0.28	0.3
Young's Modulus [MPa]	46.7	1.68

Data Source:
[a] https://patents.google.com/patent/US8163913B2/en
[b] Bassam, F., York, P., Rowe, R., & Roberts, R.J. (1990). Young's Modulus of Powders Used as Pharmaceutical Excipients. *International Journal of Pharmaceutics*, 64, 55-60.

2.2 Model Verification, Calibration, and Validation

2.2.1 CFD Model Verification and Calibration

To reduce the high computational cost using LES, verifications were done by using different RANS models to match the LES simulation data of airflow field in the flow channel of SH DPI, in order to find the RANS model that can provide similar predictions to LES. Despite the validation of the k-ω model documented in an existing publication (Donovan et al., 2012) for studying the airflow patterns in the SH device, we verified two

RANS models. Specifically, in terms of airflow dynamics prediction, we first employed an in-house wall-modeled large-eddy simulation (WMLES) subgrid-scale (SGS) model and two RANS models (k-ω and k-ε models) to simulate airflows in the SH. The best RANS model, which generated the most similar airflow field compared with the LES results, i.e., k-ω model, was selected to perform CFD simulation in DPIs. The CFD simulation results (see Fig. 2.5) verifies that k-ω model can accurately predict the airflow pattern inside the DPI comparing with the LES results.

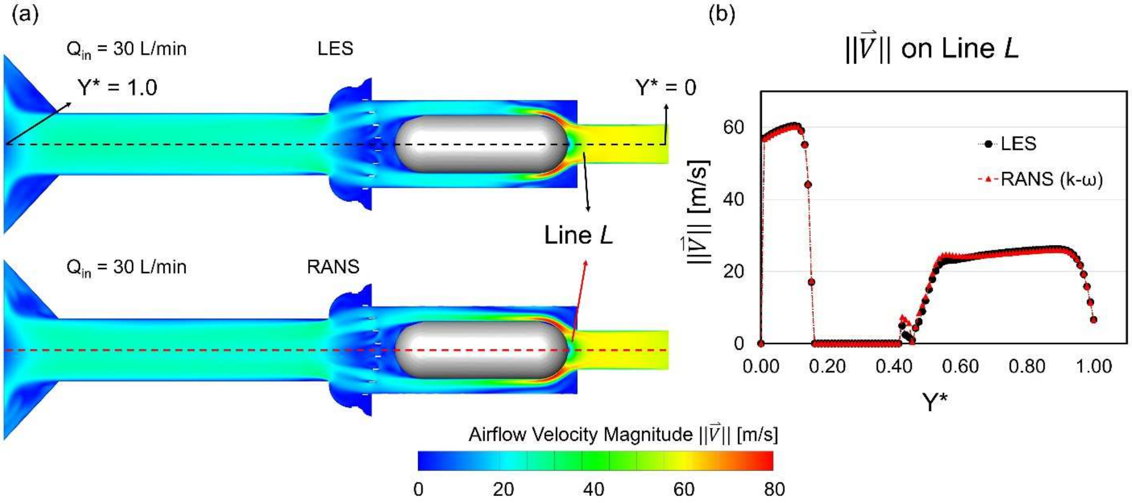

Figure 2.5: CFD model verification using SH DPI: (a) comparison of the airflow pattern between the results from LES and RANS k-ω models and (b) comparison of airflow velocity magnitude $\|\vec{V}\|$along with Line L

In terms of the CFD model validation, the resistance of the DPI was computed numerically and compared with experimental data. Comparisons of pressure drops shown in Fig. 2.6 indicate that the k-ω model is an appropriate RANS model to capture the pressure drop through the flow channel, and hence selected for modeling the airflow.

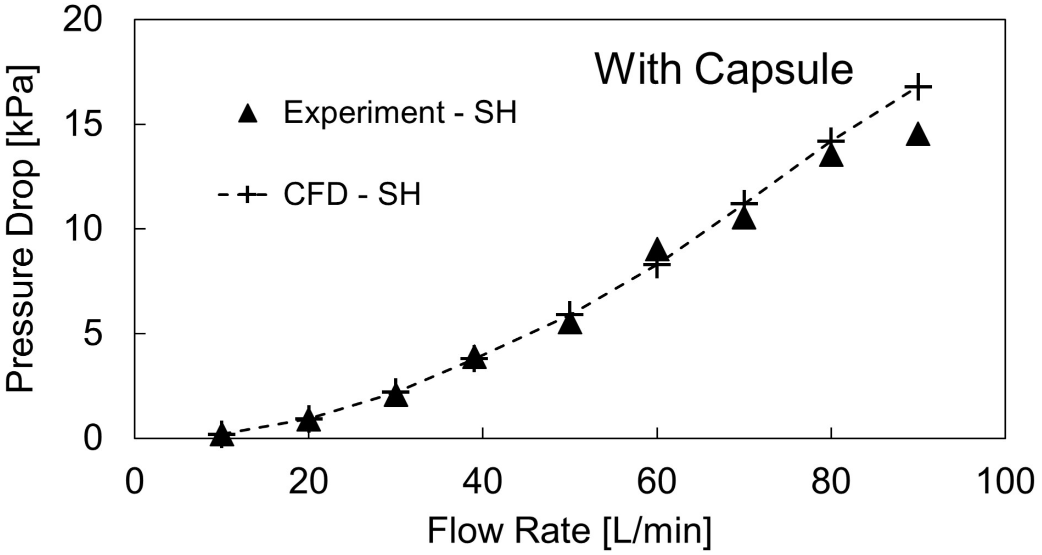

Figure 2.6: CFD model validation: comparison of SH DPI pressure drop between the results from k-ω SST model and experimental measurement

In addition, to build a reliable computational model that can accurately simulate the particle trajectories and airflow patterns in human respiratory systems, the Lagrange particle tracking model also requires validation. The customized Lagrangian particle-tracking model has been validated by the comparisons with the *in vitro* particle deposition fractions (DFs) in the oral/nasal cavities and TB tree, which are well documented in existing publications (Feng & Kleinstreuer, 2013; Feng et al., 2016; Feng et al., 2015; Feng et al., 2020; Feng et al.; Haghnegahdar et al., 2018; Haghnegahdar et al., 2019; Zhao et al., 2019; Zhao, 2020; Zhao, 2020).

2.2.2 CFD-DEM Model Calibration

Calibration is required for the hybrid CFD-DEM model, since parameters, such as surface energy between particles (both API and lactose) and DPI device wall, static friction coefficient, as well as dynamic friction coefficient are critical for the model to be able to accurately predict the particle-particle interactions and thereby the APSD. However, experimental measurements of those parameters are challenging. Therefore, calibration of friction coefficients and surface energy between particles and walls was performed using the simulation method. A range of surface energy values (from 0.01 to 0.1 J/m^2) have been tested in our CFD-DEM model to match the DPI efficacy (fraction of drug delivered to the orifice) with experimental data. Specifically, using actuation flow rate (Q_{in}) 39 L/min, we performed the CFD-DEM simulations with different values of surface energy between particle and wall, the friction coefficient between particles, and friction coefficient between particle and wall (see Table 2.3 for the simulation results with different parameter values). The API delivery efficiency of SH DPI was compared with experimental data for the parameter value calibrations. Based on the comparisons, the parameter values that were not available in experiments are determined, and the values are listed in Table 2.3.

Table 2.3: Calibrated DEM properties for API and lactose particles.

	API-API	API-Lactose	Lactose-Lactose	API-Wall	Lactose-Wall
Surface Energy [mJ/m^2]	43.4	47.5	13.4	1290	1290
Static Friction Coefficient	0.7	0.7	0.7	0.5	0.5
Dynamic Friction Coefficient	0.7	0.7	0.7	0.5	0.5

To further determine the JKR surface energy between particles and DPI wall, regressions were done to correlate the relationship between the drug particle (mass) delivery efficiency and the JKR surface energy (see Fig. 2.7). It can be observed from Fig. 2.7 that the relationship between particle delivery efficiency is a linear function of particle-wall JKR surface energy when the surface energy is less than 2 J/m^2, which can be given as:

$$DE_{API} = -43.56\,\Gamma_{particle-device} + 113.4 \quad \Gamma_{particle-device} \in [0.4, 2]\ \mathrm{J/m^2} \tag{2.41}$$

Therefore, it can be determined that $\Gamma_{particle-device}$=1.29 J/m^2. In addition, it can be further concluded that if the surface energy property between particles and DPI walls can be reduced, the delivery efficiency can be enhanced accordingly. The calibrated values of the parameters required for DEM simulations are listed in Table 2.3.

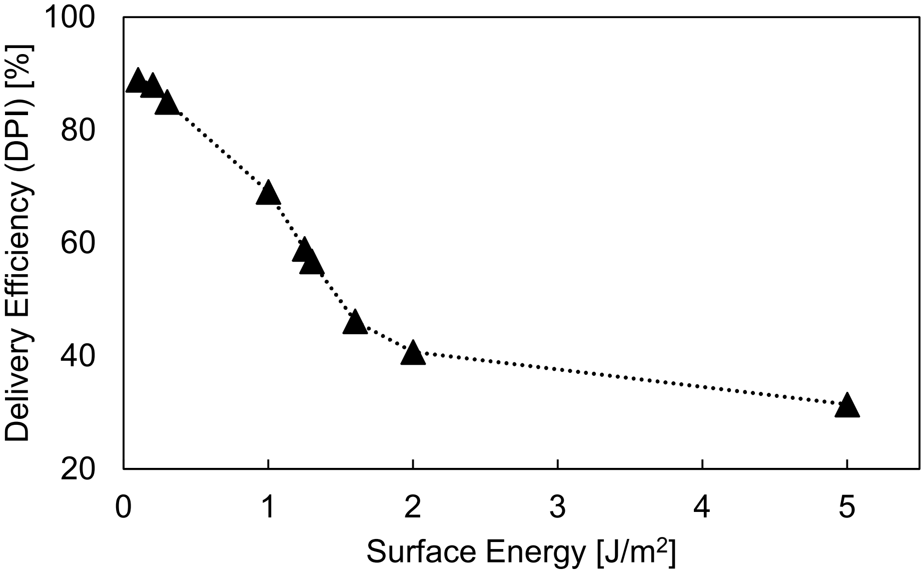

Figure 2.7: The relationship between JKR particle-wall surface energy and the DPI delivery efficiency

2.3 Numerical Setup

Both CFD simulations of the airflow field in the DPI flow channel and CFPD simulations of pulmonary air-particle flow dynamics were performed using Ansys Fluent 2020 R2 (Ansys Inc., Canonsburg, PA) and performed on a local Dell Precision T7810 workstation (Intel® Xeon® Processor E5-2643 v4 with dual processors, 64 cores and 128

GB RAM). The Semi-Implicit method for pressure-linked equations (SIMPLE) algorithm was employed for the pressure-velocity coupling, and the least-squares cell-based scheme was applied to calculate the cell gradient. The second-order scheme was employed for pressure discretization. In addition, the second-order upwind scheme was applied for the discretization of momentum and turbulent kinetic energy. Convergence is defined for continuity, momentum, and supplementary equations when residuals are less than 1.0e-5.

Coupled with CFD simulations of the airflow field in the DPI flow channel, DEM simulations were performed using Rocky 4.4.3 (ESSS, Woburn, MA) on local Dell Precision T7810 workstation (Intel® Xeon® Processor E5-2643 v4 with dual processors, 64 cores and 128 GB RAM), Dell Precision 7920 Tower workstation (Intel® Xeon® Silver 4116 with dual processors and 128 GB RAM) with dual Quadro RTX 6000 GPU.

In-house user-defined functions (UDFs) were developed for

(1) Measuring the emitted APSD from the DPI orifice and converting it into a release map that is used for lung deposition simulation;

(2) Counting coordination number which can be used for agglomeration and de-agglomeration analysis;

(3) Specifying the transient inhalation profile at the mouth;

(4) Recovering the anisotropic corrections on turbulence fluctuation velocities;

(5) Modeling the Brownian motion induced forces;

(6) Storing particle deposition data.

2.4 Results and Discussion

2.4.1 Airflow Structure in SH DPI Flow Channel

Distributions of normalized velocity magnitude and turbulence intensity (TI) with four different actuation flow rates, i.e., Q_{in}=30, 39, 60, and 90 L/min, are shown in Figs. 2.8 and 2.9. Specifically, the normalized velocity magnitude contours at plane z=0 are shown in Fig. 2.8. It can be observed that the maximum velocity locates at the bottom region of the capsule. The velocity contours with Q_{in}=30 and 39 L/min share similar patterns in the computational domain near the capsule. Flow detachments can be found downstream the locations where the airflow impacts the capsule. At higher flow rates, i.e., 60 and 90 L/min, flow separations did not occur in the capsule bottom region, but shift downstream. Indeed, with the increase in actuation flow rate, the flow momentum after the impaction of the capsule is higher. Therefore, the flow with higher Q_{in} (i.e., 60 and 90 L/min) is able to conquer the viscous dissipation effect, and generate no flow separation near the capsule wall, compared with the flow with lower actuation flow rates (i.e., 30 and 39 L/min). Due to the high Q_{in}, high turbulent intensity (i.e., TI>3) can be noticed near the capsule chamber wall in cases with higher flow rates (i.e., Q_{in}=60 and 90 L/min) (see Fig. 2.9). In contrast, for cases with Q_{in}=30 and 39 L/min, high TI (TI>3) occurs at the lower middle region near the capsule wall and the bottom region of the capsule chamber. The TI is approximately 30% in most of the upper region from the top of the capsule chamber to the mouthpiece. In addition, as Q_{in} increases, the TI in extending tube increases.

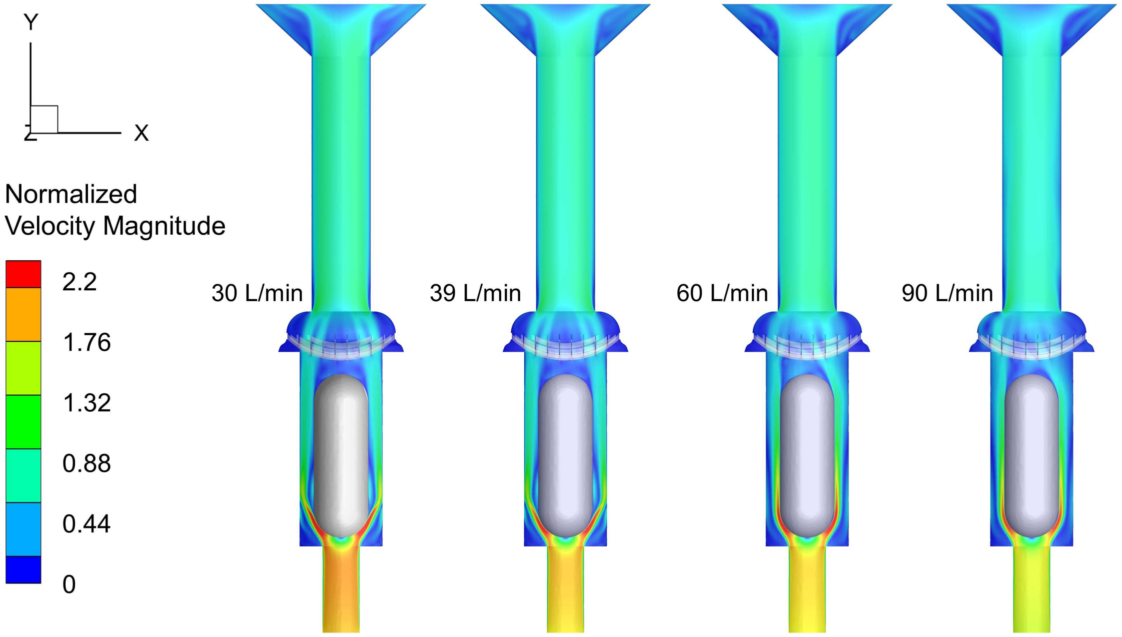

Figure 2.8: Normalized velocity magnitude contours ($\|\vec{V}\|/\|\overrightarrow{V_{in}}\|$) at plane z=0 in DPI at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min)

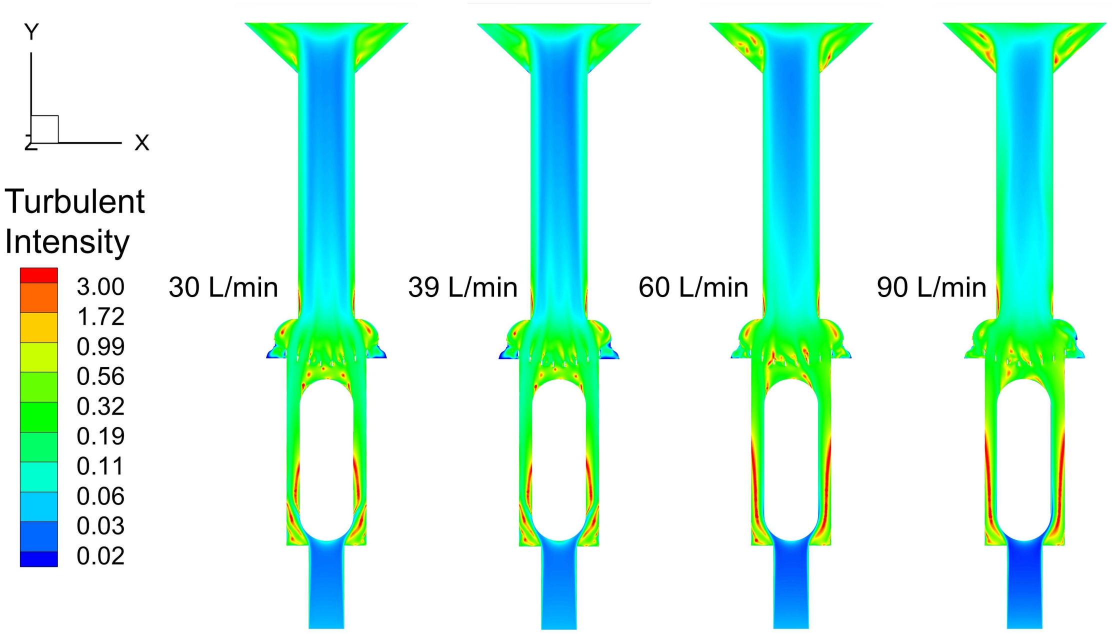

Figure 2.9: Turbulent intensity (TI) distributions at the mid-plane z=0 in the DPI flow channel with different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min)

2.4.2 Drug Particle Deposition in DPI and DPI Delivery Efficiency

The drug particle deposition distribution inside the flow channel of the SH DPI with different Q_{in} and different AR of lactose carrier particles are shown in Fig. 2.10. Since only the AR of lactose was investigated and all the APIs were assumed as spherical particles, in this study AR is used to represent AR of lactose for the purpose of conciseness. At 30 and 39 L/min, the “hot spots” of lactose particle depositions are the surface of the capsule and the capsule chamber wall near the bottom opening of the chamber. Another concentrated deposition site for lactose particles is the grid region, especially for spherical lactose particles. At 60 or 90 L/min, the number of deposited lactose particles in DPI decreases compared with 30 and 39 L/min, since more lactose particles are transported to the mouthpiece by the stronger airflow. It can also be observed that the carrier particle shape has a noticeable influence on lactose deposition distributions in the DPI. Specifically, at Q_{in} of 30 and 39 L/min, the deposited lactose in the capsule chamber is reduced as the AR increases (see Fig. 2.11 (b) for the total $DF_{lactose\text{-}DPI}$). At 60 L/min, when AR increases, fewer lactose particles are trapped in the capsule chamber but are deposited on the wall of extending tube. Similar deposition patterns can be observed for 90 L/min cases, although the majority of the lactose is emitted, which makes this result not evident for 90 L/min cases. Therefore, it can be concluded that with the same particle volume, more elongated lactose particles can better avoid collision with the wall and more accessible to be resuspended by the airflow after deposition, which leads to less deposition in the DPI flow channel than particles with more isotropic shapes.

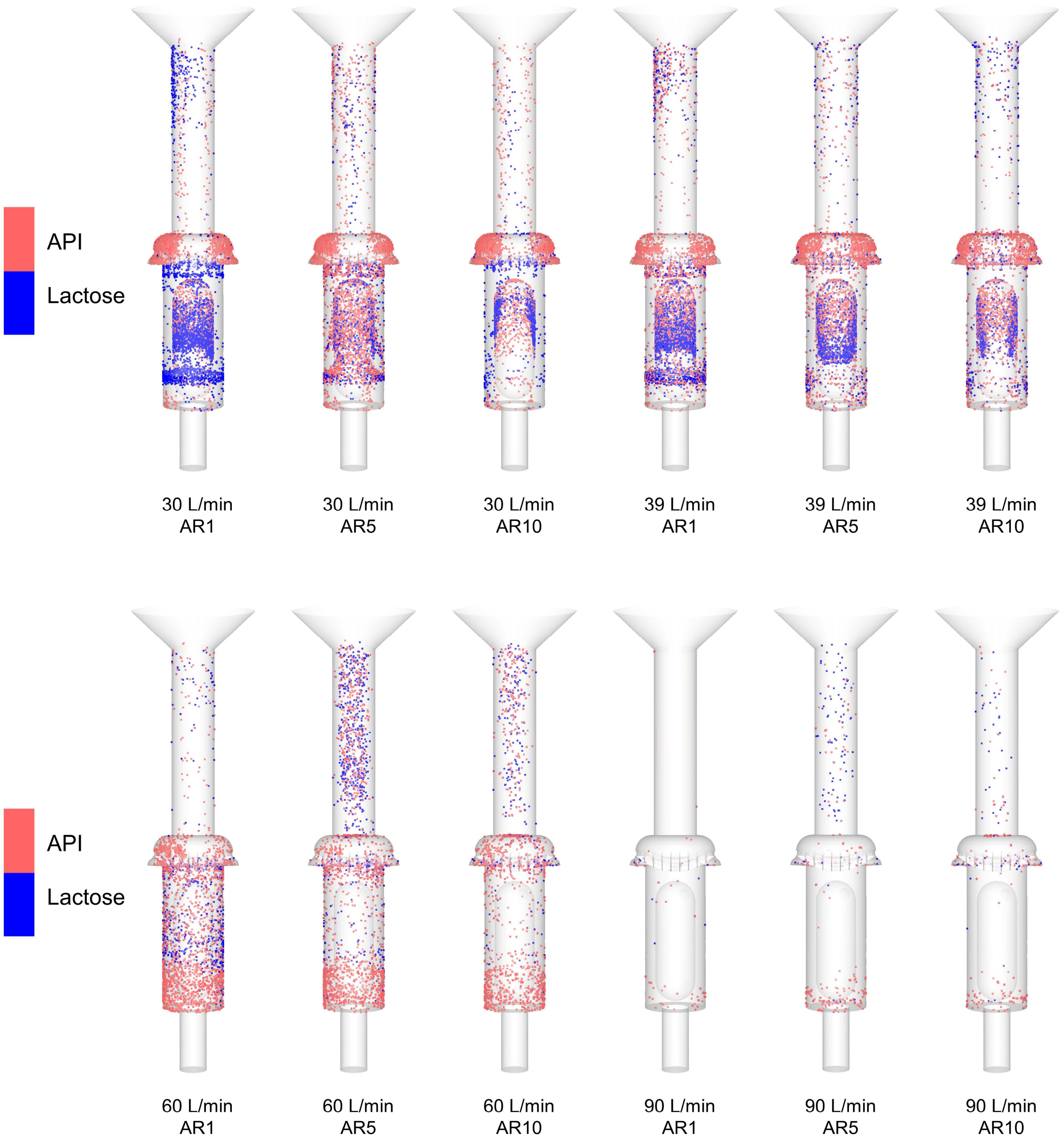

Figure 2.10: Deposition distributions of API and lactose particles in SH DPI at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min)

For API particles, Fig. 2.10 shows that most API particles deposited in the capsule chamber, capsule surface, and the cap wall above the grid for cases with Q_{in}=30 and 39 L/min. At Q_{in}=60 L/min, the number of API deposited on the cap wall and surface of the capsule is reduced compared with 30 and 39 L/min cases, while more API particles deposited at the bottom of the capsule chamber. At Q_{in} = 90 L/min, most API particles were emitted through the mouthpiece opening as there are few particles trapped either inside the capsule chamber of the cap wall.

The total $DF_{API\text{-}DPI}$ and $DF_{lactose\text{-}DPI}$ in the SH device are measured and shown in Fig. 2.11. In general, the AR has little effect on the total $DF_{API\text{-}DPI}$, except for cases with a flow rate of 30 L/min. At 30 L/min, the total $DF_{API\text{-}DPI}$ reaches a peak of 8.8% with AR=5. When AR=1 or 10, with the increase of Q_{in} from 30 to 90 L/min, the total $DF_{API\text{-}DPI}$ first increases (until 60 L/min) and then decreases. At low flow rates, even though the convection effect is weaker than the high flow rate condition, the turbulent effect is also weak (see Fig. 2.9) so that fewer APIs are deposited in the capsule chamber compared with 39 and 60 L/min cases. At high Q_{in} (e.g., 60 L/min), the TI in the capsule chamber can reach as high as 2700, which leads to a high DF of API in the bottom region of the capsule chamber (see Fig. 2.10 60 L/min cases). Moreover, the deposited API in that region cannot be driven up into the airflow as the convection effect in the chamber at 60 L/min is not strong enough. However, when the Q_{in} increases to 90 L/min, the convection effect dominates as it is strong enough to overcome the surface energy between API and the device wall. Thus, APIs can resuspend into the airflow and be delivered to the mouthpiece opening of the DPI, which results in a low $DF_{API\text{-}DPI}$ at 90 L/min.

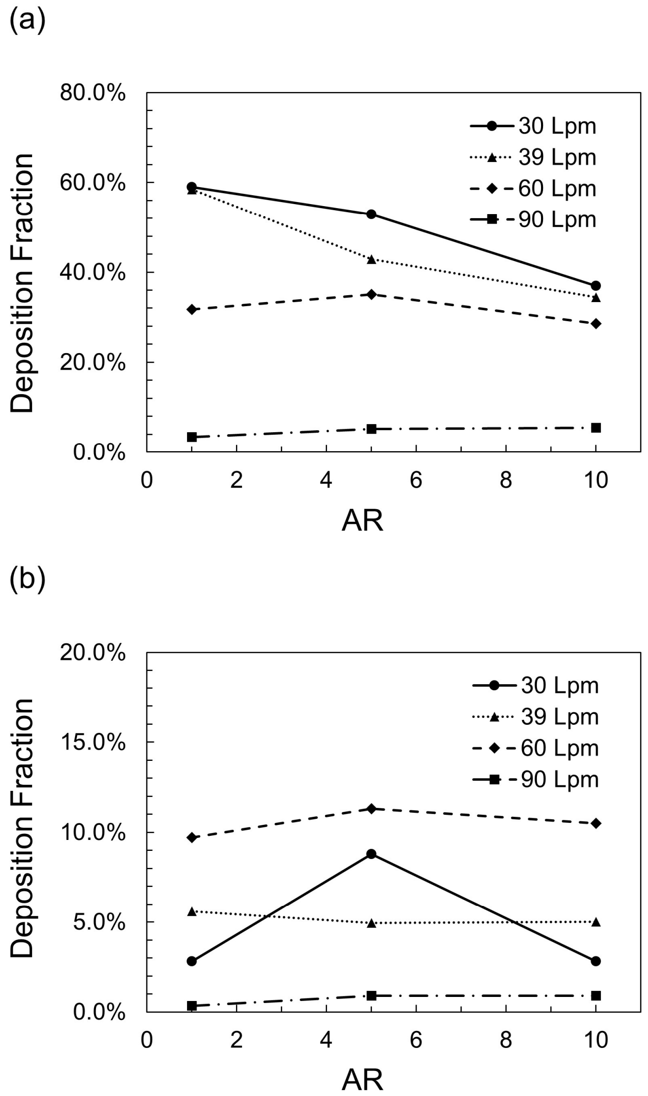

Figure 2.11: Deposition fractions (DFs) of (a) API and (b) lactose in the flow channel of SH DPI

The total $DF_{lactose\text{-}DPI}$ in the SH DPI is impacted both by Q_{in} and AR as shown in Fig. 2.11 (b). Specifically, the total $DF_{lactose\text{-}DPI}$ in the DPI decreases significantly from 59% to less than 6.0% as the flow rate increases from 30 to 90 L/min when AR=1. This result implies that the turbulence has a weaker effect on the $DF_{lactose\text{-}DPI}$ than $DF_{API\text{-}DPI}$. When AR=10, although the same trend is observed, the $DF_{lactose\text{-}DPI}$ only decreases by 8.6% as the flow rate increases from 30 to 60 L/min. In terms of the AR effect on the $DF_{lactose\text{-}DPI}$, at flow rates of 30 and 39 L/min, the total $DF_{lactose\text{-}DPI}$ decreases from over 50% to approximately 35% as AR increases from 1 to 10. The AR influence on DF is not evident when the flow rate reaches 60 and 90 L/min, as the total $DF_{lactose\text{-}DPI}$ fluctuates around 30% and 4%, respectively.

2.4.3 Effects of Particle Shape and Actuation Flow Rate on the Emitted APSD

The effects of particle shape of lactose (AR) on emitted APSD are shown in Fig. 2.12. It should be noted that the number fractions (NF) in Figs. 2.12 and 2.13 is defined as the number of particles within a specific size bin divided by the total number of emitted particles, including both API and lactose. At Q_{in}=30 L/min, similar APSD patterns can be observed for particles with d_p from 50 μm to 114 μm with different lactose particle shapes (see Figs. 2.12 (a)). Moreover, the case with $AR_{lactose}$=0 predicts the highest NF (i.e., 95%) for small particles ($d_p \leq 4.3$ μm), which are mostly API. At Q_{in}=39 L/min, the case with $AR_{lactose}$=10 predicts lower NF for small particles due to the high NF of particles with d_p=90 μm predicted (see Fig. 2.12 (b)). When the flow rate increases to 60 and 90 L/min (see Figs. 2.12 (c) and (d)), using more elongated carrier particles (AR=10) predicted higher NF for small particles ($d_p \leq 4.3$ μm) than using carrier particles with less anisotropic shapes (AR=1 and 5). Therefore, to enhance the API delivery efficiency, more elongated lactose with AR=10 is favored compared with spherical and less elongated shapes.

The effect of Q_{in} on the APSD is presented in Fig. 2.13. In general, the NF_{API} ($d_p \leq 4.3$ μm) are at a high level ranging from 92% to 96% for all values of Q_{in}. To be specific, in cases of spherical lactose (see Fig. 2.13 (a)), the NF_{API} decreases with Q_{in}, since particles with large size (d_p>30 μm) are easier to be emitted at a higher flow rate. For particles with 10 μm <d_p<60 μm, the NF$_{\text{lactose}}$ increases with Q$_{\text{in.}}$ Especially for Q$_{\text{in}}$=90 L/min, the NF of particles with d_p=40 μm reaches 2.7%. At AR=5 (see Fig. 2.13 (b)), cases with all four Q$_{\text{in}}$ setups predict a similar trend of APSD as at AR=1 (see Fig. 2.13 (a)). Specifically, Q$_{\text{in}}$=90 L/min case predicts the lowest NF$_{\text{API}}$ (93.1%) in all four Q$_{\text{in}}$ setups. For particles with d_p>20 μm, high Q$_{\text{in}}$ cases (e.g., 60 and 90 L/min) generate higher NF$_{\text{lactose}}$ than low Q$_{\text{in}}$ (i.e., 30 L/min) case. In contrast, at AR=10 (see Fig. 2.13 (c)), Q$_{\text{in}}$=39 L/min lead to the lowest NF$_{\text{API}}$ (d_p≤4.3 μm) compared with Q$_{\text{in}}$=30, 60 and 90 L/min cases. For all four Q$_{\text{in}}$ setups, NF$_{\text{lactose}}$ (d_p>20 μm), high Q$_{\text{in}}$ cases (e.g., 39, 60 and 90 L/min) tend to generate higher NF$_{\text{lactose}}$ than low Q$_{\text{in}}$ (i.e., 30 L/min) case.

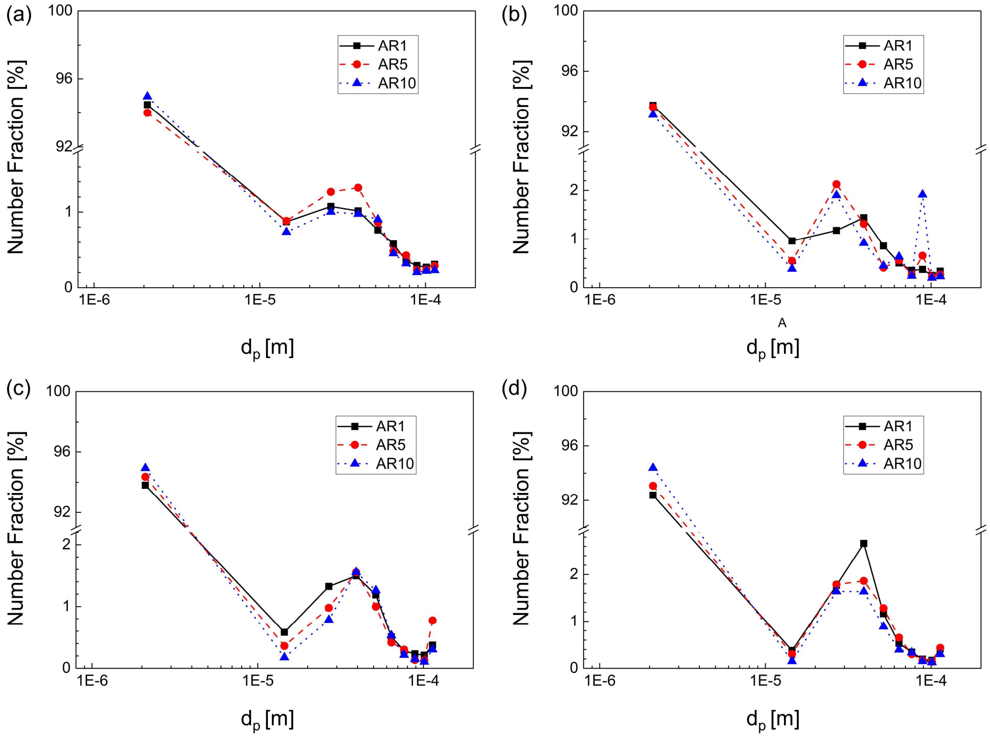

Figure 2.12: DPI emitted APSDs at different actuation flow rates: (a) 30 L/min, (b) 39 L/min, (c) 60 L/min, and (d) 90 L/min

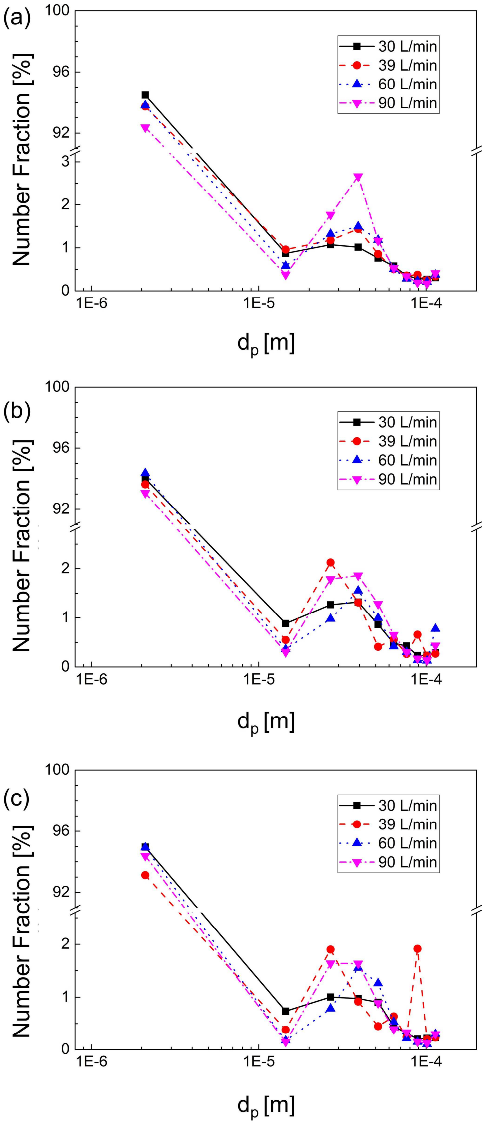

Figure 2.13: DPI emitted APSD with different lactose aspect ratios (ARs): (a) AR=1, (b) AR=5, and (c) AR=10

2.4.4 Airflow Structure in the Human Respiratory System

The inspiratory airflow structures at the sagittal plane y=0 are shown in Figs. 2.14 and 2.15. It should be noted that the mouth entrance has the same elliptic shape as the mouthpiece of the SH DPI. It is evident that the highest flow velocity occurs at the mouth entrance due to the narrowed mouth opening area (see Fig. 2.14). When the inhalation flow rate, which is equal to the actuation flow rate Q_{in}, increases to 90 L/min, the airflow recirculation in the oral cavity is obvious, and the laryngeal jet becomes more noticeable. The turbulent kinetic energy (TKE) illustrated in Fig. 2.15 also demonstrates a strong turbulence effect in the oral cavity and oropharynx for the case with Q_{in}=90 L/min.

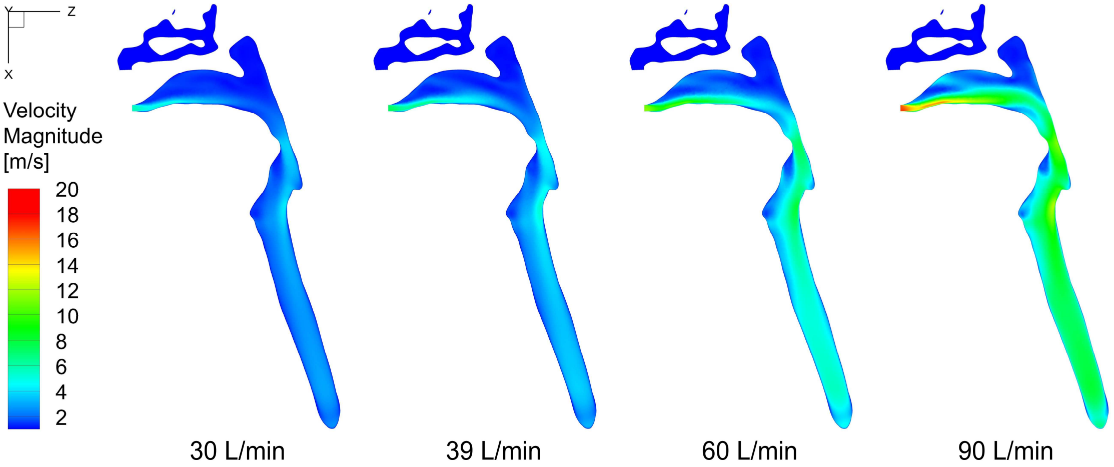

Figure 2.14: Velocity magnitude contours at sagittal plane y=0 in airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min)

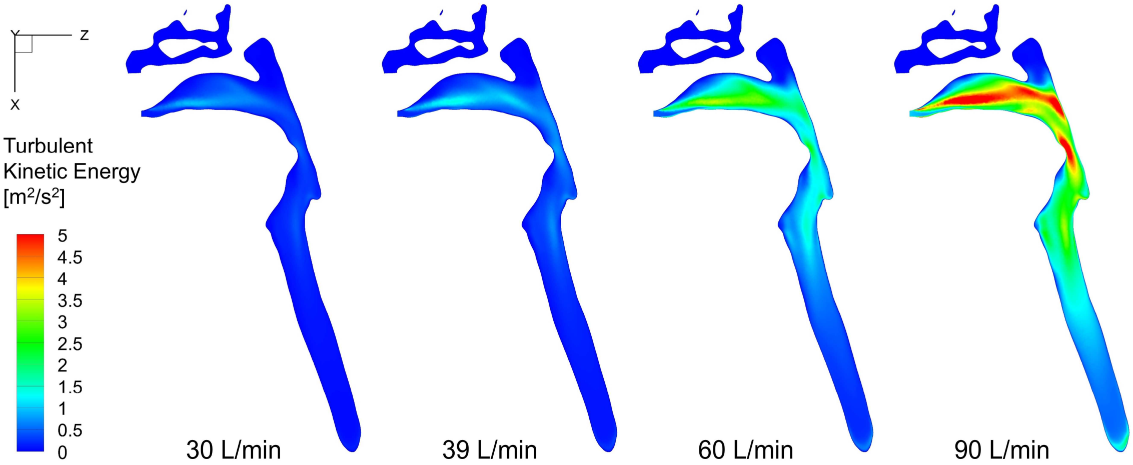

Figure 2.15: Turbulent kinetic energy contours at sagittal plane y=0 in airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min)

2.4.5 Effects of Actuation Flow Rate and AR on Particle Deposition in Airways

The lung deposition distributions of lactose (AR=1) and its regional deposition fractions (RDFs) in the airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min) with AR=1, 5, and 10 are shown in Figs. 2.16 and 2.17, respectively. Figures 2.18 and 2.19 illustrate the lung deposition distributions of API and $RDF_{API\text{-}lung}$ under different Q_{in} and AR setups, respectively.

All the lactose particles are trapped in the oral cavity, oropharynx, and laryngopharynx, as shown in Fig. 2.16. The lactose deposited on the tongue is mainly due to the gravitational effect, as it dominates the trajectory of large particles (d_p>50 μm). Another deposition locations for lactose are at the posterior of the oropharynx and laryngopharynx. This is due to the impaction of the mouth jet (see Fig. 2.14) and the large inertia of lactose. In terms of $RDF_{lactose\text{-}lung}$, several observations can be made based on the results shown in Fig. 2.17: (1) At Q_{in}=30 and 39 L/min, the $DF_{lactose\text{-}oral\ cavity}$ decreases with the AR, while at Q_{in}=60 and 90 L/min, AR has little influence on $DF_{lactose\text{-}oral\ cavity}$ with a value around 50%; (2) At low Q_{in} (30 L/min), the $DF_{lactose\text{-}oropharynx}$ increases with the AR, while at Q_{in}=39, 60 and 90 L/min, $DF_{lactose\text{-}oropharynx}$ decreases with AR; (3) $DF_{lactose\text{-}laryngopharynx}$ increases with AR for all Q_{in} setups, especially at Q_{in}=90 L/min, which indicates that elongated particles are easier to be transported by the convective flow.

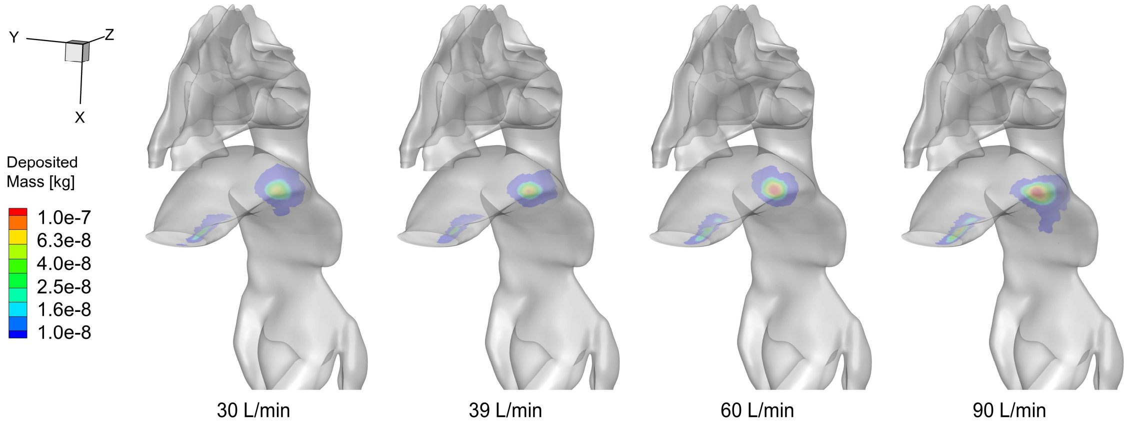

Figure 2.16: Deposition distributions of lactose in the upper airway at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min) with lactose AR=1

It can be observed from Fig. 2.18 that due to much higher particle inertia at high Q_{in} (e.g., 90 L/min), there is a significant amount of particles get entrapped in the oropharynx, glottis, and trachea. For example, with spherical lactose, when the Q_{in} increases from 30 to 90 L/min, the DF of API in the upper airway (from mouth to the trachea) increases from 20.1% to 44.2% (see Fig. 2.19). Moreover, the stronger laryngeal jet effect at 90 L/min also results in the highest DF of API in the first bifurcation (8.8%) compared with 4.1% at 30 L/min, 5.0% at 39 L/min, and 6.0% at 60 L/min (see Fig. 2.19). Although a high flow rate leads to high DF of API in the upper airway, which is not optimal in terms of drug delivery efficacy, it does enhance the DF of API in the lower airway. For example, at lactose AR of 1, DF of API is increased by 14.1% (more than twice) when Q_{in} increases from 30 to 90 L/min. Figure 2.18 also shows that at the same Q_{in} investigated in this study, AR has little effect on the $DF_{API\text{-}lung}$. By combining the DPI delivery efficiency, the general DPI-airway drug delivery efficiency is calculated and listed in Table 2.4. The result demonstrates that high Q_{in} is favored to achieve the best drug delivery efficacy for the SH

DPI, since it is the dominant factor in terms of the DF of API in the region from G3 to G13 compared with the particle shape of lactose.

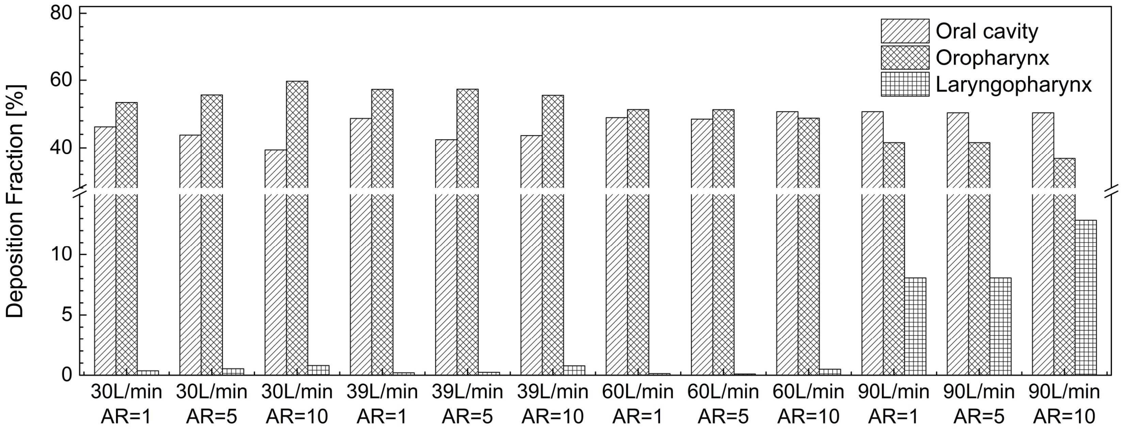

Figure 2.17: Regional deposition fractions (RDFs) of lactose in airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min) with AR=1, 5, and 10

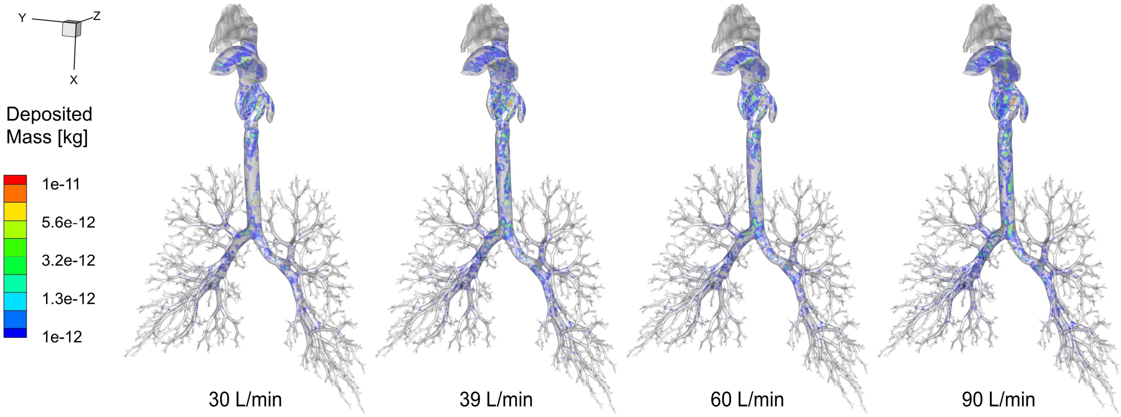

Figure 2.18: Deposition distributions of API in the airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min) with lactose AR=1

Table 2.4: DPI delivery efficiencies *vs.* AR and Q_{in}

AR	30 L/min	39 L/min	60 L/min	90 L/min
1	12.67%	15.38%	17.30%	26.99%
5	11.96%	15.58%	16.66%	26.31%
10	12.86%	15.18%	17.06%	26.22%

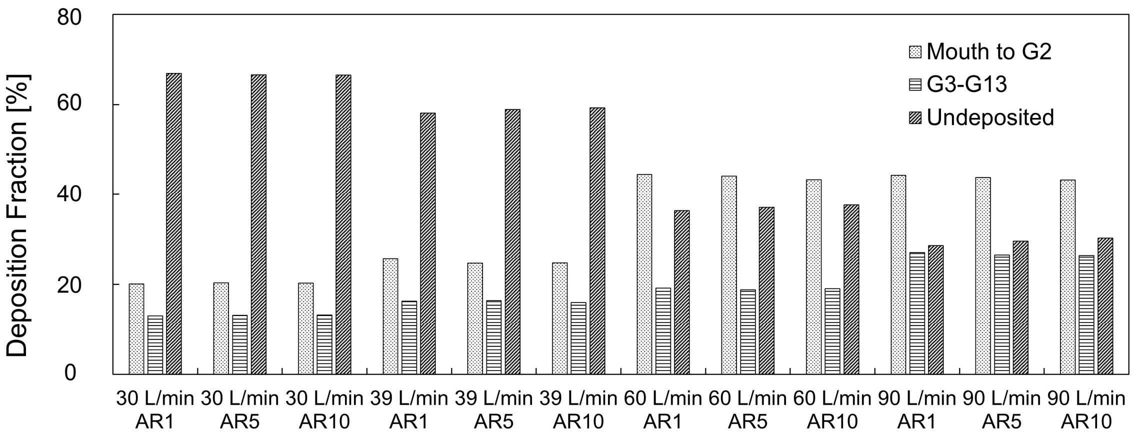

Figure 2.19: Regional deposition fractions (RDFs) of API in airway model at different actuation flow rates (Q_{in}=30, 39, 60, and 90 L/min) with AR=1, 5 and 10

2.5 Conclusions

The following conclusion can be drawn from the results of the CFD-DEM and CFPD of drug particle transport in SH DPI and a subject-specific respiratory system:

(1) Although the AR of lactose influences the API and lactose deposition distribution in the SH DPI significantly, it has little effect on the DPI delivery efficiency;

(2) The SH DPI delivery efficiency decreases with Q_{in} increasing from 30 to 60 L/min, and then increases up to 95% when the Q_{in} reaches 90 L/min;

(3) $DF_{lactose-DPI}$ decreases by approximately 20% with the increase of lactose AR from 1 to 10 at Q_{in} of 30 and 39 L/min;

(4) High Q_{in} is favored in terms of enhancing the DPI-airway drug delivery efficacy, since it is the dominant factor of DF_{API} in the region G3 to G13 than lactose AR.

CHAPTER III

DISEASE-SPECIFIC AIRWAY DEFORMATION EFFECT ON INHALED AIR-PARTICLE DYNAMICS

3.1 Theory

3.1.1 Geometry and Mesh

The structural design of the airway tree is functionally vital because the branching pattern plays a role in determining airflow and particle deposition. In modeling the airways tree, this study assumes that the airway branches follow the rules of regular dichotomy after generation 3 (G3) (ICRP, 1994), i.e., each parent tube of a treelike structure gives rise to two daughter branches with identical geometric dimensions. Based on this assumption and the fact that the airflow transits from turbulence to laminar flow quickly after the trachea (Feng et al., 2018), the flow patterns at the two daughter branches are similar to each other. With such simplification, the truncated whole-lung modeling strategy (Longest et al., 2016; Tena et al., 2017) can be a feasible method to reduce the computational cost for the lung aerosol dynamics simulations from mouth/nose to alveoli without sacrificing computational accuracy.

The newly established truncated whole-lung (TWL) model presented in this chapter consists of four connected parts (see Figs. 3.1 and 3.2): (i) a representative idealized mouth-throat (MT) model (Feng et al., 2018; Longest & Xi, 2007); (ii) upper tracheobronchial (UTB) airways extending through the first generation (second bifurcations); (iii) lower tracheobronchial (LTB) airways up to G17; and (iv) heterogenous acinus. Specifically, the first three parts represent the conductive airway zone extending from the mouth to the deepest bronchioles, where the alveoli start to appear. The MT and UTB geometries were created based on the realistic airway model of the human upper airway (Feng et al., 2018) constructed from the computerized tomography (CT) data of a healthy adult (Xi & Longest, 2007). The LTB model was created using SolidWorks (Dassault Systèmes SolidWorks Corporation, Waltham, MA), assuming that the branching angles (ϕ_n's) at the same generation are the same. Such an assumption indicates that all bifurcations are geometrically symmetric. Figure 3.1 shows the schematic outline of the construction of the symmetric path model of the airway. The dimensions of the bronchi, i.e., airway radius R_n, straight segment length L_{t_n}, and branching angle ϕ_n are based on the anatomical data provided by ICRP (1994). The radius of the carinal ridge r_n is assumed to be equal to $0.5R_n$ (Tena et al., 2017). Each bifurcation was created in a different plane with an inclination angle ψ_n, as indicated by the G_n Plane and G_{n+1} Plane in Fig. 3.1. The value of ψ_n was assumed between 30 to 65 degrees (Wang, 2005), and was determined by a series of random number generated within 30 to 65 degrees. It is worth mentioning that the LTB geometry model can be fully defined with parameters R_n, L_{t_n}, ϕ_n, r_n, and ψ_n. Table 3.1 lists all the parameters used for developing the LTB airways geometry model.

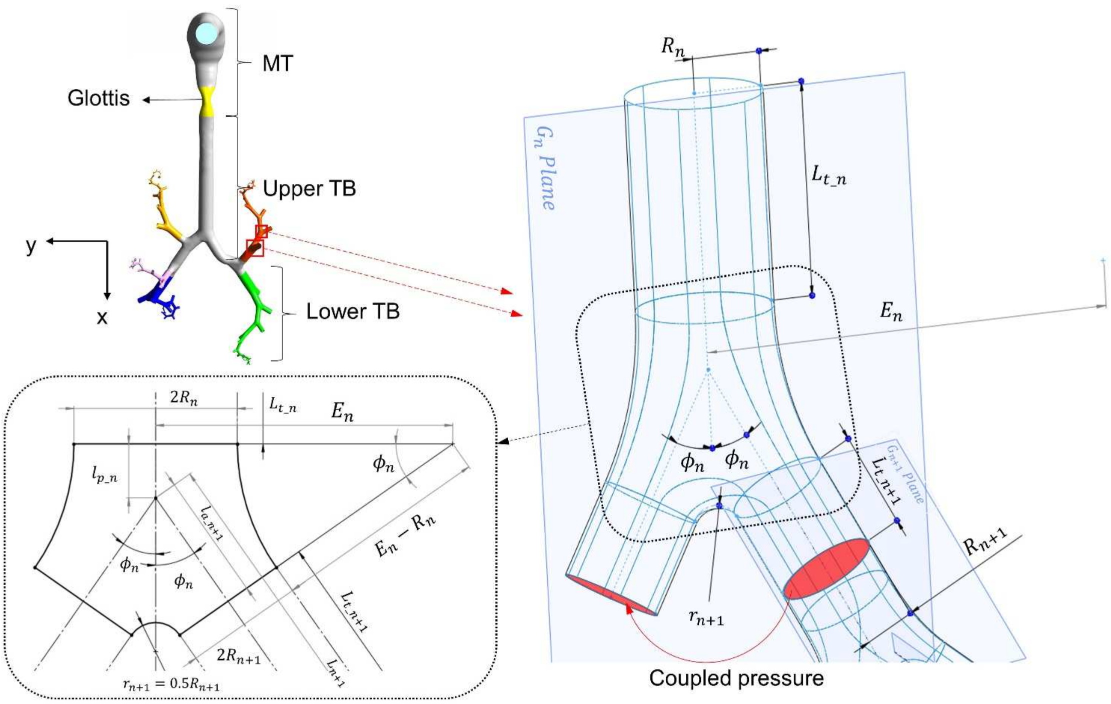

Figure 3.1: Schematic diagram of the whole-lung airway geometry and construction of the truncated symmetric path model at G_n

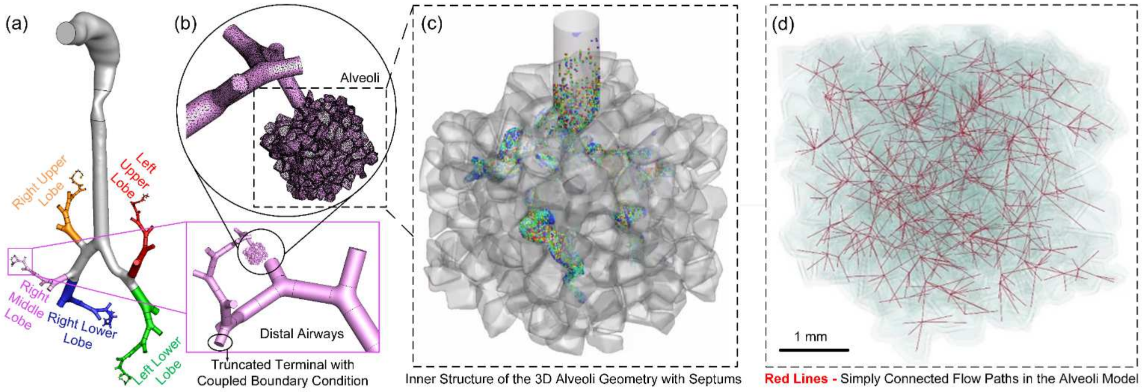
(a)
(b)
(c)
(d)
Right Upper Lobe
Left Upper Lobe
Right Middle Lobe
Right Lower Lobe
Left Lower Lobe
Alveoli
Distal Airways
Truncated Terminal with Coupled Boundary Condition
Inner Structure of the 3D Alveoli Geometry with Septums
1 mm
Red Lines - Simply Connected Flow Paths in the Alveoli Model

Figure 3.2: Schematic diagram of the reconstructed alveoli model: (a) whole-lung airway, (b) truncated small airway and alveoli structure, (c) alveoli geometry, and (d) airflow path inside alveoli structure

Table 3.1: Geometric characteristics of the respiratory tract (ICRP, 1994).

Generation	Airway radius	Straight segment length	Branching angle	Radius of carinal ridge	Inclination angle				Total branch length
G_n	R_n	L_{t_n}	ϕ_n	r_n	ψ_n	$E_n{}^{*}$	$l_{a\_n}{}^{**}$	$l_{p\_n}{}^{***}$	L_n
	(mm)	(mm)	degrees	(mm)	degrees	(mm)	(mm)	(mm)	(mm)
2	4.250	15.00	35	-	-	25.46	-	3.79	18.791
3	3.050	8.30	28	1.525	53	11.09	8.60	2.01	18.921
4	2.200	9.00	35	1.1	35.7	8.02	4.71	2.20	15.918
5	1.800	8.10	39	0.9	54.7	8.85	3.33	2.08	13.514
6	1.450	6.60	34	0.725	31.1	3.13	3.22	1.05	10.884
7	1.200	6.00	48	0.6	33.4	1.96	1.62	0.72	8.350
8	1.000	5.30	53	0.5	58.8	1.51	1.13	0.55	6.986
9	0.825	4.37	54	0.4125	41.1	1.46	0.89	0.50	5.774
10	0.675	3.62	51	0.3375	63.3	1.56	0.82	0.48	4.923
11	0.545	3.01	46	0.2725	31.2	1.19	0.78	0.37	4.173
12	0.440	2.50	47	0.22	45.4	1.18	0.61	0.48	3.602
13	0.410	2.07	48	0.205	43.4	0.54	0.55	0.12	2.750
14	0.300	1.70	52	0.15	31.6	0.87	0.35	0.31	2.365
15	0.265	1.38	45	0.1325	47.4	1.07	0.39	0.39	2.176
16	0.255	1.10	42	0.1275	32	0.57	0.42	0.23	1.761
17	0.230	0.92	50	0.115	-	-	-	-	-

* ** *** Please see the definition of E_n, l_{a_n}, and l_{p_n} in Fig. 3.1 and Section 3.1.1.

The total branch length L_n consists of three parts, a segment contained in the daughter portion of the previous bifurcation (l_{a_n}), a straight length of the generation n (L_{t_n}), and a segment contained in the parent portion of the successive bifurcation (l_{p_n}) (Tena et al., 2017) (see Fig. 3.1). For generation n (G_n), L_n can be expressed as:

$$L_n = l_{a\_n} + L_{t\_n} + l_{p\_n} \tag{3.1}$$

where

$$l_{p\_n} = E_n \tan\phi_n - \frac{E_n/\cos\phi_n - (E_n - R_n + R_{n+1})}{\sin\phi_n} \tag{3.2}$$

$$l_{a\_n} = \frac{E_{n-1}(1-\cos\phi_{n-1}) + (R_{n-1} - R_n)}{\sin\phi_{n-1}} \cos\phi_{n-1} \tag{3.3}$$

Based on the symmetric bifurcation assumption, the geometry of the LTB was reduced by truncating one of the daughter branches of each bifurcation in the model to reduce computational cost. The airflow pressure at the truncated plane is paired with the pressure of the cross-sectional plane at the corresponding location of the paring daughter branch.

Table 3.2: Summary of the structure information of the heterogeneous acinar model.

Number of alveoli	406
Minimum generation	3
Maximum generation	11
Mean generation	6.7

The acinus model was created based on the algorithm developed by Koshiyama and Wada (2015). An illustration of the acinar structure and its dimensions are shown in Fig. 3.2. Specifically, the average volume of the five acini (one for each lobe) at residual volume is 6.2e-9 m^3. The acinar geometry contains 406 alveoli with a mean generation of 6.7 (see Table 3.2). Such a geometric structure was constructed to investigate the role of the deforming acinar on particle transport and deposition outcomes.

The tetrahedral mesh with prism layers was generated using Ansys Fluent Meshing 2020 R2 (Ansys Inc., Canonsburg, PA). Considering the high computational cost for running the whole-lung model simulation, the mesh independence test was performed separately for the truncated whole-lung geometry without acinus and single acinus geometry. For both geometries, three meshes were generated. First, a steady inhalation simulation was performed with the truncated whole-lung geometry without acinus. The inhalation flow rate was set to 90 L/min, which is higher than the maximum flow rate (72 L/min) that can be reached in the elastic whole-lung model simulation cases. The area-averaged total pressure at each truncated outlet of five lobes (i.e., a total of 70 outlets) was compared, and the results are shown in Fig. 3.3 (see Table 3.3 for the correspondence of *x*-axis of Fig. 3.3). The comparison demonstrates that Mesh 2 (approximately 7.9 million cells) of the TWL model can accurately predict the pressure drop while reducing the computational cost. Therefore, Mesh 2 was selected as the final mesh. For the single acinus geometry, CFD simulations were also performed using three meshes (see Table 3.4 for mesh details). Figure 3.4 compares the velocity magnitudes at a sagittal plane and a specific line. Results indicate that Mesh 2 of the single acinus geometry with approximately 4.8 million cells is the best choice among those three meshes, which provided the optimal balance between computational accuracy and efficiency. Accordingly, the final mesh for the elastic whole-lung model with acinus (see Fig. 3.2) has a total of 31,867,870 cells and minimum orthogonal quality of 0.12.

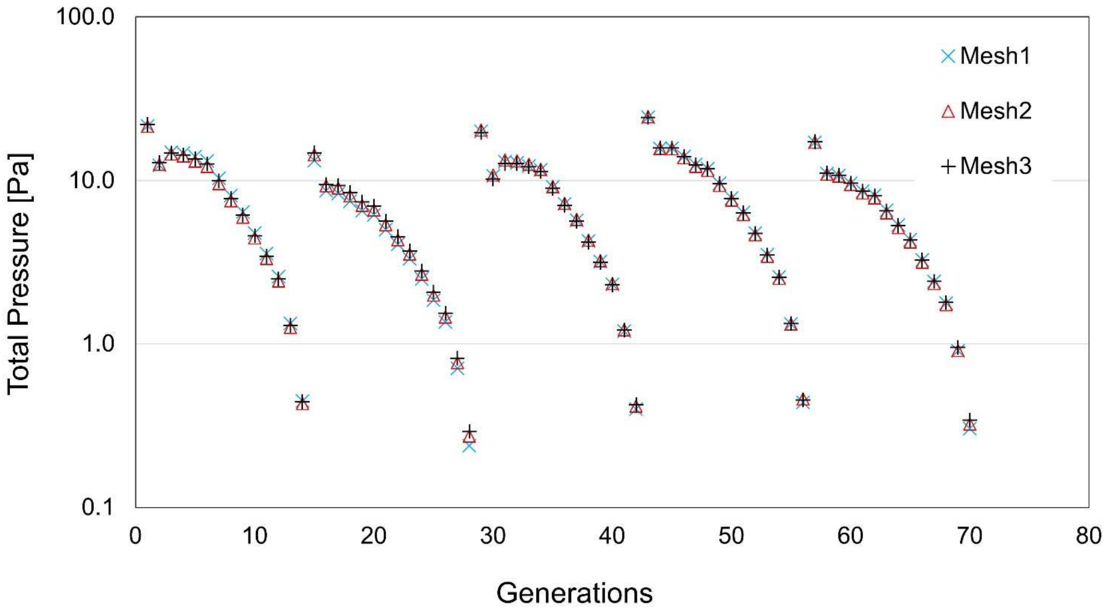

Figure 3.3: Total pressure at each truncated outlet of five lobes with an average inhalation flow rate of 90 L/min

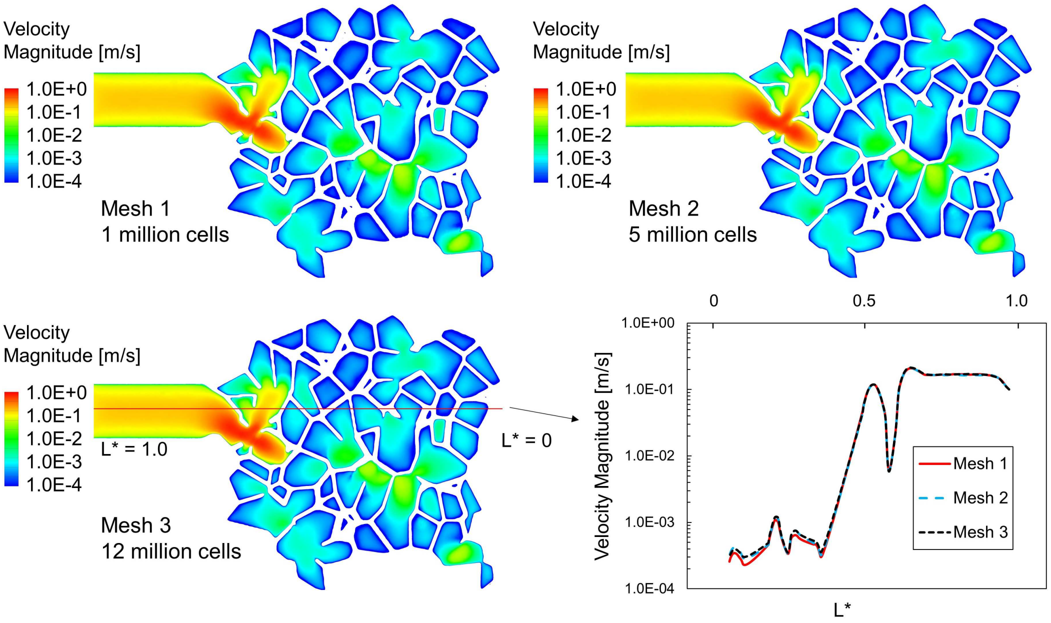

Figure 3.4: Velocity magnitude contour at the sagittal plane of acinus model

Table 3.3: Correspondence between generations and numbers (*x*-axis in Fig. 3.3).

No.	Gen.	No.	Gen.	No.	Gen.	No.	Gen.	No.	Gen.
1	lll-G4	15	lul-G4	29	rll-G4	43	rml-G5	57	rul-G5
2	lll-G5	16	lul-G5	30	rll-G5	44	rml-G6	58	rul-G6
…	…	…	…	…	…	…	…	…	…
14	lll-G17	28	lul-G17	42	rll-G17	56	rml-G18	70	rul-G18

Table 3.4: Mesh information (cell number in million) of the TWL model without acinus and the acinus model.

	Mesh1	Mesh2	Mesh3
TWL model without acinus	4.8	7.9	16.0
Single acinus model	1.2	4.8	11.9

3.1.2 Generalized Function for the Prescribed Airway Deformation Kinematics

The lung expansion and contraction motion during a full breathing cycle are shown in Fig 3.5. Such anisotropic airway deformation kinematics of the elastic whole-lung model is mathematically achieved using dynamic mesh in ANSYS Fluent environment customized via user-defined functions (UDFs). Specifically, the airway wall from the trachea to G17 expands and constricts in all three directions (arm-arm (*y*) direction, head-foot (*x*) and back-front (*z*) directions) with a deformation ratio of *x*:y:*z* =1:1:0.375 (Plathow et al., 2004; Xi et al., 2018). The glottis expands and constricts in *y*-direction only (Zhao et al., 2020). The generalized function that controls the nodal displacements of the surface mesh cells can be given by

$$x_i^n = x_{i,r} + \frac{f_t(t^n)}{f_t(t^{n-1})} f_s(x_i^{n-1})\left(x_i^{n-1} - x_{i,r}\right) \tag{3.4}$$

$$f_t(t^n) = 1 + \frac{d_{t,i}}{2}\left(1 - \cos\frac{2\pi t^n}{T_c}\right) \tag{3.5}$$

$$f_s(x_i^n) = \begin{cases} 0.5\left[1 - \cos\frac{(x_i^n - x_0)\pi t^{n-1}}{x_1 - x_0}\right] & trachea\ when\ x_i = x \\ 0.5\left[1 - \cos\frac{(x_i^n - x_1)\pi t^{n-1}}{x_0 - x_1}\right] & trachea\ when\ x_i = y, z \\ 1 \quad other\ parts & \end{cases} \tag{3.6}$$

where $x_i = (x, y, z)$ are the axis coordinates of each node within the dynamic region, $x_{i,r} = (x_r, y_r, z_r)$ is the reference point, t^n is the current time step, T_c is the time period for a full breathing cycle, and $d_{t,i}$ are the deformation ratios of airways in three directions in Cartesian coordinate which are disease-specific. To achieve a smooth transition from the location where the expansion and contraction starts at the trachea to the first bifurcation, an extra term $f_s(x_i^n)$ was added to Eq. (3.4) to control the motion of the trachea. The equations that guide the motion of glottis and corresponding numerical investigation results can be found in the previous publication (Zhao et al., 2020). Specifically, the glottis motion functions are expressed as

$$y(x,t) = (d_{g,r} - 1)f(x)g(t) + y_{r,0} \tag{3.7}$$

$$f(x) = \sin^m\left(\frac{x(t) - x_1}{x_2 - x_1}\pi\right) \tag{3.8}$$

$$g(t) = a_0 + \sum_{\beta=1}^{n}\left[a_\beta \cos(\beta\omega t) + b_\beta \sin(\beta\omega t)\right] \tag{3.9}$$

where $y_{r,0}$ is the initial y-coordinate of the node, and $d_{g,r}$ is the deformation ratio of glottis between maximum glottis width and the width of the glottis at the neutral position as shown in Fig. 3.5 (b) . In addition, the nodal displacement function $g(t)$ is a time-dependent Fourier series that controls the nodal motion separately. It is worth mentioning that $g(t)$ is simplified as a single-term sinusoidal function, which is employed to simulate the idealized glottal motion in this study (see Fig. 3.5 (b)).

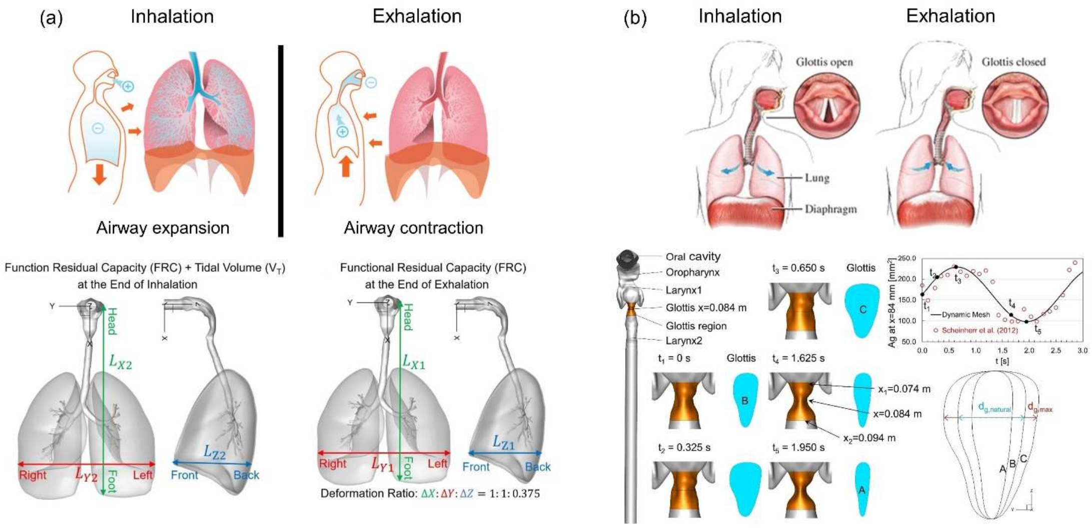

Figure 3.5: The deformation kinematics of (a) tracheobronchial (TB) tree and (b) glottis in a full inhalation-exhalation cycle

Data source:

[a] https://courses.lumenlearning.com/boundless-biology/chapter/breathing/

[b] Manzoor, Tariq & Muneer, Faiza & Ibrahim, Muhammad & Tahira, Sadia & Azmat, Rabia. (2015). Effectiveness Of Voice Therapy In Dysphonia: A Speech-Pathologist Perspective.

By manipulating the values of $d_{t,i}$, the TWL model can achieve the disease-specific airway deformation kinematics. Specifically, the values of $d_{t,i}$ and its related lung conditions are listed in Table 3.5. The validation of the airway deformation kinematics is documented in Section 3.3.1.

Table 3.5: Deformation ratio of airways for different lung conditions.

$d_{t,i}$	0.4	0.36	0.2
Lung condition	Healthy	Mild COPD	Severe COPD

3.1.3 Computational Fluid Particle Dynamics (CFPD) Model

CFPD models have been widely employed for modeling the airflow and micro/nano particle transport and deposition in human respiratory systems. This study used Ansys

Fluent 2020 R2 (Ansys Inc., Canonsburg, PA) enhanced with in-house UDFs to predict the particle dynamics in the laminar-to-turbulence flow fields inside the airways.

3.1.3.1 Continuous Phase: Pulmonary Airflow

In this study, airflow is assumed to be isothermal and incompressible (ρ = 1.204 kg/m^3), with a dynamic viscosity μ = 1.825e-5 Pa·s. The continuity and Navier-Stokes (N-S) equations with moving boundaries can be given by:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{3.10}$$

$$\frac{\partial u_i}{\partial t} + \left(u_j - u_j^{mov}\right)\frac{\partial u_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\mu}{\rho}\frac{\partial \tau_{ij}}{\partial x_j} + g_i \tag{3.11}$$

$$\tau_{ij} = \mu\left[\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial u_k}{\partial x_k}\right] \tag{3.12}$$

where δ_{ij} is the Kronecker delta. The convective velocity $u_i - u_i^{mov}$ in Eq. (3.11) is induced by the difference between the air velocity u_i and the dynamic mesh velocity u_i^{mov} induced by the airway deformation. u_i^{mov} can be given by

$$u_i^{mov} = \partial\, x_i/\partial\, t \tag{3.13}$$

where x_i from the trachea to alveoli can be obtained from Eq. (3.4) and x_i for the moving glottis can be obtained from Eq. (3.7). The turbulence characteristics of the pulmonary airflow are modeled using k-ω Shear Stress Transport (SST) model, which has been extensively validated in previous works (Feng & Kleinstreuer, 2013; Feng et al., 2018; Zhang et al., 2012).

3.1.3.2 Discrete Phase: Inhaled Particle Transport Dynamics

The one-way coupled Euler-Lagrange approach has been widely used and experimentally validated for pulmonary particle-laden airflow predictions (Feng et al., 2018; Koullapis et al., 2018; Zhang et al., 2012). Particles are assumed to be spheres with constant aerodynamic diameter. In this study, particles with different diameters, i.e., d_p= 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, and 10.0 μm, are investigated. The velocity and trajectory of every single particle are calculated by solving Newton's second law, which considering the drag force, gravitational force, random force induced by Brownian motion and the force induced by turbulence dispersion (Feng et al., 2018; Zhao et al., 2020). Furthermore, the regional deposition of particles in the airways can be calculated by regional deposition fraction (RDF), i.e.,

$$RDF_{specific\ region} = \frac{Number\ of\ particles\ deposited\ in\ a\ specific\ region}{Number\ of\ particles\ injected\ through\ the\ mouth\ inlet} \quad (3.14)$$

3.1.3.3 Boundary and Initial Conditions

The starting time and initial conditions of the airway model are at the end of a previous inhalation-exhalation cycle, which mimics the inhalation of aerosolized drug particles in real-world inhalation therapy scenarios. At the end of exhalation, the lung capacity is equal to the residual volume defined in the pulmonary function test (PFT). The pressure of the truncated branch outlet is coupled with the pressure of the identical surface at its paired daughter branch (see Fig. 3.1). A full breathing cycle of 2 seconds is simulated, including both inhalation and exhalation. The breathing profile at the mouth is determined only by the lung deformation kinematics. Accordingly, the pressure-inlet boundary condition is specified at the mouth opening. A total of 50,000 particles are released at the mouth from time t = 0.2s to 0.25s, which is aligned with the duration of drug particle emissions from inhalers (Tian et al., 2015). Particles are considered "deposited", when the distance between the center of the particle and the airway wall is less than the particle radius.

3.2 Numerical Setup

The numerical approach, i.e., the elastic TWL model used in this study, is based on the in-house prescribed dynamic mesh method (Zhao et al., 2020), one-way coupled Euler-Lagrange method (Feng et al., 2018), and k-ω RANS model, which enables the predictions of anisotropic airway deformation and air-particle flows in the whole-lung in tandem where turbulent, transitional, and laminar flows coexist. To realize the algorithm mentioned above, in-house user-defined functions (UDFs) were developed and compiled for:

(1) Specifying the airway deformation kinematics;

(2) Specifying the coupled pressure boundary conditions at truncated branch outlets;

(3) Recovering the anisotropic corrections on turbulence fluctuation velocities;

(4) Modeling the Brownian motion induced forces;

(5) Storing particle deposition data.

The CFPD simulations were executed using Ansys Fluent 2020 R2 (Ansys Inc., Canonsburg, PA) and performed on a local Dell Precision T7910 workstation (Intel® Xeon® Processor E5-2683 v4 with dual processors, 32 cores, and 256 GB RAM), the supercomputer “Pete” at the High Performance Computing Center (HPCC) at Oklahoma State University (OSU) (Intel® Xeon® Processor Gold 6130 CPU with dual processors, 32 cores, 64 threads, and 96 GB RAM), and Microsoft Azure (120 AMD EPYC 7V12 processor cores with 4 GB RAM per CPU core) enabled by Ansys Cloud Computing COVID-19 HPC Consortium. The Semi-Implicit method for pressure-linked equations (SIMPLE) algorithm was employed for the pressure-velocity coupling, and the least-squares cell-based scheme was applied to calculate the cell gradient. The second-order scheme was employed for pressure discretization. In addition, the second-order upwind scheme was applied for the discretization of momentum and turbulent kinetic energy. Convergence is defined for continuity, momentum, and supplementary equations when residuals are lower than 1.0e-5.

3.3 Model Calibration and Validation

3.3.1 Validation and Calibration of the Elastic TWL Model

The elastic TWL model was first validated by comparing the total lung volume change during a full breathing cycle predicted by the numerical method with experimentally measured results from the literature (Feher, 2017; Hall, 2015; "Respiratory Physiology," 2019) (see Fig. 3.6). It should be noted that the initial lung volume equals residual volume (RV) (see Fig. 3.7 for the definition of RV). Moreover, to calculate the whole lung volume of the elastic TWL model, the acinus volume is multiplied by 2^{15} (i.e., 15 generations were truncated) to recover the total volume of a whole lung. The total lung volume through breathing matches well with the data in the open literature. Thus, the generalized airway deformation function and the elastic TWL model (see Eqs. (3.4) to (3.6)) have been proved to be able to capture the deformation kinematics of a real human respiratory system.

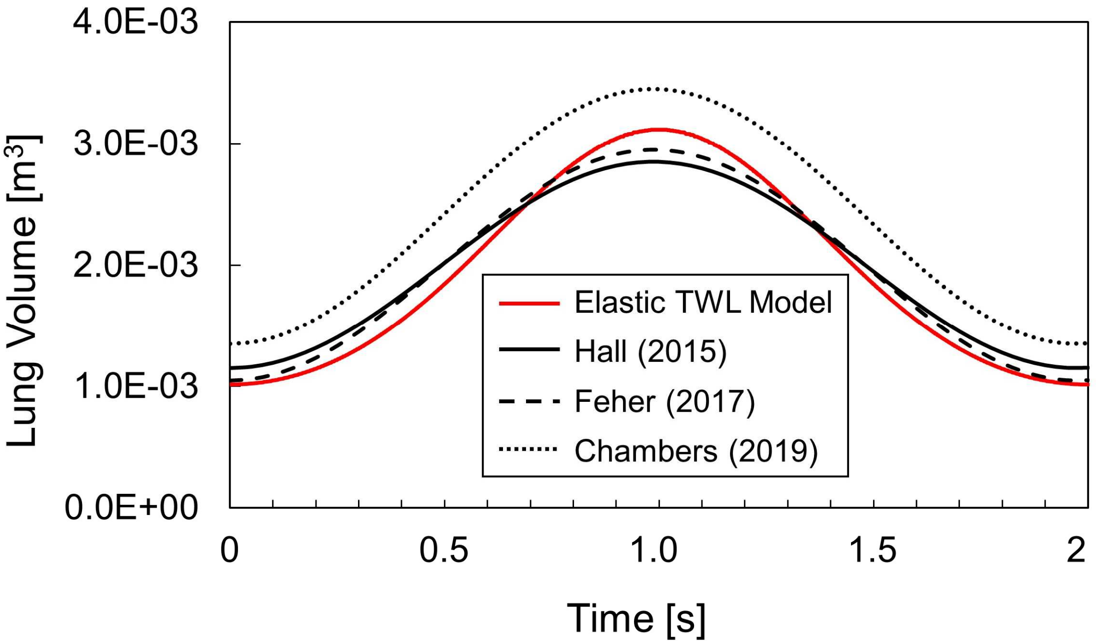

Figure 3.6: Validation of the elastic TWL model: comparison of the total lung volume of the elastic TWL model and experimentally measured data

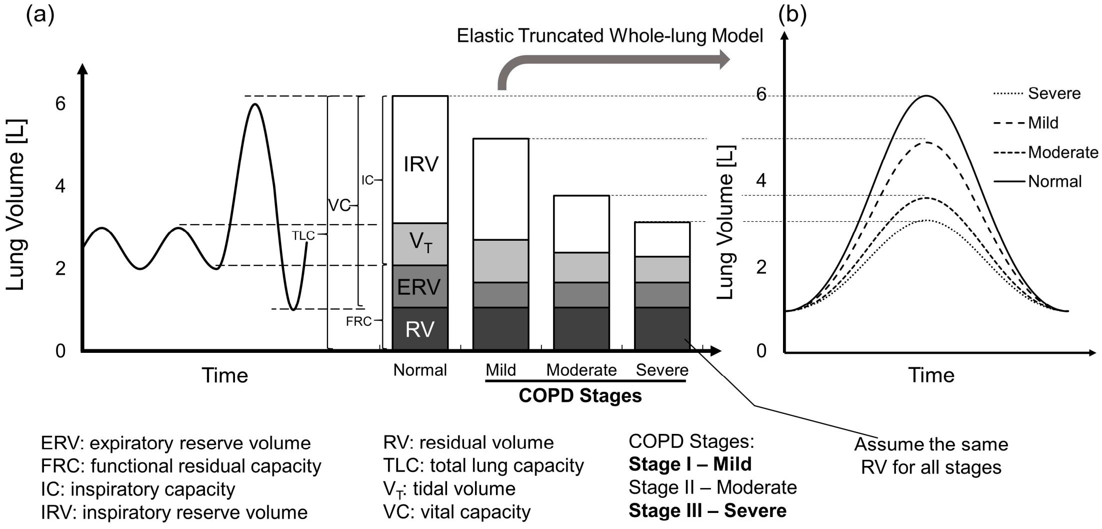

Figure 3.7: Lung volume change calibrations for the elastic TWL model via matching pulmonary function test data for different lung disease conditions

To achieve the disease-specific airway deformation kinematics, the elastic TWL model was calibrated by varying the values of $d_{t,i}$ (see Eq. (3.5)). Specifically, the values of $d_{t,i}$ are determined by matching the total lung capacity (TLC) under two COPD conditions, i.e., mild and severe COPD, as well as the TLC of a healthy lung. It should be noted that, to simplify the problem in this study, the lung residual volume (RV) is assumed to be the same for healthy and diseased lungs. Lung volumes under different health conditions, including one healthy condition and three COPD stages (Mead et al., 1967) are given in Fig. 3.7 (a). Correspondingly, the lung volume changes calculated using the elastic TWL model is given in Fig. 3.7 (b). It is evident that the numerically predicted TLC under different lung conditions matches very well with the clinical data from the literature (Mead et al., 1967). The value of $d_{t,i}$ for different lung conditions is given in Table 3.5.

3.3.2 CFPD Model Validation

The k-ω SST model has been extensively validated and exploited in previous research to resolve the flow field based on its ability to predict pressure drop, velocity profiles accurately, and shear stress for both transitional and turbulent flows (Feng & Kleinstreuer, 2013; Feng et al., 2018; Zhang et al., 2012). The one-way coupled Euler-Lagrange method was also well-proved with *in vitro* and *in vivo* data in the previous research for accurate predictions of the aerosol dynamics in human respiratory systems (Feng et al., 2018; Longest et al., 2016; Walenga et al., 2013; Xi et al., 2016).

In this study, the particle deposition fraction (DF) predicted using the static TWL model at a steady inhalation flow rate of 30 L/min was compared with both numerically predicted and experimentally measured data from open literature (Longest et al., 2016; Stahlhofen et al., 1989). Table 3.6 compares the total DF of particles with d_p=1.0, 2.0 and 5.0 µm. In general, the total DF either predicted by numerical methods or measured experimentally follow the same trend as d_p increases from 1.0 to 5.0 μm. It can be noticed that the static TWL model predicts slightly lower total DF for all three sizes of particles tested compared with literature data. This difference in total DF could be related to the different airway structures used in the three studies.

Table 3.6: Total lung deposition fraction (DF) comparison with data from Longest et al. (2015) and Stahlhofen et al. (1989).

d_p [μm]	Static TWL	Longest et al. (2015)	Stahlhofen et al. (1989)
1.0	17.5%	32.8%	24.2%
2.0	38.4%	44.2%	45.3%.
5.0	71.5%	75.4%	81.0%

3.4 Results and Discussion

3.4.1 Airflow Characteristics: Static TWL vs. Elastic TWL

The objective of Section 3.4.1 is to determine whether the widely used static virtual lung modeling framework can predict pulmonary airflow field similar to the more physiologically realistic elastic TWL model. The widely used static lung modeling framework has two major differences compared with the elastic TWL model: (1) velocity mouth/nose inlet condition instead of realistic pressure boundary conditions; and (2) neglected glottis and TB tree deformation kinematics. To compare the airflow fields, one full breathing cycle was simulated for three lung conditions, i.e., normal, mild COPD, and severe COPD, using the elastic TWL model. The static TWL model was also employed to predict the airflow structure for those three lung conditions, with sinusoidal breathing mass flow rate waveforms applied at the mouth opening. The sinusoidal functions were derived for providing the equivalent lung volume changes to the elastic TWL modeling results. The inspiratory airflow structures at the sagittal plane are given in Figs. 3.8 and 3.9. It should be noted that the normalized velocity $\|\widetilde{\vec{V}}\|$ is dimensionalized using the averaged velocity at mouth opening at $t=\frac{1}{4}T_c$, where T_c is the period of breathing cycles. In other words, the velocity is normalized with the area-averaged velocity at mouth entrance occurred at the peak inhalation flow rate. Figure 3.8 shows the normalized velocity contour at the sagittal plane (y=0) at $t=\frac{1}{8}T_c$ and $t=\frac{1}{4}T_c$. All six cases show similar inspiratory airflow structure, except that the elastic TWL model predicts relatively weaker convection in the glottis region than the static TWL model for all three lung conditions. In addition, the elastic TWL model predicts weaker convection in the oropharynx for severe COPD conditions compared with normal and mild COPD conditions. The flow pattern during inhalation changes significantly as the flow rate reaches its peak value. To be specific, the mouth jet and laryngeal jet become much stronger at $t=\frac{1}{4}T_c$ than $t=\frac{1}{8}T_c$.

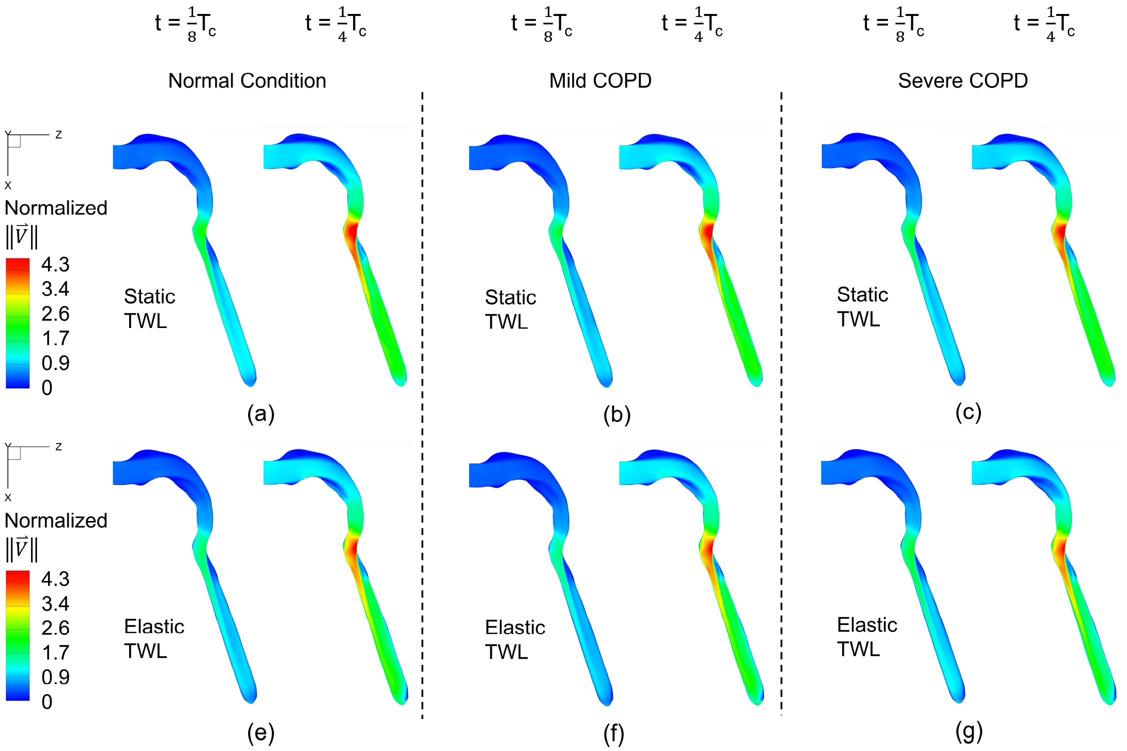

Figure 3.8: Normalized velocity magnitude contour at a sagittal plane y=0: (a) static model with normal condition, (b) static model with mild COPD, (c) static model with severe COPD, (d) elastic model with the normal condition, (e) elastic model with mild COPD condition, and (f) elastic model with severe COPD condition

To visualize the lung deformation effect on airflow pattern in MT, trachea, and G1 to G3, $\|\widetilde{\vec{V}}\|$ contour and tangential velocity vector on selected slices AA' to EE' at $t=\frac{1}{4}T_c$ are given in Fig. 3.9. Specifically, the flow structures shown in slice AA' are similar for all six cases, with no evident secondary flow observed. This indicates that the flow rates investigated in this study are not strong enough to alter the flow structure in the oropharynx. When it comes to the throat, one can notice the glottis expansion in elastic TWL model cases. As a result of the glottis expansion, different airflow patterns can be observed on slice BB'. For example, for normal conditions, although both static and elastic models predict counterclockwise in-plane recirculation zones near the center of BB' slice, the vortices locations are more to the left side in elastic lung cases. Also, the secondary flow has different directions on the top left corner of slice BB'. In addition, $\|\widetilde{\vec{V}}\|$ on slices CC' and DD' shows the flow structure in the trachea. It can be seen from slice CC', two counter r-rotating vortices are formed at the center of slice CC' in the static model, while only one counterclockwise vortex can be observed in the elastic model. Moreover, $\|\widetilde{\vec{V}}\|$ contour on slice CC' shows that the elastic TWL model predicts higher $\|\widetilde{\vec{V}}\|$ at the anterior of the trachea (i.e., bottom of slice CC') for normal and severe COPD conditions than the other cases. In slice DD', the counterclockwise secondary flow existing upstream is diminished and challenging to be observed. As the flow enters the first bifurcation, the airflow structure appears to be apparently different between static and elastic TWL models. For the static TWL model, vortices can be found on both left and right sides of slice EE'. However, in the elastic TWL model, the vortices shift to the top-right and bottom left of slice EE'. After the third bifurcation (G4), the airflow structure is affected by lung deformation kinematics and the inhalation flow rate (lung conditions). Specifically, for slice FF', although Dean's flows can be observed in all cases, the predicted location and number of the vortices differs between static and elastic TWL models. It can be concluded that the airflow structure in the upper airway is weakly affected by the lung deformation kinematics. However, as the airflow reaches the main bronchus, the effect of lung deformation kinematics on airflow structure becomes manifest.

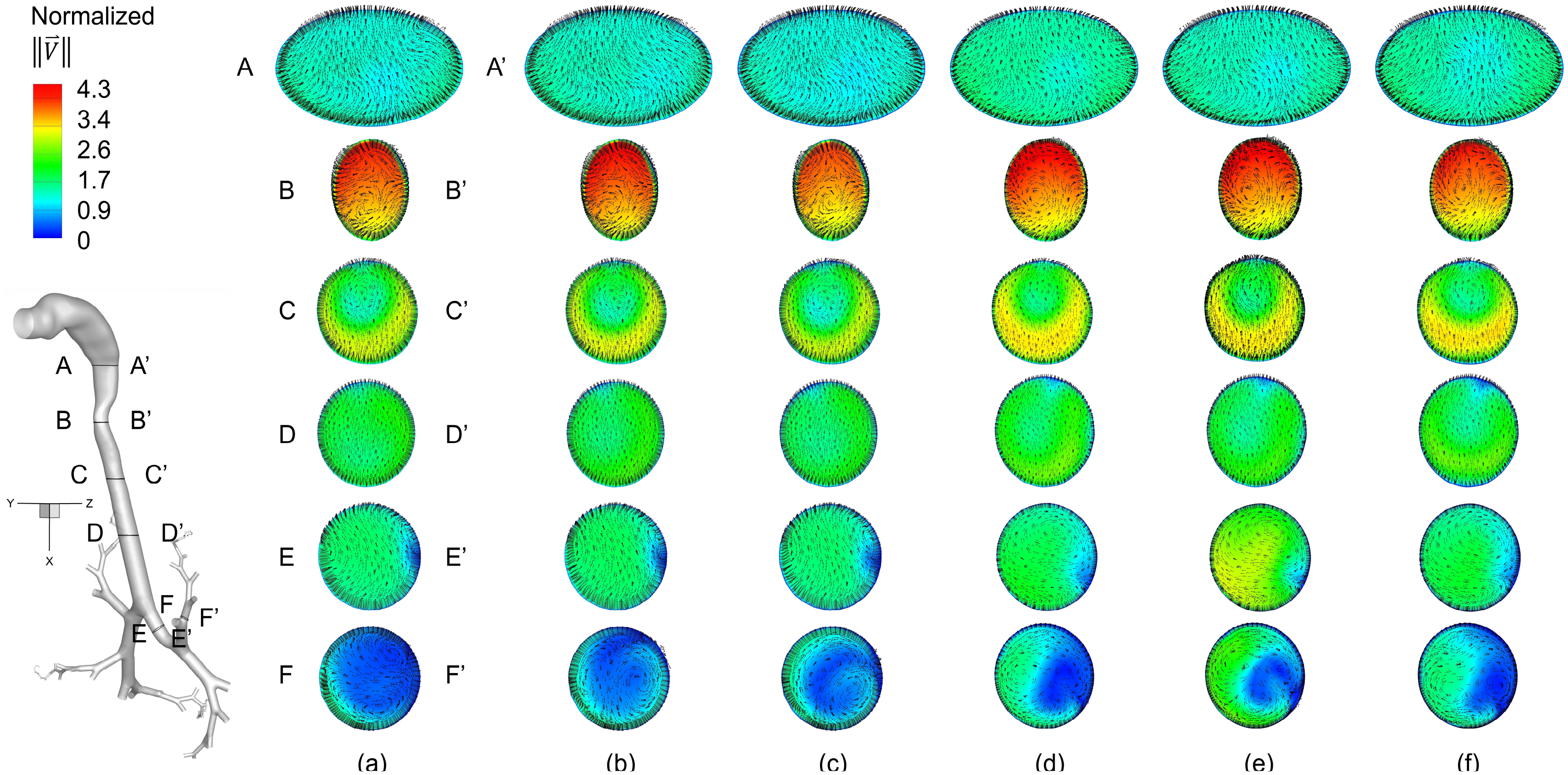

Figure 3.9: Normalized velocity magnitude contour and tangential velocity vector on selected slices at $t=\frac{1}{4}T_c$ for cases: (a) static model with normal condition, (b) static model with mild COPD, (c) static model with severe COPD, (d) elastic model with normal condition, (e) elastic model with mild COPD, and (f) elastic model with severe COPD

3.4.2 Effect of Airway Deformation Kinematics on Particle Transport and Deposition

To further investigate how the neglected airway deformation kinematics can influence the predictions of lung aerosol dynamics, the transport and deposition of particles with d_p=0.1, 0.2, 0.5, 1.0, 2.0, 5.0 and 10.0 µm in the TWL model are investigated individually under the above mentioned three lung conditions. The deposition distribution of particles with d_p=0.1, 1.0, and 10.0 µm, in both the static and elastic TWL models after one full breathing cycle are visualized in Fig. 3.10. For all the cases shown in Fig. 3.10, deposition occurs in the throat, main bronchus, and first three bifurcations. However, the particle deposition distribution predicted by static and elastic TWL models differs a lot. For example, at the normal lung condition, for d_p=0.1, 1.0 and 10 µm, particles are more likely to be entrapped in the trachea in the static TWL model compared with the elastic TWL model. Previous research demonstrates that Brownian motion induced force has a strong impact on the transport and deposition of small particles (e.g., d_p<0.5 µm), even though the inertia effect on small particles is negligible. This explains the deposition of 0.1 µm particles in the trachea for the static TWL model. The static TWL modeling strategy predicted a significantly higher deposition in the trachea for 1.0 µm particles than what can be observed in the elastic TWL model (see Figs. 3.10 (a) and (d)). The deposition of 0.1 and 1.0 µm particles is hard to observe in the elastic TWL model (see Fig. 3.10 (d)). The deposition differences in the trachea between static and elastic TWL models are due to the different intensities of the secondary flow observed in Fig. 3.9 (a) slices BB' and CC'. The possible reason is that because the airway goes through expanding during the inhalation, the chances of the trachea being hit by particles are reduced significantly. This speculation can be supported by the observation on the deposition distributions of 10.0 μm particles shown in Figs. 3.10 (a) and (d). To be specific, when compared with the static TWL model, although a lower deposition concentration of 10.0 μm particles in the trachea is observed in the elastic TWL model, the deposition concentration is higher in the first two bifurcations of right lobes. It seems that the lung expansion delays the deposition of particles with d_p=10 μm. In addition, the static TWL model predicts much higher deposition concentration in MT of large particles (d_p =10 μm) than elastic TWL model. This observation agrees with the findings of the authors' previous research on the effect of glottis motion on particle transport and deposition in an upper airway model (Zhao, 2020). Similar results can be seen from mild and severe COPD cases.

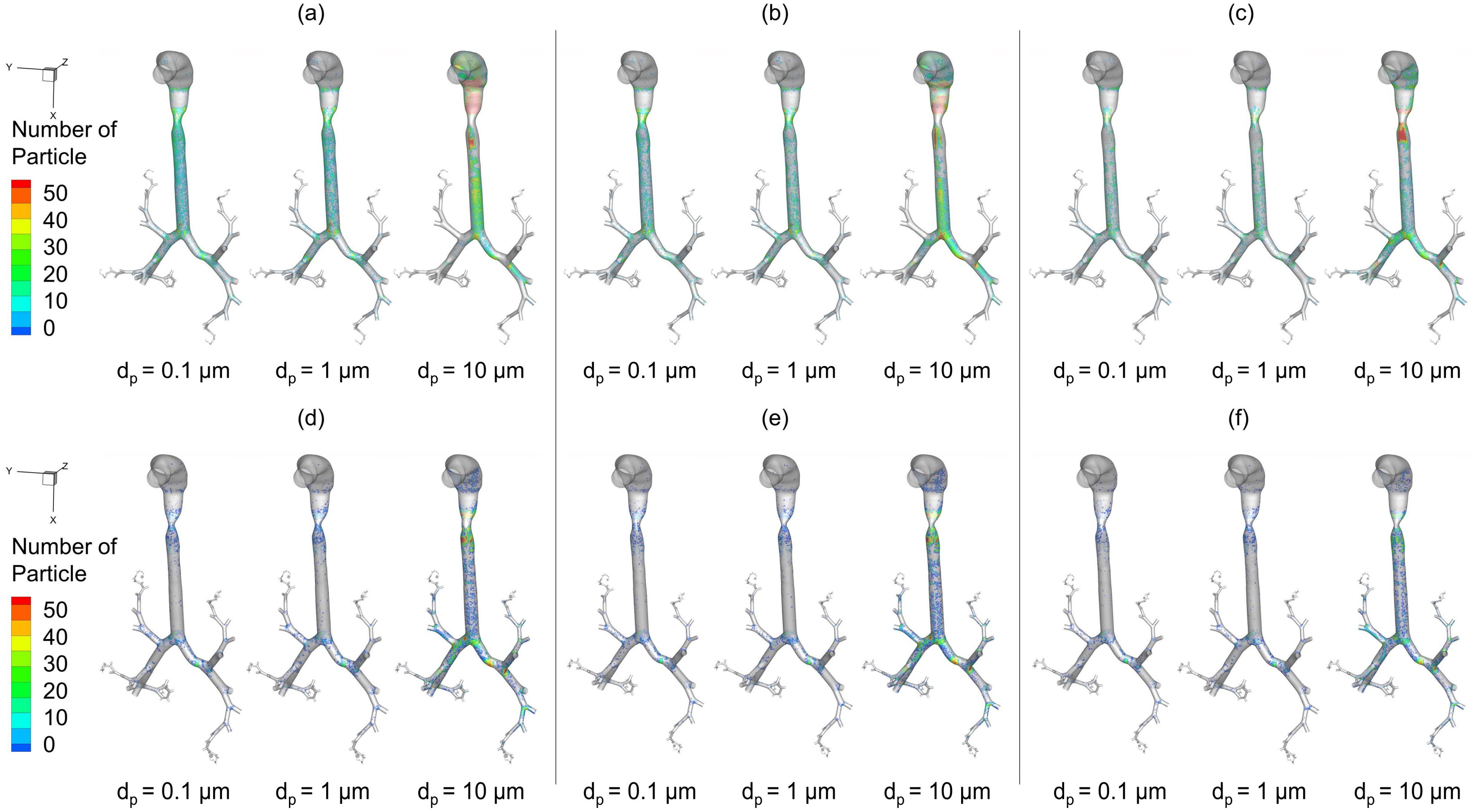

Figure 3.10: Lung deposition distribution of particles with sizes of 0.1, 1.0 and 10 μm: (a) static model with the normal condition, (b) static model with mild COPD, (c) static model with severe COPD, (d) elastic model with the normal condition, (e) elastic model with mild COPD, and (f) elastic model with severe COPD

3.4.3 Effect of Airway Deformation and Disease Conditions on Total DF

The effect of lung deformation on particle deposition is also analyzed by comparing the total deposition fraction (DF) of particles with the diameter ranging from 0.1 to 10 μm under different lung health conditions. In general, both static and elastic TWL models predict a parabolic trend between total DF and d_p. This is demonstrated extensively in previous research. For lungs under normal condition, the static TWL model predicts 13.4% higher total DF of particles with d_p =0.1 μm than the elastic TWL model. For particle size ranging from 0.2 to 2.0 μm, the total DFs generated with static and elastic TWL models are in good agreement (difference within 7%). However, as particle size increases to 5.0 and 10.0 μm, the static model predicts 16.9% and 13.1% less total DFs than the elastic model, respectively. For mild COPD conditions, the difference in total DF predicted by static and elastic models is not obvious. Specifically, the highest difference is 5.1%, as the elastic model generates a higher total DF for particles with d_p=0.2 μm than the static model. For the severe COPD condition, both static and elastic models predict similar total DF for small (d_p=0.1 and 0.2 μm) and large (d_p=10 μm) particles. However, for particles with d_p between 0.5 and 5 μm, the static model gives lower total DF than the elastic model. Especially for d_p=2 μm, the static model predicts 16% lower total DF than the elastic model. It can be concluded that a static TWL model can be used for predicting the total DF of particles (0.1<d_p<10 μm) for airways under mild COPD condition. However, regional DF also needs to be analyzed to gain better insight into the effect of airway deformation on particle dynamics in the respiratory system.

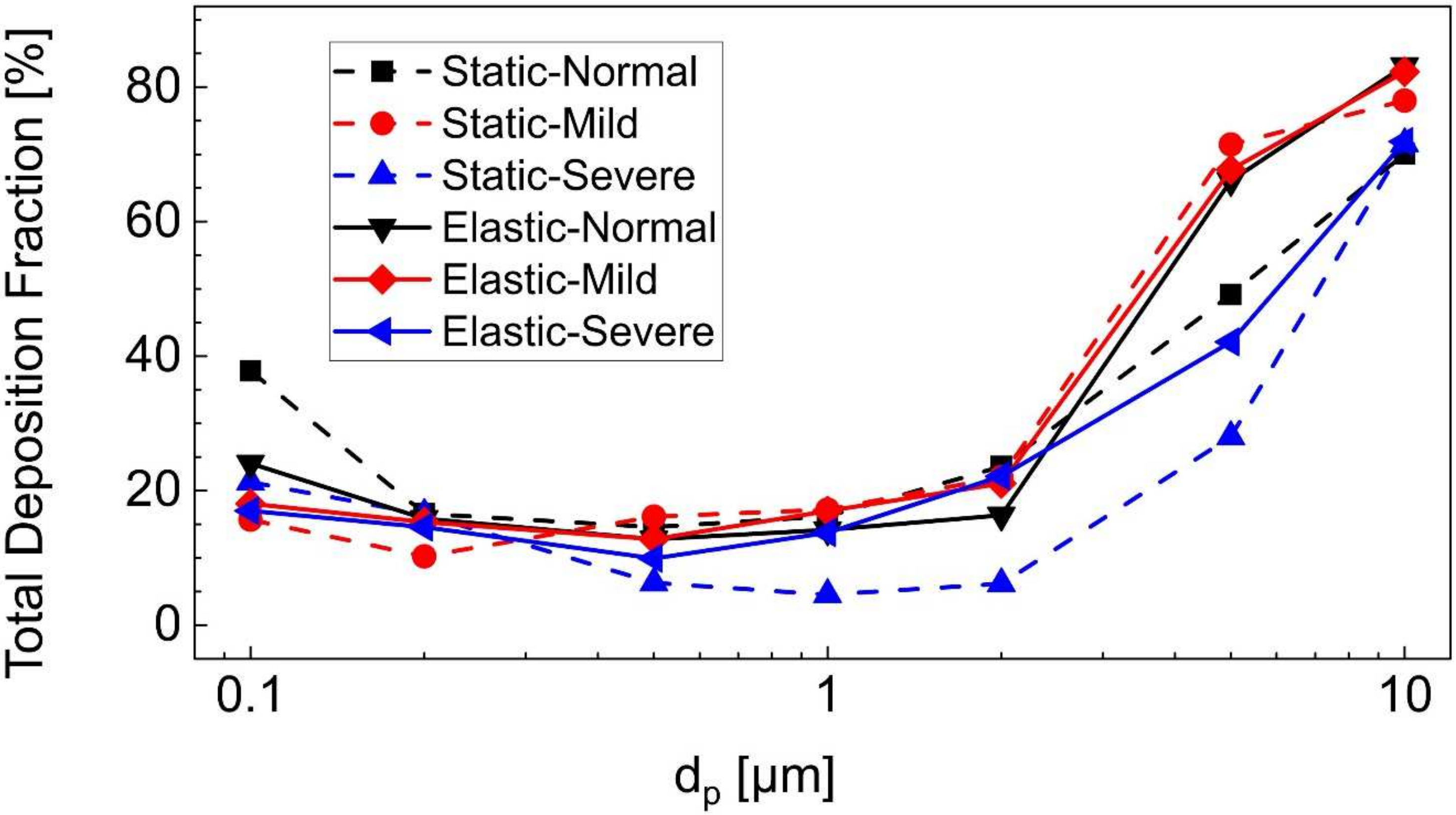

Figure 3.11: Total deposition fractions (DF) of particles with the diameter ranging from 0.1 to 10 μm under different lung health conditions

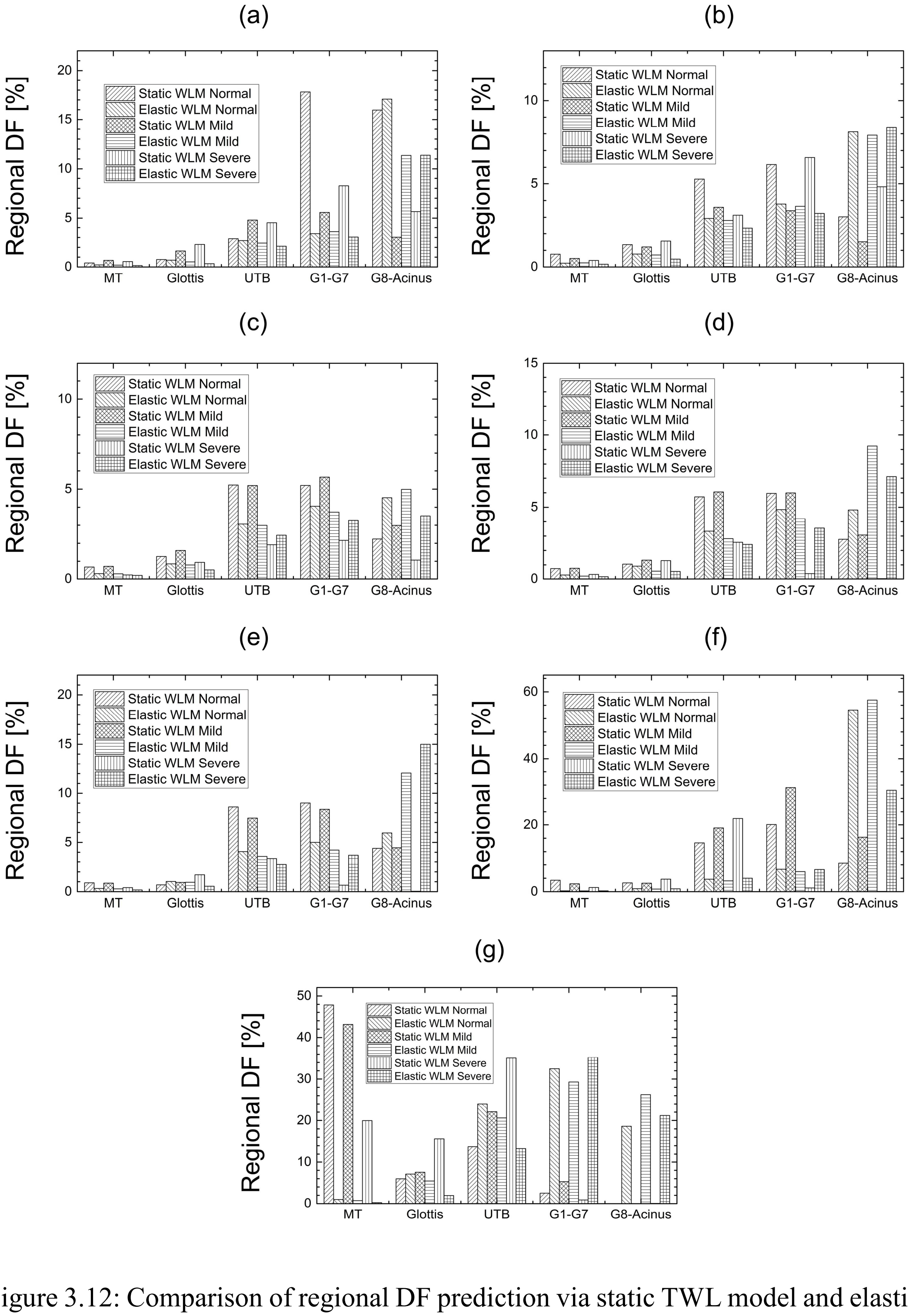

Figure 3.12: Comparison of regional DF prediction via static TWL model and elastic TWL model under three lung health conditions for particles with different diameters: (a) d_p=0.1 μm, (b) d_p=0.2 μm, (c) d_p=0.5 μm, (d) d_p=1.0 μm, (e) d_p=2.0 μm, (f) d_p=5.0 μm, and (g) d_p=10 μm

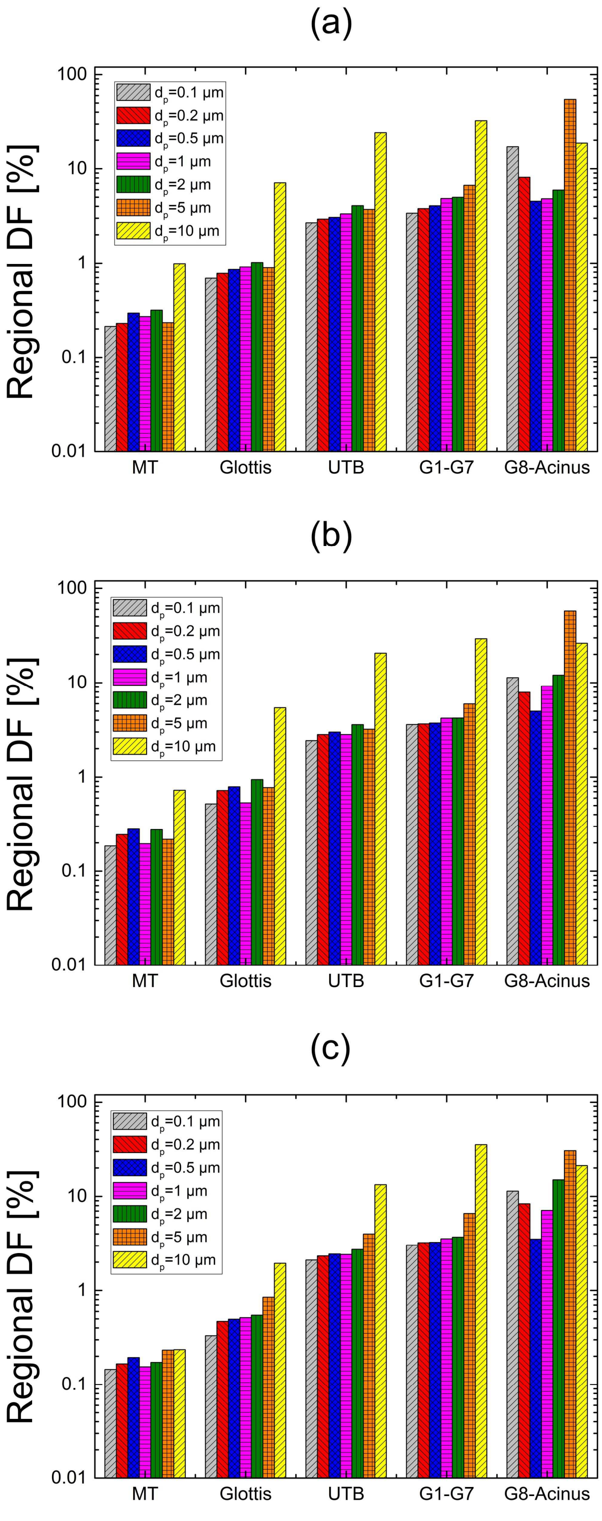

Figure 3.13: Comparison of regional DF predicted via elastic TWL model under different lung conditions: (a) normal, (b) mild COPD, and (c) severe COPD

3.4.4 Effect of Airway Deformation Kinematics on Regional DF

To better understand the effect of airway deformation kinematics on particle dynamics in the respiratory system, regional DFs predicted by static and elastic models are compared (see Fig. 3.12). Explicitly, for particles with 0.1 μm $\leq d_p \leq$5 μm (see Figs. 3.12 (a)-(f)), regardless of the lung conditions (i.e., normal, mild or severe), static TWL model predicts higher regional DF in the TB tree (from MT to G7) while lower regional DF in lower airways (G8 to acinus) than the elastic model. For particles with d_p =10 μm (see Fig. 3.12 (g)), although the same results can be noticed in MT and glottis, the regional DF in UTB and lower airways predicted by the static model is much lower compared with the elastic model. In addition, the static model predicts no deposition of large particles (d_p=10 μm) after G8, while the elastic model shows that the DF of that particles is about 18.6% for a normal lung condition. In conclusion, huge differences exist in regional DF predicted by static and elastic TWL models, especially for the DF in the lower airway.

3.4.5 Effect of Disease Condition on Regional Deposition Fraction

To enhance the delivery dosage of the drugs to the designated lung sites and the treatment effectiveness, the effect of disease-specific airway deformation on regional DF is examined and given in Fig. 3.13. For all three lung conditions, the DFs of particles with 0.1$\leq d_p \leq$10 μm in MT are less than 1%. Moreover, particles with d_p =5 μm has the highest $DF_{G8\text{-}acinus}$. With the increase in particle size, the $DF_{G8\text{-}acinus}$ first decreases (until d_p=0.5 μm) and then increases. To be specific, for the normal lung condition, $DF_{G8\text{-}acinus}$ of particles with d_p=0.1 μm is 17.1%. For particle size in 0.2$\leq d_p \leq$2 μm, the $DF_{G8\text{-}acinus}$ is approximately 6%. However, $DF_{G8\text{-}acinus}$ increases dramatically to 54.6% for particles with d_p =0.5 μm. A similar trend can be observed for mild and severe COPD conditions, although for the severe COPD condition, the highest $\mathrm{DF}_{\mathrm{G8\text{-}acinus}}$ is only 30.4% (when d_p=0.5 μm). Considering that better treatment for COPD can be achieved as higher drug dosage is delivered into deep airways (after G8), both small (e.g., d_p =0.1 μm) and large particles (e.g., d_p=5 and 10 μm) are favored. Assuming that the drug particle has side effects on the human body, then drug particles in small size would be the best option.

3.5 Conclusions

The following conclusions can be drawn from the results of the elastic TWL model simulations in Chapter 3:

(1) The airway deformation has an apparent influence on the airflow structure in the respiratory system from the glottis to the trachea for the three lung conditions investigated in this work. The effect of airway deformation on airflow structure becomes more evident after the main bronchus.

(2) Elastic TWL model predicts much lower deposition concentration in MT region for large particles, e.g., d_p=10 μm (see Figs. 3.10 (a)-(f)) compared with static model.

(3) With increasing particle size from 0.1 to 10 μm, both static and elastic models predict parabolic curves for total DF. However, the regional DFs predicted by static and the elastic whole-lung models are different as higher DF (particle size in 0.1 μm $\leq d_p \leq$10 μm) in lower airways is observed in the results from the elastic truncated whole-lung model.

(4) For the particle sizes investigated in this work, d_p=5 μm gives the highest $\mathrm{DF}_{\mathrm{G8\text{-}acinus}}$ in the elastic TWL model for all three lung conditions investigated in this study.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

4.1.1 Particle Transport and Deposition for the Dry Powder Inhaler and Human Lung

(1) ***DPI flow field vs. actuation flow rate***: At relatively low Q_{in} (i.e., 30 and 39 L/min), flow detachments occur in the region near the capsule bottom, which significantly influence the transport and deposition of API and lactose particles in the DPI flow channel. At relatively high Q_{in} (i.e., 60 and 90 L/min), no flow detachment was observed, due to the higher momentum remained in the mainstream airflow after impacting the capsule. With the increase in Q_{in}, the turbulent intensity (TI) in the capsule chamber of DPI increases significantly.

(2) ***Particle deposition in DPI vs. actuation flow rate***: At 30 and 39 L/min, concentrated depositions of API and lactose particles were found in the SH DPI flow channel at the capsule surface, the capsule chamber wall near the bottom hole of the chamber, and the grid region downstream of the capsule chamber. At 60 L/min, deposition of API particles mainly concentrated at the bottom of the capsule

chamber while the concentration in grid region is reduced. For 90 L/min, fewer particles were found deposited in the DPI due to the resuspension effect.

(3) ***Particle delivery efficiency vs. actuation flow rate & particle aspect ratio (AR)***: The SH DPI delivery efficiency decreases with the increase of Q_{in} from 30 to 60 L/min, and then increases up to 95% when the Q_{in} reaches 90 L/min. The interesting non-monotonic trend between the particle delivery efficiency and the Q_{in} is due to the resuspension of the deposited API and lactose particles at 90 L/min induced by the stronger convection effect. The deposition fractions of lactose in the SH DPI flow channel decreases by approximately 20% with the increase of lactose AR from 1 to 10 at Q_{in} of 30 and 39 L/min. Furthermore, although the AR of lactose influences the API and lactose deposition distributions in the SH DPI flow channel significantly, it has little effect on the DPI delivery efficiency.

(1) ***Particle deposition in lung vs. actuation flow rate & particle aspect ratio (AR)***: Higher Q_{in} induced higher DF_{API} in the region from G3 to G13. In contrast, lactose AR has a negligible influence on DF_{API} in the region from G3 to G13.

(2) ***Recommended coordination between patients and the SH DPI***: a high Q_{in} is favored in terms of the enhancement of the DPI-airway drug delivery efficiency, since it is the dominant factor of DF_{API} in the region of G3 to G13.

4.1.2 Disease-specific Airway Deformation Effect on Inhaled Air-particle Dynamics

(1) ***Airflow characteristics vs. airway deformation kinematics:*** The airway deformation has little influence on the airflow characteristics in the mouth-to-trachea (MT) region for all three lung conditions investigated in this study, because of the dominant inertial effect of the airflow. The effect of airway deformation on secondary airflow

structure becomes more noticeable in airways downstream to the main bronchi due to the increasing viscous effect, which are influenced by the moving wall boundary conditions significantly.

(2) ***Particle deposition vs. airway deformation kinematics:*** Due to the airway expansion during inhalation, fewer particles are entrapped in the trachea in elastic lung models than static lung models. The elastic TWL model predicts lower deposition concentration in MT region for large particles (i.e., d_p =10 μm) compared with the static lung modeling result. With particle size increasing from 0.1 to 10 μm, both static and elastic lung models predicted the total deposition fractions (DFs) following the classical U-curve trend (ICRP, 1994). However, the regional DFs predicted by static and elastic lung models are different as higher $\mathrm{DF}_{\text{G8-acinus}}$ (particle size in $0.1 \leq d_p \leq 10$ μm) can be observed in the results from elastic model.

(3) ***Pulmonary air-particle flow dynamics vs. lung disease condition***: Severe COPD condition causes difficulties for particles to be delivered to the G8-acinus region than normal and mild COPD conditions. Such an observation indicates the necessity to seek other strategies to enhance the delivery efficiency of drug particles in small airways for severe COPD patients.

(4) ***Recommended particle size for the optimized delivery efficiency to small airways:*** For the particle sizes investigated in this work, d_p =5 μm is recommended as the optimal size of drug particle for all three lung conditions investigated in this study, since it gives the highest $\mathrm{DF}_{\text{G8-acinus}}$ based on elastic model results.

4.2 Future Work

Future work is listed below to further investigate the particle-laden airflow transport phenomena in DPIs and human respiratory systems using the CFD-DEM modeling framework and elastic TWL models:

- Q_{in} between 60 and 90 L/min can be studied for the DPI delivery efficiency to find the threshold Q_{in} at which the flow convection effect overcomes the API and device wall surface energy.
- Rattling capsule effect on DPI delivery efficiency/particle DF can be considered in the future study.
- The mixing process of API and lactose inside the capsule can be included to have a better understanding of the particle interaction mechanism in DPI.
- Drug particle effect on airflow in DPI can be included due to the high volume fraction of drug particles in the DPI (two-way coupled CFD-DEM modeling).
- For the study of diseased airway deformation kinematics, varying residual volume at different stages of COPD as well as other lung diseases for both adults and children can be modeled.
- To investigate the intersubject variability, more elastic TWL geometries can be constructed using subject-specific mouth/nose-to-throat geometries (Feng et al., 2018).

REFERENCES

Aghasafari, P., & Pidaparti, R. (2018). Influence of tidal-volume setting, emphysema and ARDS on human alveolar sacs mechanics. *Acta Mechanica Sinica, 34*(5), 983-993.

Benque, B., & Khinast, J. G. (2021). Estimating inter-patient variability of dispersion in dry powder inhalers using CFD-DEM simulations. *European Journal of Pharmaceutical Sciences, 156*, 105574. doi:https://doi.org/10.1016/j.ejps.2020.105574

Calvert, G., Hassanpour, A., & Ghadiri, M. (2011). Mechanistic analysis and computer simulation of the aerodynamic dispersion of loose aggregates. *Chemical Engineering Research and Design, 89*(5), 519-525. doi:https://doi.org/10.1016/j.cherd.2010.08.013

Carrillo, J.-M. Y., Raphael, E., & Dobrynin, A. V. (2010). Adhesion of Nanoparticles. *Langmuir, 26*(15), 12973-12979. doi:10.1021/la101977c

CDER. (2017). Draft Guidance on Tiotropium Bromide. https://www.accessdata.fda.gov/drugsatfda_docs/psg/Tiotropium%20bromide_inhalation%20powder_NDA%20021395_RC10-17.pdf

Chandel, A., Goyal, A. K., Ghosh, G., & Rath, G. (2019). Recent advances in aerosolised drug delivery. *Biomedicine & Pharmacotherapy, 112*, 108601. doi:https://doi.org/10.1016/j.biopha.2019.108601

Chen, X., Feng, Y., Zhong, W., & Kleinstreuer, C. (2017). Numerical investigation of the interaction, transport and deposition of multicomponent droplets in a simple mouth-throat model. *Journal of Aerosol Science, 105*, 108-127. doi:https://doi.org/10.1016/j.jaerosci.2016.12.001

Chow, A. H. L., Tong, H. H. Y., Chattopadhyay, P., & Shekunov, B. Y. (2007). Particle Engineering for Pulmonary Drug Delivery. *Pharmaceutical research, 24*(3), 411-437. doi:10.1007/s11095-006-9174-3

Clift, R., & Gauvin, W. H. (1971). Motion of entrained particles in gas streams. *The Canadian Journal of Chemical Engineering, 49*(4), 439-448. doi:https://doi.org/10.1002/cjce.5450490403

Clift, R., Grace, J. R., & Weber, M. E. (2005). *Bubbles, drops, and particles*: Courier Corporation.

Clift, R., Grace, J. R., & Weber, M. E. (2005). Bubbles, drops, and particles.

Comerford, A., Forster, C., & Wall, W. A. (2010). Structured Tree Impedence Outflow Boundary Conditions for 3D Lung Simulations. *Journal of Biomechanical Engineering, 132*, 10.

Conway, J., Fleming, J., Bennett, M., & Havelock, T. (2013). The Co-imaging of Gamma Camera Measurements of Aerosol Deposition and Respiratory Anatomy. *Journal* *of aerosol medicine and pulmonary drug delivery, 26*(3), 123-130. doi:10.1089/jamp.2011.0960

Cui, X., Wu, W., & Ge, H. (2020). Investigation of airflow field in the upper airway under unsteady respiration pattern using large eddy simulation method. *Respiratory Physiology & Neurobiology*, 103468.

De Boer, A., Hagedoorn, P., Hoppentocht, M., Buttini, F., Grasmeijer, F., & Frijlink, H. (2017). Dry powder inhalation: past, present and future. *Expert opinion on drug delivery, 14*(4), 499-512.

Delvadia, R. R., Wei, X., Longest, P. W., Venitz, J., & Byron, P. R. (2016). In vitro tests for aerosol deposition. IV: Simulating variations in human breath profiles for realistic DPI testing. *Journal of aerosol medicine and pulmonary drug delivery, 29*(2), 196-206.

Donovan, M. J., Kim, S. H., Raman, V., & Smyth, H. D. (2012). Dry Powder Inhaler Device Influence on Carrier Particle Performance. *Journal of Pharmaceutical Sciences, 101*(3), 1097-1107. doi:https://doi.org/10.1002/jps.22824

Drew, D. A. (1983). Mathematical Modeling of Two-Phase Flow. *Annual Review of Fluid Mechanics, 15*(1), 261-291. doi:10.1146/annurev.fl.15.010183.001401

Dutta, A. (2016). *Numerical simulation of the air flow and particulate deposition in emphysematous human acini.* University of British Columbia.

Feher, J. (2017). Lung Volumes and Airway Resistance. In J. Feher (Ed.), *Quantitative Human Physiology (Second Edition)* (pp. 633-641). Boston: Academic Press.

Feng, Y. (2013). *Non-spherical particle dynamics analysis with applications to inhaled aerosol transport and deposition in human upper airway models.* (Ph.D.), North Carolina State University, Raleigh, NC, USA.

Feng, Y., & Kleinstreuer, C. (2013). *Analysis of non-spherical particle transport in complex internal shear flows* (Vol. 25).

Feng, Y., & Kleinstreuer, C. (2013). Analysis of non-spherical particle transport in complex internal shear flows. *Physics of Fluids, 25*(9), 091904.

Feng, Y., & Kleinstreuer, C. (2013). *DDPM-DEM Simulations of Particulate Flows in Human Tracheobronchial Airways.* Paper presented at the ASME 2013 International Mechanical Engineering Congress and Exposition.

Feng, Y., & Kleinstreuer, C. (2014). Micron-particle transport, interactions and deposition in triple lung-airway bifurcations using a novel modeling approach. *Journal of Aerosol Science, 71*, 1-15.

Feng, Y., Kleinstreuer, C., Castro, N., & Rostami, A. (2016). Computational transport, phase change and deposition analysis of inhaled multicomponent droplet--vapor mixtures in an idealized human upper lung model. *Journal of Aerosol Science, 96*, 96-123.

Feng, Y., Kleinstreuer, C., Castro, N., & Rostami, A. (2016). Computational transport, phase change and deposition analysis of inhaled multicomponent droplet--vapor mixtures in an idealized human upper lung model. *Journal of Aerosol Science, 96 %6*, 96-123 %&.

Feng, Y., Kleinstreuer, C., Castro, N., & Rostami, A. (2016). Computational transport, phase change and deposition analysis of inhaled multicomponent droplet–vapor mixtures in an idealized human upper lung model. *Journal of Aerosol Science, 96*, 96-123. doi:https://doi.org/10.1016/j.jaerosci.2016.03.001

Feng, Y., Kleinstreuer, C., & Rostami. (2015). Evaporation and condensation of multicomponent electronic cigarette droplets and conventional cigarette smoke particles in an idealized G3–G6 triple bifurcating unit. *Journal of Aerosol Science, 80*, 58-74. doi:10.1016/j.jaerosci.2014.11.002

Feng, Y., Kleinstreuer, C., & Rostami, A. (2015). Evaporation and condensation of multicomponent electronic cigarette droplets and conventional cigarette smoke particles in an idealized G3–G6 triple bifurcating unit. *Journal of Aerosol Science, 80*, 58-74.

Feng, Y., Marchal, T., Sperry, T., & Yi, H. (2020). Influence of wind and relative humidity on the social distancing effectiveness to prevent COVID-19 airborne transmission: A numerical study. *Journal of Aerosol Science*, 105585.

Feng, Y., Xu, Z., & Haghnegahdar, A. (2016). Computational Fluid-Particle Dynamics Modeling for Unconventional Inhaled Aerosols in Human Respiratory Systems *Aerosols-Science and Case Studies*: InTech.

Feng, Y., Zhao, J., Chen, X., & Lin, J. (2017). An in silico subject-variability study of upper airway morphological influence on the airflow regime in a tracheobronchial tree. *Bioengineering, 4*(4), 90.

Feng, Y., Zhao, J., Hayati, H., Sperry, T., & Yi, H. Tutorial: Understanding the transport, deposition, and translocation of particles in human respiratory systems using Computational Fluid-Particle Dynamics and Physiologically Based Toxicokinetic models. *Journal of Aerosol Science, 151*, 105672.

Feng, Y., Zhao, J., Kleinstreuer, C., Wang, Q., Wang, J., Wu, D. H., & Lin, J. (2018). An in silico Inter-subject Variability Study of Extra-thoracic Morphology Effects on Inhaled Particle Transport and Deposition. *Journal of Aerosol Science*.

Feng, Y., Zhao, J., Kleinstreuer, C., Wang, Q., Wang, J., Wu, D. H., & Lin, J. (2018). An in silico inter-subject variability study of extra-thoracic morphology effects on inhaled particle transport and deposition. *Journal of Aerosol Science, 123*, 185-207. doi:https://doi.org/10.1016/j.jaerosci.2018.05.010

Ganser, G. H. (1993). A rational approach to drag prediction of spherical and nonspherical particles. *Powder Technology, 77*(2), 143-152. doi:https://doi.org/10.1016/0032-5910(93)80051-B

GBD. (2017). Global, regional, and national deaths, prevalence, disability-adjusted life years, and years lived with disability for chronic obstructive pulmonary disease and asthma, 1990–2015: a systematic analysis for the Global Burden of Disease Study 2015. *The Lancet Respiratory Medicine, 5*(9), 691-706. doi:https://doi.org/10.1016/S2213-2600(17)30293-X

Generic Drug User Fee Amendments (GDUFA). (2017). Retrieved from https://www.fda.gov/industry/fda-user-fee-programs/generic-drug-user-fee-amendments#:~:text=GDUFA%20is%20based%20on%20an,GDUFA%20through%20September%2030%2C%202022.

Haghnegahdar, A., Feng, Y., Chen, X., & Lin, J. (2018). Computational Analysis of Deposition and Translocation of Inhaled Nicotine and Acrolein in the Human Body with E-cigarette Puffing Topographies. *Aerosol Science and Technology, 52*(5), 483-493. doi:10.1080/02786826.2018.1447644

Haghnegahdar, A., Zhao, J., & Feng, Y. (2019). Lung aerosol dynamics of airborne influenza A virus-laden droplets and the resultant immune system responses: An in silico study. *Journal of Aerosol Science, 134*, 21.

Haghnegahdar, A., Zhao, J., Kozak, M., Williamson, P., & Feng, Y. (2019). Development of a hybrid CFD-PBPK model to predict the transport of xenon gas around a human respiratory system to systemic regions. *Heliyon, 5*(4), e01461-e01461. doi:10.1016/j.heliyon.2019.e01461

Haider, A., & Levenspiel, O. (1989). Drag coefficient and terminal velocity of spherical and nonspherical particles. *Powder Technology, 58*(1), 63-70. doi:https://doi.org/10.1016/0032-5910(89)80008-7

Hall, J. E. (2015). Pulmonary Ventilation *Guyton and Hall Textbook of Medical Physiology* (13th edition ed.). London: Elsevier Health Sciences.

Hasler, D., Anagnostopoulou, P., Nyilas, S., Latzin, P., Schittny, J., & Obrist, D. (2019). A multi-scale model of gas transport in the lung to study heterogeneous lung ventilation during the multiple-breath washout test. *PLoS computational biology, 15*(6), e1007079.

Hayati, H., Feng, Y., & Hinsdale, M. (2021). Inter-species variabilities of droplet transport, size change, and deposition in human and rat respiratory systems: An in silico study. *Journal of Aerosol Science, 154*, 105761. doi:https://doi.org/10.1016/j.jaerosci.2021.105761

Heravi, F., Nazari, M., Chouly, F., Perrier, P., & Payan, Y. (2016). Computational fluid dynamics in the upper airway: comparison between different models and experimental data for a simplified geometry with major obstruction.

Hertz, H. (1882). Ueber die Berührung fester elastischer Körper. *Journal für die reine und angewandte Mathematik, 92*, 156-171.

Hofemeier, P., & Sznitman, J. (2016). The role of anisotropic expansion for pulmonary acinar aerosol deposition. *Journal of biomechanics, 49*(14), 3543-3548.

Horabik, J., & Molenda, M. (2016). Parameters and contact models for DEM simulations of agricultural granular materials: A review. *Biosystems Engineering, 147*, 206-225. doi:https://doi.org/10.1016/j.biosystemseng.2016.02.017

ICRP. (1994). Human Respiratory Tract Model for Radiological Protection. *Annals of the ICRP, Publication 66, 24*(1-3).

Islam, N., & Cleary, M. J. (2012). Developing an efficient and reliable dry powder inhaler for pulmonary drug delivery – A review for multidisciplinary researchers. *Medical Engineering & Physics, 34*(4), 409-427. doi:https://doi.org/10.1016/j.medengphy.2011.12.025

Jiang, M., Dai, Y., Cui, L., & Xi, B. (2018). Experimental and DEM analyses on wheel-soil interaction. *Journal of Terramechanics, 76*, 15-28. doi:https://doi.org/10.1016/j.jterra.2017.12.001

Johnson, K. L., Kendall, K., Roberts, A. D., & Tabor, D. (1971). Surface energy and the contact of elastic solids. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 324*(1558), 301-313. doi:10.1098/rspa.1971.0141

Johnson, R. W. (2016). *Handbook of fluid dynamics*: Crc Press.

Kinnunen, H., Hebbink, G., Peters, H., Shur, J., & Price, R. (2014). An Investigation into the Effect of Fine Lactose Particles on the Fluidization Behaviour and Aerosolization Performance of Carrier-Based Dry Powder Inhaler Formulations. *AAPS PharmSciTech, 15*(4), 898-909. doi:10.1208/s12249-014-0119-6

Kleinstreuer, C., & Feng, Y. (2013). Computational Analysis of Non-Spherical Particle Transport and Deposition in Shear Flow With Application to Lung Aerosol Dynamics—A Review. *Journal of Biomechanical Engineering, 135*(2). doi:10.1115/1.4023236

Kolanjiyil, A. V., & Kleinstreuer, C. (2016). Computationally efficient analysis of particle transport and deposition in a human whole-lung-airway model. Part I: Theory and model validation. *Computers in biology and medicine, 79*, 193-204.

Kolanjiyil, A. V., & Kleinstreuer, C. (2017). Computational analysis of aerosol-dynamics in a human whole-lung airway model. *Journal of Aerosol Science, 114*, 301-316.

Kolanjiyil, A. V., & Kleinstreuer, C. (2019). Modeling Airflow and Particle Deposition in a Human Acinar Region. *Computational and mathematical methods in medicine, 2019*, 5952941-5952941. doi:10.1155/2019/5952941

Koshiyama, K., & Wada, S. (2015). Mathematical model of a heterogeneous pulmonary acinus structure. *Computers in Biology and Medicine, 62*, 25-32. doi:https://doi.org/10.1016/j.compbiomed.2015.03.032

Koullapis, P., Kassinos, S. C., Muela, J., Perez-Segarra, C., Rigola, J., Lehmkuhl, O., . . . Nicolaou, L. (2018). Regional aerosol deposition in the human airways: The SimInhale benchmark case and a critical assessment of in silico methods. *European Journal of Pharmaceutical Sciences, 113*, 77-94. doi:https://doi.org/10.1016/j.ejps.2017.09.003

Koullapis, P. G., Kassinos, S. C., Bivolarova, M. P., & Melikov, A. K. (2016). Particle deposition in a realistic geometry of the human conducting airways: Effects of inlet velocity profile, inhalation flowrate and electrostatic charge. *Journal of Biomechanics, 49*(11), 2201-2212. doi:https://doi.org/10.1016/j.jbiomech.2015.11.029

Koullapis, P. G., Nicolaou, L., & Kassinos, S. C. (2018). In silico assessment of mouth-throat effects on regional deposition in the upper tracheobronchial airways. *Journal of Aerosol Science, 117*, 164-188. doi:https://doi.org/10.1016/j.jaerosci.2017.12.001

Koullapis, P. G., Stylianou, F. S., Sznitman, J., Olsson, B., & Kassinos, S. C. (2020). Towards whole-lung simulations of aerosol deposition: A model of the deep lung. *Journal of Aerosol Science, 144*, 105541. doi:https://doi.org/10.1016/j.jaerosci.2020.105541

Labiris, N. R., & Dolovich, M. B. (2003). Pulmonary drug delivery. Part I: physiological factors affecting therapeutic effectiveness of aerosolized medications. *British journal of clinical pharmacology, 56*(6), 588-599. doi:10.1046/j.1365-2125.2003.01892.x

Lee, S. L., Saluja, B., García-Arieta, A., Santos, G. M. L., Li, Y., Lu, S., . . . Lyapustina, S. (2015). Regulatory Considerations for Approval of Generic Inhalation Drug

Products in the US, EU, Brazil, China, and India. *The AAPS journal, 17*(5), 1285-1304. doi:10.1208/s12248-015-9787-8

Lewis, D., Rouse, T., Singh, D., & Edge, S. (2017). Defining the 'Dose' for Dry Powder Inhalers: The Challenge of Correlating In-Vitro Dose Delivery Results with Clinical Efficacy. *American Pharmaceutical Review, 20*, 54-62.

Longest, P. W., Bass, K., Dutta, R., Rani, V., Thomas, M. L., El-Achwah, A., & Hindle, M. (2019). Use of computational fluid dynamics deposition modeling in respiratory drug delivery. *Expert Opinion on Drug Delivery, 16*(1), 7-26. doi:10.1080/17425247.2019.1551875

Longest, P. W., Tian, G., Khajeh-Hosseini-Dalasm, N., & Hindle, M. (2016). Validating Whole-Airway CFD Predictions of DPI Aerosol Deposition at Multiple Flow Rates. *Journal of aerosol medicine and pulmonary drug delivery, 29*(6), 461-481. doi:10.1089/jamp.2015.1281

Longest, P. W., & Xi, J. (2007). Effectiveness of Direct Lagrangian Tracking Models for Simulating Nanoparticle Deposition in the Upper Airways. *Aerosol Science and Technology, 41*(4), 380-397. doi:10.1080/02786820701203223

Malvè, M., Pérez del Palomar, A., Chandra, S., López-Villalobos, J., Mena, A., Finol, E., . . . Doblaré, M. (2011). FSI analysis of a healthy and a stenotic human trachea under impedance-based boundary conditions. *Journal of Biomechanical Engineering, 133*(2).

Marheineke, N., & Wegener, R. (2011). Modeling and application of a stochastic drag for fibers in turbulent flows. *International Journal of Multiphase Flow, 37*(2), 136-148. doi:https://doi.org/10.1016/j.ijmultiphaseflow.2010.10.001

Martin, A. R., Moore, C. P., & Finlay, W. H. (2018). Models of deposition, pharmacokinetics, and intersubject variability in respiratory drug delivery. *Expert Opinion on Drug Delivery, 15*(12), 1175-1188. doi:10.1080/17425247.2018.1544616

McLoud, T. C., & Aquino, S. L. (2010). Chapter 1 - Thoracic Radiology: Imaging Methods, Radiographic Signs, and Diagnosis of Chest Disease. In T. C. McLoud & P. M. Boiselle (Eds.), *Thoracic Radiology (Second Edition)* (pp. 1-58). Philadelphia: Mosby.

Mead, J., Turner, J. M., & Little, J. B. (1967). Significance of the relationship between lung recoil and maximum expiratory flow. *Journal of Applied Physiology, 22*(1), 95-108. doi:doi: 10.1152/jappl.1967.22.1.95

Mindlin, R. D. J. J. A. M. (1953). Elastic spheres in contact under varying oblique forces. *20*, 327-344.

Mitani, R., Ohsaki, S., Nakamura, H., & Watano, S. (2020). Numerical Study on Particle Adhesion in Dry Powder Inhaler Device. *Chemical and Pharmaceutical Bulletin, 68*(8), 726-736. doi:10.1248/cpb.c20-00106

Modenese, C., Utili, S., & Houlsby, G. T. (2012). A study of the influence of surface energy on the mechanical properties of lunar soil using DEM *Discrete Element Modelling of Particulate Media* (pp. 69-75).

Moreno-Atanasio, R. (2012). Energy dissipation in agglomerates during normal impact. *Powder Technology, 223*, 12-18. doi:https://doi.org/10.1016/j.powtec.2011.05.016

Morrissey, J. P. (2013). *Discrete element modelling of iron ore pellets to include the effects of moisture and fines (Doctoral dissertation).* University of Edinburgh, UK.

Newman, S., Bennett, W. D., Biddiscombe, M., Devadason, S. G., Dolovich, M. B., Fleming, J., . . . Zeman, K. L. (2012). Standardization of Techniques for Using Planar (2D) Imaging for Aerosol Deposition Assessment of Orally Inhaled Products. *Journal of aerosol medicine and pulmonary drug delivery, 25*(S1), S-10-S-28. doi:10.1089/jamp.2012.1Su4

Plathow, C., Ley, S., Fink, C., Puderbach, M., Heilmann, M., Zuna, I., & Kauczor, H.-U. (2004). Evaluation of Chest Motion and Volumetry During the Breathing Cycle by Dynamic MRI in Healthy Subjects. *Investigative radiology, 39*, 202-209. doi:10.1097/01.rli.0000113795.93565.c3

Ponzini, R., Da Vià, R., Bnà, S., Cottini, C., & Benassi, A. (2021). Coupled CFD-DEM model for dry powder inhalers simulation: Validation and sensitivity analysis for the main model parameters. *Powder Technology, 385*, 199-226. doi:https://doi.org/10.1016/j.powtec.2021.02.044

Poorbahrami, K., Vignon-Clementel, I. E., Shadden, S. C., & Oakes, J. M. (2021). A whole lung in silico model to estimate age dependent particle dosimetry. *Scientific Reports, 11*(1), 11180. doi:10.1038/s41598-021-90509-8

Pritchard, P. J., & Mitchell, J. W. (2011). *Fox and McDonald's Introduction to Fluid Mechanics* (9th ed.): John Wiley & Sons.

. Respiratory Physiology. (2019). In C. Huang, D. Chambers, & G. Matthews (Eds.), *Basic Physiology for Anaesthetists* (2 ed., pp. 50-55). Cambridge: Cambridge University Press.

Seyfi, B., Santhanam, A. P., & Ilegbusi, O. J. (2016). A biomechanical model of human lung deformation utilizing patient-specific elastic property. *Journal of Cancer Therapy, 7*(6), 402-415.

Seyfi Noferest, B., Santhanam, A. P., & Ilegbusi, O. J. (2018). Effect of gravity on subject-specific human lung deformation. *Mathematical and Computer Modelling of Dynamical Systems, 24*(1), 87-101.

Stahlhofen, W., Rudolf, G., & James, A. C. (1989). Intercomparison of Experimental Regional Aerosol Deposition Data. *Journal of Aerosol Medicine, 2*(3), 285-308. doi:10.1089/jam.1989.2.285

Subramaniam, D. R., Mylavarapu, G., Fleck, R. J., Amin, R. S., Shott, S. R., & Gutmark, E. J. (2017). Effect of airflow and material models on tissue displacement for surgical planning of pharyngeal airways in pediatric down syndrome patients. *Journal of the mechanical behavior of biomedical materials, 71*, 122-135.

Sul, B., Altes, T., Ruppert, K., Qing, K., Hariprasad, D. S., Morris, M., . . . Wallqvist, A. (2019). Dynamics of the Tracheal Airway and Its Influences on Respiratory Airflows: An Exemplar Study. *Journal of Biomechanical Engineering, 141*(11).

Talaat, K., & Xi, J. (2017). Computational modeling of aerosol transport, dispersion, and deposition in rhythmically expanding and contracting terminal alveoli. *Journal of Aerosol Science, 112*, 19-33.

Taulbee, D. B., & P., Y. C. (1975). A theory of aerosol deposition in the humen respirtory tract. *Journal of Applied Physiology, 38*(1), 77-85.

Tawhai, M. H., Pullan, A. J., & Hunter, P. J. (2000). Generation of an Anatomically Based Three-Dimensional Model of the Conducting Airways. *Annals of biomedical engineering, 28*, 793-802.

Tena, A. F., Fernández, J., Álvarez, E., Casan, P., & Walters, D. K. (2017). Design of a numerical model of lung by means of a special boundary condition in the truncated branches. *International Journal for Numerical Methods in Biomedical Engineering, 33*(6), e2830. doi:10.1002/cnm.2830

Tian, G., Hindle, M., Lee, S., & Longest, P. W. (2015). Validating CFD predictions of pharmaceutical aerosol deposition with in vivo data. *Pharmaceutical research, 32*(10), 3170-3187. doi:10.1007/s11095-015-1695-1

Tian, L., Inthavong, K., Lidén, G., Shang, Y., & Tu, J. (2016). Transport and deposition of welding fume agglomerates in a realistic human nasal airway. *Annals Of Occupational Hygiene, 60*(6), 731-747.

Tong, Yang, & Yu. (2017). CFD-DEM study of the aerosolisation mechanism of carrier-based formulations with high drug loadings. *Powder Technology, 314*, 620-626. doi:https://doi.org/10.1016/j.powtec.2016.10.004

Tong, Zhong, Yu, Chan, & Yang. (2016). CFD–DEM investigation of the effect of agglomerate–agglomerate collision on dry powder aerosolisation. *Journal of Aerosol Science, 92*, 109-121. doi:https://doi.org/10.1016/j.jaerosci.2015.11.005

Walenga, R. L., Babiskin, A. H., & Zhao, L. (2019). In Silico Methods for Development of Generic Drug-Device Combination Orally Inhaled Drug Products. *CPT: pharmacometrics & systems pharmacology, 8*(6), 359-370. doi:10.1002/psp4.12413

Walenga, R. L., Tian, G., & Worth Longest, P. (2013). Development of Characteristic Upper Tracheobronchial Airway Models for Testing Pharmaceutical Aerosol Delivery. *Journal of Biomechanical Engineering, 135*(9). doi:10.1115/1.4024630

Wall, W. A., & Rabczuk, T. (2008). Fluid–structure interaction in lower airways of CT-based lung geometries. *International Journal for Numerical Methods in Fluids, 57*(5), 653-675.

Walton, O. R. (2008). Review of Adhesion Fundamentals for Micron-Scale Particles. *KONA Powder and Particle Journal, 26*, 129-141. doi:10.14356/kona.2008012

Wang, C. S. (2005). Inhaled Particles. *Interface Sci Tech, 5*, 1-187. doi:10.1016/S1573-4285(05)80005-5

Wang, J., Xi, J., Han, P., Wongwiset, N., Pontius, J., & Dong, H. (2019). Computational analysis of a flapping uvula on aerodynamics and pharyngeal wall collapsibility in sleep apnea. *Journal of biomechanics, 94*, 88-98.

Wang, Y., Chu, K., & Yu, A. (2017). *Transport and deposition of cohesive pharmaceutical powders in human airway.* Paper presented at the EPJ Web of Conferences

Werner, R., Ehrhardt, J., Schmidt, R., & Handels, H. (2008). *Modeling respiratory lung motion: A biophysical approach using finite element methods.* Paper presented at the Medical Imaging 2008: Physiology, Function, and Structure from Medical Images.

Xi, J., & Longest, P. W. (2007). Transport and Deposition of Micro-Aerosols in Realistic and Simplified Models of the Oral Airway. *Annals of Biomedical Engineering, 35*(4), 560-581. doi:10.1007/s10439-006-9245-y

Xi, J., Talaat, K., & Si, X. A. (2018). Deposition of bolus and continuously inhaled aerosols in rhythmically moving terminal alveoli. *The Journal of Computational Multiphase Flows, 10*(4), 178-193.

Xi, J., Talaat, M., Tanbour, H., & Talaat, K. (2018). Airflow and Particle Deposition in Acinar Models with Interalveolar Septal Walls and Different Alveolar Numbers. *Computational and mathematical methods in medicine, 2018*, 3649391-3649391. doi:10.1155/2018/3649391

Xi, J., Wang, Z., Talaat, K., Glide-Hurst, C., & Dong, H. (2018). Numerical study of dynamic glottis and tidal breathing on respiratory sounds in a human upper airway model. *Sleep and Breathing, 22*(2), 463-479.

Xi, J., Yuan, J. E., Yang, M., Si, X., Zhou, Y., & Cheng, Y.-S. (2016). Parametric study on mouth–throat geometrical factors on deposition of orally inhaled aerosols. *Journal of Aerosol Science, 99*, 94-106. doi:https://doi.org/10.1016/j.jaerosci.2016.01.014

Xia, G., Tawhai, M. H., Hoffman, E. A., & Lin, C.-L. (2010). Airway wall stiffening increases peak wall shear stress: a fluid–structure interaction study in rigid and compliant airways. *Annals of biomedical engineering, 38*(5), 1836-1853.

Yang, J., Wu, C.-Y., & Adams, M. (2015). Numerical Modelling of Agglomeration and Deagglomeration in Dry Powder Inhalers: A Review. *Current Pharmaceutical Design, 21*(40), 5915-5922. doi:http://dx.doi.org/10.2174/1381612821666151008150845

Yin, Y., Choi, J., Hoffman, E. A., Tawhai, M. H., & Lin, C.-L. (2013). A multiscale MDCT image-based breathing lung model with time-varying regional ventilation. *Journal of Computational Physics, 244*, 168-192. doi:https://doi.org/10.1016/j.jcp.2012.12.007

Zellnitz, S., Zellnitz, L., Müller, M. T., Meindl, C., Schröttner, H., & Fröhlich, E. (2019). Impact of drug particle shape on permeability and cellular uptake in the lung. *European Journal of Pharmaceutical Sciences, 139*, 105065. doi:https://doi.org/10.1016/j.ejps.2019.105065

Zhang, Z., Kleinstreuer, C., & Feng, Y. (2012). Vapor deposition during cigarette smoke inhalation in a subject-specific human airway model. *Journal of Aerosol Science, 53*, 40-60. doi:https://doi.org/10.1016/j.jaerosci.2012.05.008

Zhang, Z., Kleinstreuer, C., & Hyun, S. (2012). Size-change and deposition of conventional and composite cigarette smoke particles during inhalation in a subject-specific airway model. *Journal of Aerosol Science, 46*, 34-52. doi:https://doi.org/10.1016/j.jaerosci.2011.12.002

Zhang, Z., Kleinstreuer, C., & Kim, C. S. (2008). Airflow and Nanoparticle Deposition in a 16-Generation Tracheobronchial Airway Model. *Annals of Biomedical Engineering,, 36*(19), 2095-2110.

Zhao, J., Feng, Y., Bezerra, M., Wang, J., & Sperry, T. (2019). Numerical simulation of welding fume lung dosimetry. *Journal of Aerosol Science*. doi:https://doi.org/10.1016/j.jaerosci.2019.05.006

Zhao, J., Feng, Y., & Fromen, C. A. (2020). Glottis motion effects on the particle transport and deposition in a subject-specific mouth-to-trachea model: A CFPD study. *Computers in Biology and Medicine, 116*, 103532. doi:https://doi.org/10.1016/j.compbiomed.2019.103532

Zhao, J. F., Y.; Hjaghnegahdar, A.; Sarkar, S.; Bharadwaj, R. (2020). *Numerical Investigation of Particle Shape and Actuation Flow Rate Effects on Lactose Carrier Delivery Efficiency through a Dry Powder Inhaler (DPI) Using CFD-DEM*. Paper presented at the 2020 Virtual AIChE Annual Meeting https://www.aiche.org/academy/conferences/aiche-annual-meeting/2020/proceeding/paper/316e-numerical-investigation-particle-shape-and-actuation-flow-rate-effects-on-lactose-carrier

Zhao, J. F., Y; Fromen, C. (2020). Glottis Motion Effects on the Inhaled Particle Transport and Deposition in a Subject-Specific Mouth-to-Trachea Model: An CFPD Study. *Computers in Biology and Medicine, 116*.

APPENDICE

APPENDIX A

In-house Code for Coupled Pressure-outlet Boundary Condition

```
/******************************************************************************/
/*                          Copyright (C) 2021, CBBL                          */
/*                     School of Chemical Engineering                         */
/*                   Oklahoma State University, Stillwater, OK, USA           */
/******************************************************************************/

#include"udf.h"

#define NUM_C 75

int interfaceID[NUM_C] = { 676, 674, 672, 670, 668, 663, 661, 659, 657,
655, 650, 648, 646,
644, 678, 499, 497, 495, 493, 457, 455, 453, 451, 449, 445, 443, 441,
439, 437, 501, 42, 40,
38, 35, 32, 30, 28, 26, 24, 20, 18, 15, 13, 11, 45, 904, 902, 900, 898,
896, 891, 889, 887, 885,
883, 878, 876, 874, 872, 906, 273, 271, 269, 267, 262, 260, 258, 256,
254, 252, 214, 212, 210,
207, 275 };

int outletID[NUM_C] = { 717, 715, 713, 711, 706, 704, 700, 698, 696,
693, 691, 689, 687, 685,
683, 533, 531, 529, 525, 523, 521, 519, 517, 515, 513, 511, 509, 507,
505, 503, 84, 81, 79, 75,
73, 71, 69, 67, 65, 60, 58, 56, 54, 52, 47, 944, 942, 940, 938, 933,
930, 928, 925, 923, 921, 918,
915, 912, 910, 908, 314, 312, 310, 308, 303, 301, 299, 297, 295, 290,
288, 286, 284, 282, 280};

real pressure[NUM_C];

DEFINE_ADJUST(InterfacesPressure, d)
{
        int i;

        for (i = 0; i < NUM_C; i++)
        {
                real integratedpres[NUM_C];
                real pres = 0.0;
                real pres_outlet = 0.0;
                real area = 0.0;
                real area_outlet = 0.0;
                real total_area[NUM_C];

#if !RP_HOST
                cell_t c;
                Thread *t;
                Thread *t_outlet;
                face_t f;
                real A[ND_ND];
                real A_outlet[ND_ND];
                t = Lookup_Thread(d, interfaceID[i]);
                t_outlet = Lookup_Thread(d, outletID[i]);

                begin_f_loop(f, t)
                {
                        if (PRINCIPAL_FACE_P(f, t))
                        {
                                F_AREA(A, f, t);
                                area += NV_MAG(A);
                                pres += F_P(f, t)*NV_MAG(A);
                        }
                }
                end_f_loop(f, t);

                begin_f_loop(f, t_outlet)
                {
                        if (PRINCIPAL_FACE_P(f, t_outlet))
                        {
                                F_AREA(A_outlet, f, t_outlet);
                                area_outlet += NV_MAG(A_outlet);
                                pres_outlet += F_P(f,
t_outlet)*NV_MAG(A_outlet);
                        }
                }
                end_f_loop(f, t_outlet);

                total_area[i] = area;
                integratedpres[i] = pres;
#endif
#if RP_NODE
                integratedpres[i] = PRF_GRSUM1(integratedpres[i]);
                total_area[i] = PRF_GRSUM1(total_area[i]);
                pres_outlet = PRF_GRSUM1(pres_outlet);
                area_outlet = PRF_GRSUM1(area_outlet);
#endif


                node_to_host_real_4(integratedpres[i], total_area[i],
pres_outlet, area_outlet);

#if !RP_NODE
                pressure[i] = integratedpres[i] / total_area[i];

#endif

#if RP_NODE
                pressure[i] = integratedpres[i] / total_area[i];
#endif

        }

}

 DEFINE_PROFILE(CoupledPressure_BC, t, p)
 {
         face_t f;
         int zoneID = THREAD_ID(t);
         int i = 0;

         while (zoneID != outletID[i])
         {
                 i++;
         }
         begin_f_loop(f, t)
         {
                 F_PROFILE(f, t, p) = pressure[i];
         }
         end_f_loop(f, t);
 }
```

APPENDIX B

In-house Code for TWL Model Deformation Kinematics

```
/********************************************************************************/
/*                          Copyright (C) 2021, CBBL                            */
/*                     School of Chemical Engineering                           */
/*                   Oklahoma State University, Stillwater, OK, USA             */
/********************************************************************************/

#include "udf.h"
#include "math.h"

#define x1 0.12
#define x0 0.18
#define y0 0.00081
#define z0 0.11738
#define dt_x 0.0918
#define dt_y 0.0344
#define dt_z 0.0918

#define dt_x2 0.4
#define dt_y2 0.4
#define dt_z2 0.4

#define Tc 2.0

#define n1_lll -0.58498647
#define n2_lll 0.00857555
#define n3_lll 0.52204073
#define n1_lul 0.81470376
#define n2_lul 0.02437883
#define n3_lul -0.47292647
#define n1_rll 0.28214610
#define n2_rll 0.03087414
#define n3_rll -0.91866648
#define n1_rml 0.58678423
#define n2_rml -0.17094797
#define n3_rml 0.49716052
#define n1_rul 0.86671369
#define n2_rul 0.33406560
#define n3_rul -0.24391174

real xr_lll, yr_lll, zr_lll;
real xr_lul, yr_lul, zr_lul;
real xr_rll, yr_rll, zr_rll;
real xr_rml, yr_rml, zr_rml;
real xr_rul, yr_rul, zr_rul;
real pt_x_lll, pt_y_lll, pt_z_lll, pb_x_lll, pb_y_lll, pb_z_lll;
real pt_x_lul, pt_y_lul, pt_z_lul, pb_x_lul, pb_y_lul, pb_z_lul;
real pt_x_rll, pt_y_rll, pt_z_rll, pb_x_rll, pb_y_rll, pb_z_rll;
real pt_x_rml, pt_y_rml, pt_z_rml, pb_x_rml, pb_y_rml, pb_z_rml;
real pt_x_rul, pt_y_rul, pt_z_rul, pb_x_rul, pb_y_rul, pb_z_rul;

DEFINE_GRID_MOTION(LobeMotion, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD(dt);
        face_t f;
        Node *node_p;
        int n;

        SET_DEFORMING_THREAD_FLAG(THREAD_T0 (tf));

        begin_f_loop(f, tf)
        {
                f_node_loop(f, tf, n)
                {
                        node_p = F_NODE(f, tf, n);

                        if (THREAD_ID(tf) == 635)
                        {
                                if (NODE_X(node_p) >= x1 &&
NODE_X(node_p) <= x0)
                                {
                                        if (NODE_POS_NEED_UPDATE(node_p))
                                        {
                                                NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) *
                                                        (1 + 0.5 * dt_x *
(1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos((NODE_X(node_p) - x0)
/ (x1 - x0) * 2 * M_PI))) /
                                                        (1 + 0.5 * dt_x *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos((NODE_X(node_p) - x0) / (x1 - x0) * 2 * M_PI))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) *
                                                        (1 + 0.5 * dt_y *
(1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos((NODE_X(node_p) - x1)
/ (x0 - x1) * M_PI))) /
                                                        (1 + 0.5 * dt_y *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos((NODE_X(node_p) - x1) / (x0 - x1) * M_PI))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) *
                                                        (1 + 0.5 * dt_z *
(1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos((NODE_X(node_p) - x1)
/ (x0 - x1) * M_PI))) /
```

```
                                                       (1 + 0.5 * dt_z *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos((NODE_X(node_p) - x1) / (x0 - x1) * M_PI))) + z0;
                                        }

                                }
                                else if (NODE_X(node_p) > x0)
                                {
                                        if (NODE_POS_NEED_UPDATE(node_p))
                                        {
                                                NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                       (1 + 0.5 * dt_x *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                       (1 + 0.5 * dt_y *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                       (1 + 0.5 * dt_z *
(1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                        }
                                }
                        }
                        else
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) = (NODE_X(node_p)
- x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * time))) /
                                               (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) = (NODE_Y(node_p)
- y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * time))) /
                                               (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) = (NODE_Z(node_p)
- z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc * time))) /
                                               (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                }
                        }
                }
        }
        end_f_loop(f, tf)
}

DEFINE_EXECUTE_AT_END(execute_at_end)
{
        if (CURRENT_TIME == 0)
        {
                xr_lll = 0.322817;
                yr_lll = -0.0357061;
                zr_lll = 0.130845;
                pt_x_lll = 0.32367;
                pt_y_lll = -0.0355509;
                pt_z_lll = 0.130295;
                pb_x_lll = 0.32299;
                pb_y_lll = -0.0354847;
                pb_z_lll = 0.130724;

                xr_lul = 0.15117395;
                yr_lul = -0.041381524;
                zr_lul = 0.1160977;
                pt_x_lul = 0.150511;
                pt_y_lul = -0.0415025;
                pt_z_lul = 0.116703;
                pb_x_lul = 0.151096;
                pb_y_lul = -0.0415155;
                pb_z_lul = 0.116336;

                xr_rll = 0.26993951;
                yr_rll = 0.037225799;
                zr_rll = 0.22607505;
                pt_x_rll = 0.269594;
                pt_y_rll = 0.0369288;
                pt_z_rll = 0.227075;
                pb_x_rll = 0.269757;
                pb_y_rll = 0.0369957;
                pb_z_rll = 0.226189;

                xr_rml = 0.22432601;
                yr_rml = 0.071808398;
                zr_rml = 0.087182655;
                pt_x_rml = 0.223893;
                pt_y_rml = 0.0718294;
                pt_z_rml = 0.0866849;
                pb_x_rml = 0.224264;
                pb_y_rml = 0.0717112;
                pb_z_rml = 0.087015;

                xr_rul = 0.14188367;
                yr_rul = 0.046769503;
                zr_rul = 0.1475353;
                pt_x_rul = 0.141248;
                pt_y_rul = 0.0464001;
                pt_z_rul = 0.147897;
                pb_x_rul = 0.14179;
                pb_y_rul = 0.0467204;
                pb_z_rul = 0.147779;

                xr_lll = (xr_lll - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
```

```
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_lll = (yr_lll - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * 
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_lll = (zr_lll - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * 
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_lll = (pt_x_lll - x0) * (1 + 0.5 * dt_x * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_lll = (pt_y_lll - y0) * (1 + 0.5 * dt_y * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_lll = (pt_z_lll - z0) * (1 + 0.5 * dt_z * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lll = (pb_x_lll - x0) * (1 + 0.5 * dt_x * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_lll = (pb_y_lll - y0) * (1 + 0.5 * dt_y * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_lll = (pb_z_lll - z0) * (1 + 0.5 * dt_z * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;

                xr_lul = (xr_lul - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * 
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_lul = (yr_lul - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * 
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_lul = (zr_lul - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * 
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_lul = (pt_x_lul - x0) * (1 + 0.5 * dt_x * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_lul = (pt_y_lul - y0) * (1 + 0.5 * dt_y * (1 - 
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc * 
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
```

```
                pt_z_lul = (pt_z_lul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lul = (pb_x_lul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_lul = (pb_y_lul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_lul = (pb_z_lul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;

                xr_rll = (xr_rll - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rll = (yr_rll - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rll = (zr_rll - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rll = (pt_x_rll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rll = (pt_y_rll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rll = (pt_z_rll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rll = (pb_x_rll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rll = (pb_y_rll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rll = (pb_z_rll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
```

```
                xr_rml = (xr_rml - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rml = (yr_rml - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rml = (zr_rml - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rml = (pt_x_rml - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rml = (pt_y_rml - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rml = (pt_z_rml - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rml = (pb_x_rml - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rml = (pb_y_rml - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rml = (pb_z_rml - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;

                xr_rul = (xr_rul - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rul = (yr_rul - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rul = (zr_rul - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rul = (pt_x_rul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rul = (pt_y_rul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
```

```
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rul = (pt_z_rul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rul = (pb_x_rul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rul = (pb_y_rul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rul = (pb_z_rul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
        }
        else
        {

                xr_lll = (xr_lll - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_lll = (yr_lll - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_lll = (zr_lll - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_lll = (pt_x_lll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_lll = (pt_y_lll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_lll = (pt_z_lll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lll = (pb_x_lll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_lll = (pb_y_lll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
```

```
                pb_z_lll = (pb_z_lll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lll = (pb_x_lll - xr_lll) * (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + xr_lll;
                pb_y_lll = (pb_y_lll - yr_lll) * (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + yr_lll;
                pb_z_lll = (pb_z_lll - zr_lll) * (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + zr_lll;

                xr_lul = (xr_lul - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_lul = (yr_lul - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_lul = (zr_lul - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_lul = (pt_x_lul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_lul = (pt_y_lul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_lul = (pt_z_lul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lul = (pb_x_lul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_lul = (pb_y_lul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_lul = (pb_z_lul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_lul = (pb_x_lul - xr_lul) * (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
```

```
                        (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + xr_lul;
                pb_y_lul = (pb_y_lul - yr_lul) * (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + yr_lul;
                pb_z_lul = (pb_z_lul - zr_lul) * (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + zr_lul;

                xr_rll = (xr_rll - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rll = (yr_rll - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rll = (zr_rll - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rll = (pt_x_rll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rll = (pt_y_rll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rll = (pt_z_rll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rll = (pb_x_rll - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rll = (pb_y_rll - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rll = (pb_z_rll - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rll = (pb_x_rll - xr_rll) * (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + xr_rll;
                pb_y_rll = (pb_y_rll - yr_rll) * (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + yr_rll;
```

```
                pb_z_rll = (pb_z_rll - zr_rll) * (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + zr_rll;

                xr_rml = (xr_rml - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rml = (yr_rml - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rml = (zr_rml - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rml = (pt_x_rml - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rml = (pt_y_rml - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rml = (pt_z_rml - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rml = (pb_x_rml - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rml = (pb_y_rml - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rml = (pb_z_rml - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rml = (pb_x_rml - xr_rml) * (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + xr_rml;
                pb_y_rml = (pb_y_rml - yr_rml) * (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + yr_rml;
                pb_z_rml = (pb_z_rml - zr_rml) * (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + zr_rml;
```

```
                xr_rul = (xr_rul - x0) * (1 + 0.5 * dt_x * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                yr_rul = (yr_rul - y0) * (1 + 0.5 * dt_y * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                zr_rul = (zr_rul - z0) * (1 + 0.5 * dt_z * (1 - cos(2 *
M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pt_x_rul = (pt_x_rul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pt_y_rul = (pt_y_rul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pt_z_rul = (pt_z_rul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rul = (pb_x_rul - x0) * (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + x0;
                pb_y_rul = (pb_y_rul - y0) * (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + y0;
                pb_z_rul = (pb_z_rul - z0) * (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + z0;
                pb_x_rul = (pb_x_rul - xr_rul) * (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + xr_rul;
                pb_y_rul = (pb_y_rul - yr_rul) * (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + yr_rul;
                pb_z_rul = (pb_z_rul - zr_rul) * (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * CURRENT_TIME))) /
                        (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
MAX(CURRENT_TIME - CURRENT_TIMESTEP, 0.0)))) + zr_rul;
        }
}


DEFINE_GRID_MOTION(AcinusMotion_lll, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD (dt);
        Thread *c_thread;
        cell_t c;
        face_t f;
        Node *node_p;
        real num = THREAD_ID(tf);
        int n;
        real D_t_lll, D_b_lll, d_m2t_lll, d_m2b_lll;
        real D_t_lul, D_b_lul, d_m2t_lul, d_m2b_lul;
        real D_t_rll, D_b_rll, d_m2t_rll, d_m2b_rll;
        real D_t_rml, D_b_rml, d_m2t_rml, d_m2b_rml;
        real D_t_rul, D_b_rul, d_m2t_rul, d_m2b_rul;

        D_t_lll = -n1_lll * pt_x_lll - n2_lll * pt_y_lll - n3_lll *
pt_z_lll;
        D_b_lll = -n1_lll * pb_x_lll - n2_lll * pb_y_lll - n3_lll *
pb_z_lll;
        real D_lll = D_t_lll - D_b_lll;
        D_t_lul = -n1_lul * pt_x_lul - n2_lul * pt_y_lul - n3_lul *
pt_z_lul;
        D_b_lul = -n1_lul * pb_x_lul - n2_lul * pb_y_lul - n3_lul *
pb_z_lul;
        real D_lul = D_t_lul - D_b_lul;
        D_t_rll = -n1_rll * pt_x_rll - n2_rll * pt_y_rll - n3_rll *
pt_z_rll;
        D_b_rll = -n1_rll * pb_x_rll - n2_rll * pb_y_rll - n3_rll *
pb_z_rll;
        real D_rll = D_t_rll - D_b_rll;
        D_t_rml = -n1_rml * pt_x_rml - n2_rml * pt_y_rml - n3_rml *
pt_z_rml;
        D_b_rml = -n1_rml * pb_x_rml - n2_rml * pb_y_rml - n3_rml *
pb_z_rml;
        real D_rml = D_t_rml - D_b_rml;
        D_t_rul = -n1_rul * pt_x_rul - n2_rul * pt_y_rul - n3_rul *
pt_z_rul;
        D_b_rul = -n1_rul * pb_x_rul - n2_rul * pb_y_rul - n3_rul *
pb_z_rul;
        real D_rul = D_t_rul - D_b_rul;

        #if !RP_NODE
          Message("\nacinus motion     time = %e              ",
CURRENT_TIME);
          Message("pt_x_lll = %e, pt_y_lll = %e, pt_z_lll = %e, D_t_lll
= %e\n", pt_x_lll, pt_y_lll, pt_z_lll, D_t_lll);
          Message("pb_x_lll = %e, pb_y_lll = %e, pb_z_lll = %e, D_b_lll
= %e\n", pb_x_lll, pb_y_lll, pb_z_lll, D_b_lll);
        #endif

          SET_DEFORMING_THREAD_FLAG (THREAD_T0 (tf));

          begin_f_loop(f, tf)
          {
                  f_node_loop(f, tf, n)
                  {
                          node_p = F_NODE(f, tf, n);
```

```
                         if (THREAD_ID(tf) == 642)
                         {
                                 if ((n1_lll*NODE_X(node_p) +
n2_lll*NODE_Y(node_p) + n3_lll*NODE_Z(node_p) + D_t_lll < 0))
                                 {
                                        if
(NODE_POS_NEED_UPDATE(node_p))
                                        {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                        }
                                 }
                                 else if ((n1_lll*NODE_X(node_p) +
n2_lll*NODE_Y(node_p) + n3_lll*NODE_Z(node_p) + D_t_lll >= 0) &&
(n1_lll*NODE_X(node_p) + n2_lll*NODE_Y(node_p) + n3_lll*NODE_Z(node_p)
+ D_b_lll <= 0))
                                 {
                                        if
(NODE_POS_NEED_UPDATE(node_p))
                                        {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                                d_m2t_lll =
fabs(n1_lll*NODE_X(node_p) + n2_lll*NODE_Y(node_p) +
n3_lll*NODE_Z(node_p) + D_t_lll) / sqrt(n1_lll*n1_lll + n2_lll*n2_lll +
n3_lll*n3_lll);

                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_lll) *
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lll / D_lll *
M_PI))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lll / D_lll * M_PI))) + xr_lll;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lll) *
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lll / D_lll *
M_PI))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lll / D_lll * M_PI))) + yr_lll;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lll) *
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lll / D_lll *
M_PI))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lll / D_lll * M_PI))) + zr_lll;
                                        }
                                }
                                else if ((n1_lll*NODE_X(node_p) +
n2_lll*NODE_Y(node_p) + n3_lll*NODE_Z(node_p) + D_b_lll > 0))
                                {
                                        if
(NODE_POS_NEED_UPDATE(node_p))
                                        {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_lll) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_lll;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lll) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_lll;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lll) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_lll;
                                        }
                                }
                        }
                        else if (THREAD_ID(tf) == 640)
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                        NODE_X(node_p) =
(NODE_X(node_p) - xr_lll) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_lll;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lll) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_lll;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lll) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_lll;
                                }
```

```
                         }

                         else if (THREAD_ID(tf) == 435)
                         {
                                 if ((n1_lul*NODE_X(node_p) +
n2_lul*NODE_Y(node_p) + n3_lul*NODE_Z(node_p) + D_t_lul < 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                 NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                         }
                                 }
                                 else if ((n1_lul*NODE_X(node_p) +
n2_lul*NODE_Y(node_p) + n3_lul*NODE_Z(node_p) + D_t_lul >= 0) &&
(n1_lul*NODE_X(node_p) + n2_lul*NODE_Y(node_p) + n3_lul*NODE_Z(node_p)
+ D_b_lul <= 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                 NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                 d_m2t_lul =
fabs(n1_lul*NODE_X(node_p) + n2_lul*NODE_Y(node_p) +
n3_lul*NODE_Z(node_p) + D_t_lul) / sqrt(n1_lul*n1_lul + n2_lul*n2_lul +
n3_lul*n3_lul);

                                                 NODE_X(node_p) =
(NODE_X(node_p) - xr_lul) *
                                                         (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lul / D_lul *
M_PI))) /
                                                         (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lul / D_lul * M_PI))) + xr_lul;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lul) *
                                                         (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lul / D_lul *
M_PI))) /
                                                         (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lul / D_lul * M_PI))) + yr_lul;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lul) *
                                                         (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_lul / D_lul *
M_PI))) /
                                                         (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_lul / D_lul * M_PI))) + zr_lul;
                                         }
                                 }
                                 else if ((n1_lul*NODE_X(node_p) +
n2_lul*NODE_Y(node_p) + n3_lul*NODE_Z(node_p) + D_b_lul > 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                 NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_lul) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_lul;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lul) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_lul;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lul) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_lul;
                                        }
                                }
                        }
                        else if (THREAD_ID(tf) == 430)
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                        NODE_X(node_p) =
(NODE_X(node_p) - xr_lul) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_lul;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - yr_lul) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_lul;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - zr_lul) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_lul;
                                }
```

```
                         }

                         else if (THREAD_ID(tf) == 7)
                         {
                                 if ((n1_rll*NODE_X(node_p) +
n2_rll*NODE_Y(node_p) + n3_rll*NODE_Z(node_p) + D_t_rll < 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                 NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                         }
                                 }
                                 else if ((n1_rll*NODE_X(node_p) +
n2_rll*NODE_Y(node_p) + n3_rll*NODE_Z(node_p) + D_t_rll >= 0) &&
(n1_rll*NODE_X(node_p) + n2_rll*NODE_Y(node_p) + n3_rll*NODE_Z(node_p)
+ D_b_rll <= 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                 NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                 NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                 NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                         (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                d_m2t_rll =
fabs(n1_rll*NODE_X(node_p) + n2_rll*NODE_Y(node_p) +
n3_rll*NODE_Z(node_p) + D_t_rll) / sqrt(n1_rll*n1_rll + n2_rll*n2_rll +
n3_rll*n3_rll);

                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rll) *
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rll / D_rll *
M_PI))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rll / D_rll * M_PI))) + xr_rll;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rll) *
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rll / D_rll *
M_PI))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rll / D_rll * M_PI))) + yr_rll;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rll) *
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rll / D_rll *
M_PI))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rll / D_rll * M_PI))) + zr_rll;
                                        }
                                }
                                else if ((n1_rll*NODE_X(node_p) +
n2_rll*NODE_Y(node_p) + n3_rll*NODE_Z(node_p) + D_b_rll > 0))
                                {
                                        if
(NODE_POS_NEED_UPDATE(node_p))
                                        {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rll) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rll;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rll) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rll;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rll) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rll;
                                        }
                                 }
                         }
                         else if (THREAD_ID(tf) == 5)
                         {
                                 if (NODE_POS_NEED_UPDATE(node_p))
                                 {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                        NODE_X(node_p) =
(NODE_X(node_p) - xr_rll) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rll;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rll) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rll;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rll) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rll;
                                 }
                         }

                         else if (THREAD_ID(tf) == 870)
                         {
                                 if ((n1_rml*NODE_X(node_p) +
n2_rml*NODE_Y(node_p) + n3_rml*NODE_Z(node_p) + D_t_rml < 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                         NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                         NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                         NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                         }
                                 }
                                 else if ((n1_rml*NODE_X(node_p) +
n2_rml*NODE_Y(node_p) + n3_rml*NODE_Z(node_p) + D_t_rml >= 0) &&
(n1_rml*NODE_X(node_p) + n2_rml*NODE_Y(node_p) + n3_rml*NODE_Z(node_p)
+ D_b_rml <= 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                         NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                         NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                         NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                 (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                d_m2t_rml =
fabs(n1_rml*NODE_X(node_p) + n2_rml*NODE_Y(node_p) +
n3_rml*NODE_Z(node_p) + D_t_rml) / sqrt(n1_rml*n1_rml + n2_rml*n2_rml +
n3_rml*n3_rml);

                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rml) *
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rml / D_rml *
M_PI))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rml / D_rml * M_PI))) + xr_rml;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rml) *
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rml / D_rml *
M_PI))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rml / D_rml * M_PI))) + yr_rml;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rml) *
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rml / D_rml *
M_PI))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rml / D_rml * M_PI))) + zr_rml;
                                         }
                                 }
                                 else if ((n1_rml*NODE_X(node_p) +
n2_rml*NODE_Y(node_p) + n3_rml*NODE_Z(node_p) + D_b_rml > 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rml) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rml;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rml) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rml;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rml) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rml;
                                        }
                                }
                        }
                        else if (THREAD_ID(tf) == 868)
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                        NODE_X(node_p) =
(NODE_X(node_p) - xr_rml) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rml;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rml) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rml;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rml) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rml;
                                }
```

```
                         }

                         else if (THREAD_ID(tf) == 204)
                         {
                                 if ((n1_rul*NODE_X(node_p) +
n2_rul*NODE_Y(node_p) + n3_rul*NODE_Z(node_p) + D_t_rul < 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
                                         }
                                 }
                                 else if ((n1_rul*NODE_X(node_p) +
n2_rul*NODE_Y(node_p) + n3_rul*NODE_Z(node_p) + D_t_rul >= 0) &&
(n1_rul*NODE_X(node_p) + n2_rul*NODE_Y(node_p) + n3_rul*NODE_Z(node_p)
+ D_b_rul <= 0))
                                 {
                                         if
(NODE_POS_NEED_UPDATE(node_p))
                                         {

NODE_POS_UPDATED(node_p);
                                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                                (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                d_m2t_rul =
fabs(n1_rul*NODE_X(node_p) + n2_rul*NODE_Y(node_p) +
n3_rul*NODE_Z(node_p) + D_t_rul) / sqrt(n1_rul*n1_rul + n2_rul*n2_rul +
n3_rul*n3_rul);

                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rul) *
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rul / D_rul *
M_PI))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rul / D_rul * M_PI))) + xr_rul;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rul) *
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rul / D_rul *
M_PI))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rul / D_rul * M_PI))) + yr_rul;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rul) *
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * time)) * (0.5 - 0.5*cos(d_m2t_rul / D_rul *
M_PI))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0))) * (0.5 -
0.5*cos(d_m2t_rul / D_rul * M_PI))) + zr_rul;
                                        }
                                 }
                                 else if ((n1_rul*NODE_X(node_p) +
n2_rul*NODE_Y(node_p) + n3_rul*NODE_Z(node_p) + D_b_rul > 0))
                                 {
                                        if
(NODE_POS_NEED_UPDATE(node_p))
                                        {

NODE_POS_UPDATED(node_p);
                                                NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;
```

```
                                                NODE_X(node_p) =
(NODE_X(node_p) - xr_rul) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_x2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rul;
                                                NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rul) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_y2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rul;
                                                NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rul) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                        (1 + 0.5 * dt_z2
* (1 - cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rul;
                                        }
                                }
                        }
                        else if (THREAD_ID(tf) == 201)
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);
                                        NODE_X(node_p) =
(NODE_X(node_p) - x0) * (1 + 0.5 * dt_x * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + x0;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - y0) * (1 + 0.5 * dt_y * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + y0;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - z0) * (1 + 0.5 * dt_z * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + z0;

                                        NODE_X(node_p) =
(NODE_X(node_p) - xr_rul) * (1 + 0.5 * dt_x2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_x2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + xr_rul;
                                        NODE_Y(node_p) =
(NODE_Y(node_p) - yr_rul) * (1 + 0.5 * dt_y2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_y2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + yr_rul;
                                        NODE_Z(node_p) =
(NODE_Z(node_p) - zr_rul) * (1 + 0.5 * dt_z2 * (1 - cos(2 * M_PI / Tc *
time))) /
                                                (1 + 0.5 * dt_z2 * (1 -
cos(2 * M_PI / Tc * MAX(time - dtime, 0.0)))) + zr_rul;
                                }
```

```
                         }

        }
    }
        end_f_loop (f, tf)

}

DEFINE_GRID_MOTION(GlottalMotion, domain, dt, time, dtime)
{
        Thread *tf = DT_THREAD(dt);
        face_t f;
        Node *node_p;
        double  xr, yr, zr, xx1, x2, dg_ratio;
        int n;

        xr = 66e-3;
        yr = 0.2979e-3;
        zr = 79.7044e-3;
        xx1 = 56e-3;
        x2 = 76e-3;
        dg_ratio = 1.4;

        SET_DEFORMING_THREAD_FLAG(THREAD_T0 (tf));


        begin_f_loop(f, tf)
        {
                f_node_loop(f, tf, n)
                {
                        node_p = F_NODE(f, tf, n);

                        if (NODE_X(node_p) >= xx1 && NODE_X(node_p) <=
x2)
                        {
                                if (NODE_POS_NEED_UPDATE(node_p))
                                {
                                        NODE_POS_UPDATED(node_p);

                                        NODE_Y(node_p) = (NODE_Y(node_p)
- yr)*(1 + pow(sin((NODE_X(node_p) - xx1) / (x2 - xx1)*M_PI),
4)*cos(time * 2 * M_PI / Tc - M_PI / 2)*(dg_ratio - 1))
                                                / (1 +
pow(sin((NODE_X(node_p) - xx1) / (x2 - xx1)*M_PI), 4)*cos(MAX(time -
dtime, 0.0) * 2 * M_PI / Tc - M_PI / 2)*(dg_ratio - 1)) + yr;
                                }
                        }

                }
        }
        end_f_loop(f, tf)
}
```

VITA

Jianan Zhao

Candidate for the Degree of

Doctor of Philosophy

Dissertation: UNVEILING ADVANCED MECHANISMS OF INHALABLE DRUG AEROSOL DYNAMICS USING COMPUTATIONAL FLUID DYNAMICS AND DISCRETE ELEMENT METHOD

Major Field: Chemical Engineering

Biographical:

Education:

Completed the requirements for the Doctor of Philosophy in Chemical Engineering at Oklahoma State University, Stillwater, Oklahoma in July, 2021.

Completed the requirements for the Master of Science in Mechanical Engineering at University of Southern California, Los Angeles, California in May, 2015.

Completed the requirements for the Bachelor of Engineering in Thermodynamics and Power Engineering at Southeast University, Nanjing, China in June, 2013.

Experience:

Graduate Research Assistant, School of Chemical Enginnering, Oklahoma State University 2017-2021

Professional Memberships:

Reviewer of American Journal of Chemical and Biochemical Engineering.
Member and reviewer of American Association for Aerosol Research.