THE UNIVERSITY OF
SYDNEY

School of Aeronautical, Mechanical and Mechatronic Engineering

---

A better method of testing neonatal ventilation equipment

---

by

Corinne Theresa Gard
440314347

Co-Primary Supervisors
Dr. Mark Brian Tracy (Westmead Hospital NICU)
Prof. Alistair McEwan (AMME Faculty USYD)

Assistant Supervisor
Dr. Murray Kenneth Hinder (Westmead Hospital NICU)

An honours thesis submitted in fulfilment of the requirements for the degree of Bachelor of Biomedical Engineering

Student/Departmental Disclaimer

The work contained in this thesis is substantially my own, and to the extent that the work is not my own, I have indicated that it is not my own by acknowledging the source of those parts. I have read and understood the *University of Sydney Student Plagiarism: Coursework Policy and Procedure* documentation, and I understand that failure to comply can lead to the University commencing proceedings against me for potential student misconduct.

This thesis was prepared for the school of Aerospace, Mechanical and Mechatronic Engineering at the University of Sydney, Australia. The opinions, conclusions and recommendations therein are those of the author and do not necessarily reflect the University of Sydney, the NICU department at Westmead Hospital or any other parties involved in this project.

Statement of Contribution

The list below represents Corinne Gard's contribution:

- I designed the procedure, including the test parameters according to which the lung model, GINA, would be validated.
- I wrote the software GINA VALIDATOR. The code design is completely my own, except for the Bland-Altman Analysis code which has been referenced.
- I communicated with Dr Peter Schaller, inventor of GINA to discuss changes to GINAs design that would meet the Westmead NICUs testing needs.
- Based on the needs expressed by clinicians, I designed and wrote the GINA ANALYSER software for adjusting raw GINA output data and calculating and outputting useful breath by breath data. The code design is completely my own, except for some of the exOpen.m file which has been referenced.
- I designed the testing procedures to determine the validity and usability of the GINA ANALYSER software.
- I gathered and processed all results in this thesis.
- I have written this thesis. It is all my own work, with references.

Signed (Signature, Date):

Corinne Theresa Gard

Dr. Mark Tracy

Dr. Alistair McEwan

Acknowledgements

This thesis is dedicated to those people whose belief in me and support made this project a positive experience. I want to especially thank the following people.

Firstly, to my supervisors, co-supervisors and academics. Mark Tracy and Murray Hinder, I am so grateful that you saw potential in me and gave me such a great opportunity to do something real. Thank you for your willingness in helping me to learn, your flexibility with the project and your patience. To Thomas Drevhammer, thank-you for taking time to teach me and give your insights on the projects direction, you helped give the project new life. To Alistair McEwan and Graham Brooker, thank you for your valuable advice on approaching tasks as an engineer. To Dr. Peter Schaller, thank you for letting me tinker with your invention and having the kindness to correspond with me and provide me invaluable information.

Secondly, to my family and friends. To Mum, Dad, Kane, Riley and Cheeky cat. Thank you for your love and support that has kept me sane. To Prash, thank you for not letting me let myself down and for always being by my side. Thank you to Marly and Brooke for helping me believe in myself. Thank you to Zoe and Clara for allowing me to bombard you with statistics questions, even on NYE. And finally, thank you to the Boys and many others for making sure that I kept having fun throughout my thesis. I love you all.

Finally, this thesis is dedicated to the sick babies and their families around the world. I hope this work is a part of the move to improve treatments and helps to save lives.

Executive Summary

Recently a number of publications have reported concerning findings of regular potentially life-threatening faults in a number of devices used for providing life-preserving ventilation to sick neonatal infants. Alarmingly, all of these devices were approved for patient-use by the regulatory administrations whose purpose is to assess medical devices to prevent avoidable patient harm. This begs the question: to what extent are morbidities in neonatal intensive care unknowingly caused by faulty therapeutic devices? The urgency of the matter has forced neonatal intensive care units to take equipment testing into their own hands.

The primary purpose of this thesis is to design a comprehensive testing procedure for neonatal ventilation equipment in order to assess whether they are delivering appropriate therapy. The experimental process for this study had two components. The first experiment involved validity testing a recently developed, first-of-its-kind mechanical-software hybrid neonatal lung model known as GINA. It was determined that GINA was able to replicate realistic behaviour of a breathing infant
Through assessing...accuracy... realistic behaviour. but its software was lacking. The second was...
It is intended that based on the tests... TGA

Contents

List of Figures

List of Tables

Abbreviations

CGA Corrected Gestational Age

CNS Central Nervous System

CPAP Continuous Positive Airway Pressure

ETT Endotracheal tube

FDA Federal Drug Administration (USA)

FRC Functional Residual Capacity

GA Gestational Age

GHTF Global Harmonisation Task Force

GUI Graphical User Interface

ICE International Electrotechnical Commission

ISO International Standards Organisation

MCM Multi-compartment model

NALM Neonatal Active Lung Model (Schaller)

NICU Neonatal Intensive Care Unit

NL Neonatal Lung

PEEP Positive End-Expiratory Pressure

PIP Peak Inflation Pressure

Raw Airway Resistance

SCM Single-compartment model

SIB Self Inflating Bag Resuscitators

TGA Therapeutic Goods Administration

TPR T-Piece Resuscitators

VG Volume Guarantee

1 Introduction

1.1 Overview

Worldwide, approximately 7% of newborn infants experience respiratory distress requiring basic ventilation support in order to provide sufficient pulmonary oxygenation and gaseous exchange to sustain life [1]. One third of these infants go on to require advanced therapy in the form of endotracheal intubation or nasal prongs to deliver mechanical ventilation, and medication [2]. In this field of Neonatal Intensive Care the goal of 'sustaining life' is not as simple as merely keeping the patient alive, but encompasses ensuring the integrity of the central nervous system and organs so that the patient can pursue a life devoid of severe health morbidities.

Neonatal respiratory conditions are particularly complicated to treat due to the limited understanding of the physiological nature of neonatal lungs, and the delicate and unstable nature of the infant patient. For example, we still do not fully understand the transition process from having lungs filled with amniotic fluid to the babies first breath [3, 4]. There is also still uncertainty regarding the chronology of the development of respiratory tissue before and after birth and how premature birth interrupts this development [5]. Thanks to improvements in the medical field, newborns of only 23-24 weeks gestation are able to survive. Therefore, depending on the infant's corrected gestational age (CGA), the nature of the birth (ceasarean before or after labour, induced or natural labour vaginal birth), stage of pulmonary development and pathology, there is significantly more versatility of pulmonary physiology in neonates than healthy adults [6, 7, 8].

Given how small and delicate neonatal infant's are, it is very easy to cause extreme damage through delayed or erroneous treatment. Hyperoxia, hypoxia and hypercapnia can cause severe long term impairments including cerebral palsy, blindness, or lead to mortality [2, 9]. Over-pressurization can stretch or burst infant's lungs, causing lifelong disability or death. Mechanical ventilation for prolonged periods often causes a chronic respiratory disease known as bronchopulmonary dysplasia (BPD) which may cause breathing difficulty and lung infections [1]. Se-

lecting appropriate respiratory treatment for neonates is a balancing act of oxygenation, pressurisation, volume and breath frequency. Additionally, neonates with a pathology do not breathe with regularity and their stability may rapidly change, leading to vastly different treatment needs [4, 10, 11, 6]. Despite this, clinicians and/or mechanical ventilators must attempt to deliver appropriate treatment in real time.

In the field of critical care ventilation, the Australian Therapeutic Goods Administration (TGA) relies on device standards set by the international organisation of standardisation, ISO. Despite the complexities of treating and developing treatments for neonatal respiratory conditions, it would be reasonable to expect that the Regulatory Standards would ensure that patient risks are avoided.Such standards should be kept up-to-date with the most conclusive scientific literature and with technology. Unfortunately, this is not the case. Here are some poignant examples of their inadequacy.

1. There is a lack of standards for safe ventilation that particularly address the neonatal weight range (approx. 500gms to 10kgs). The few that do treat this entire weight range as one group, even though the variation of lung function and mechanical characteristics across this range is immense. Standards should be far more prescriptive to smaller weight intervals and consider more parameters than only weight.

2. The scope of standards for mechanical ventilators does not address software reliability and validity [12].

3. Where standards do not exist, the TGA permits devices to enter the market based on a recommendation by a clinician [13]. Where standards do exist, companies may have their device skip evidential requirements of the regulatory process due to its 'similarity' to other devices on the market [13, 14]. Both of these create opportunities for unscrupulous companies to have their dysfunctional or dangerous devices fast-tracked to the market [14].

4. Although jurisdictions including the TGA expect manufacturers to immediately report adverse effects, this mandate is presently more of an ethical expectation than an enforced and audited policy [14]. Therefore, there is under or inconsistent reporting. This means that devices are not recalled nearly as soon or often as required.

Perhaps the most unjust consequence of these four points is that by proxy, it renders the first users of any device, either clinician or patient, as non-consenting test subjects. One only has to look at the horrendous case of *Johnson and Johnson*'s

vaginal mesh to see that surgeons were duped into believing they were providing a therapeutic implant and women were duped into thinking it would help prevent the prolapse process. Patients have to live with the consequences of this avoidable mistake for life.

It is therefore not surprising that independent studies have found evidence of dangerous therapy that almost certainly has caused patient harm in the field of neonatal resuscitation, including self-inflating bags for manual resuscitation that provide no air or too much, and ventilator faults that cause lung overdistension [15, 16]. Recent studies have shown that different ventilator models on identical settings have significant variability of delivery [17, 18]. Given the shortcomings of the present Standards, independent studies are essential to ensure the safety of some of our most fragile patients. Dr Mark Tracy, director of the Neonatal Intensive Care Unit (NICU) at Westmead Hospital, Australia, is leading the charge in this field of research. As part of an effort to prevent avoidable damage to newborn babies, he has proposed a multitude of investigations into ventilation devices. Should it be found that they are erroneous, recommendations shall be made to the Therapeutic Goods Administration Australia (TGA) and the International Organization of Standardisation (ISO). Additionally, it advantageous for clinicians to do comparative testing of ventilators before committing to purchase or clinical use is hugely advantageous [19]. The prerequisite for this testing is that a valid mechanical model of a neonatal lung is developed. It should have the capacity to interface with ventilation devices and output real-time data of the nature of this interaction that can be used for identification of therapeutic device dysfunction. The purpose of this thesis is to fulfil this need.

1.2 Aims

The purpose of this thesis is to design a comprehensive testing procedure for equipment used for neonatal ventilation in order to assess whether they are delivering appropriate therapy. This is to be achieved by:

1. Verifying and validating a mechanical neonatal lung simulator and software.
2. Designing software programme to interface with the simulator in order to output breath by breath variables for statistical analysis.
3. Testing the response of lung model to by different therapeutic devices.

1.3 Thesis Structure

1.3.1 Literature Review

The Literature Review section of this paper conveys the of the present state of research in areas relevant to the experiments undertaken in this thesis. It first summarises the physiological and mechanical nature of the neonatal lung (NL), then evaluates attempts to model the NL computationally and mechanically. Next, it describes devices on the market that we at the Westmead NICU particularly believe must undergo bench testing by a system such as that designed in this thesis . Finally, it presents the current state of Regulatory Standards and where they need to be improved.

1.3.2 Core Study One: Verification and Validation of Neonatal Active Lung Model

This study is devoted to the experimental design for validating the Neonatal Active Lung Model (NALM) with software known as GINA, which has been designed by *Dr. Schaller Medizintechnik* ™as the first mechanical lung prototype on the market to purely model a neonatal lung. No publicly available publication assessing its validity exists. The NALM's internal tubes and components are to scale of a real infant. These and software input parameters, including compliance, are adjustable to represent neonates of varying sizes. Therefore if valid, it will be far more accurate than existing models which introduce error by consequence of adult sized components and non-variable compliance. It is found that the NALM is valid enough to be used for device testing, though the GINA software does not have all the functionality desired and does output data in a meaningful way.

1.3.3 Core Study Two: Software development for useful breath-by-breath output

The purpose of this study is to fulfil the need for data from GINA to be output in a meaningful way. Specifically, the focus of this study is to create a software which will read data continuous data from GINA and transform it into a spreadsheet of breath-by-breath calculated parameters which can be used for statistical analysis. By the end of this study, this software known as the GINA ANALYSER is capable of analysing recordings of spontaneous and non-spontaneous breaths up to a frequency of 120 breaths per minute. This simple application can be used by clinicians and will be used for device testing in the Westmead NICU.

1.3.4 Case Study: Interfacing ventilators with neonatal patients

This section fulfills the need to demonstrate understanding of the syllabus for the subject *Biomechanics and Biomaterials* (MECH4961). Given that biomechanics of the neonatal lung have been discussed in depth in the Literature Review, this shortened case study primarily focusses on covering the biomaterials aspect of the course. This section discusses how ventilators interface with neonatal patients, specifically through endotracheal tubes, and explores the impact they have on the endotracheal tissue and how the technology has been improved to reduce adverse impacts over time.

1.3.5 Case Study: Regulations in the Medical Device Industry- The Predicate Problem

This section fulfills the need to demonstrate understanding of the syllabus for the subject *Regulatory Affairs in the Medical Industry* (AMME4992). It expands on what was introduced by the Literature Review section *Shortfallings of Current Regulatory Standards*, which focusses on Regulations in Neonatal Respiratory treatment. It first explores the current state of the "Global Harmonisation Task Force" (GHTF) group of regulatory bodies worldwide. Then it goes on to discuss the issue of "predicate creep" and evaluates whether the GHTF is moving in a direction to reduce the likelihood of this occurring.

1.3.6 Overall Discussion and Conclusions

This section discusses the overall findings and points of interest from the accumulated knowledge from the Core Studies and Case Studies. It reflects on where the Core Studies have left the field of Neonatal Resuscitation today and provides direction regarding the future steps for this project of improving testing of neonatal ventilation equipment.

1.3.7 Managing Work Health and Safety Risks

This section fulfills the requirements of the *Work Health and Safety Act (2011), Section 274*, the *Code of Practice: How to Manage Work Health and Safety Risks* [20]. It includes a risk assessment which identifies hazards I may encounter in my workplace at Westmead Hospital and assesses these risks, with details how they have been managed.

2 | Literature Review

2.1 Background

In the modern setting, medical devices increasingly take over the role of a physician in providing life-saving therapy to patients. As such, medical devices should be held to the the same tenets of the Hippocratic oath, specifically, to "either help or do no harm". It is for this reason that regulatory bodies exist to enforce standards on medical devices, including the International Organisation of Standardisation (ISO), the US Food and Drug Administration (FDA), the Australian Therapeutic Goods Administration (TGA) and the International Electrotechnical Commision (ICE). These organisations have to strike a delicate balance between seeking to safeguard the public's health against devices that are unsafe or useless, while allowing beneficial innovations to reach the market as quickly as possible [14]. On the one hand, the standards of evidence backing the efficacy of the new or improved device must be quantified. More stringent standards equate to less risk. But on the other hand, more stringent standards equate to a longer time before a device is available, during which time patients who could be helped won't be. Hence, perfecting the regulation procedures for therapeutic goods is inherently plagued by the sometimes opposing forces of science versus ethics.

Given that developments in medical treatments rely on the interaction of academic progress in the fields of physiology and technology, one thing that is clear is that to control for the most educated and ethical decisions, therapeutic goods administrations must stay on top of the latest research. Unfortunately, in the field of neonatal resuscitation and ventilation support this is far from the case. In recent years a number of publications have been released reporting findings of alarming rates of equipment that pass the standards, yet are defective to the point of providing null or inappropriate therapy. It is almost certain that these devices have been responsible for the death or lifelong health morbidity of patients, with no fault nor knowledge of the physician.

For example, a 2019 investigation by Tracy et. al. found that 50% of 20 self-inflating bag and models failed to provide safe therapy, [16] despite that all models passed International Standards, and "Self-inflating bag and mask" are even recommended by the United Nations and the World Heath Organisation as to be used for "newborns requiring positive pressure ventilation" [12, 21, 22]. Another investigation by Hinder et. al. found that the humidification device in automated ventilators regularly causes water build-up in the respiratory limb, leading to potential patient overpressure exposure [15]. Automated ventilators have vastly improved therapy since the 1970s, but there in the most frequently used modes of ventilation there evidence of asynchrony between breath provisions and babies breath effort, causing large and dangerous fluctuations in intrathoracic pressure [23]. Many modern ventilators have feedback-control modes which attempt to respond to the babies effort and make up the remaining volume/pressure requirements. The problem is that the Standards do not address these modes in any way. Recent studies have shown that different ventilator models on identical settings have significant variability of delivery [17, 18]. Therefore, clinicians have no guarantee that these modes are safe to use and must choose between trialling them on patients, or resorting to older modes with definite but known risks.

Perhaps the most alarming omission of the standards is the lack of guidance concerning the air delivery needs of premature infants. Worldwide, approximately 1 in 10 children are born prematurely . Since 2010, medical innovations have resulted in the viability of extremely preterm infants as young as 23 weeks old (CGA) [24] (see fig. 2.1 for image of premature infant). Devices are simply required to be capable of delivering 5-7mls per kilogram, without pressure specifications [13]. However, alveolar development and lung tissue elastin content begins from the 27th week of gestation and continues years past the postnatal period. Therefore there is a large difference in the compliance of an extremely pre-term infant to a term infant. The ratio of chest cavity size compared with body volume is also higher in smaller infants and therefore the volume and pressure to weight ratios for these patients will be entirely different [4]. While the details of premature infant respiratory physiology and treatment are inconclusive, the above described basic knowledge of the physiological nature of infant lung development is decades old and should at least be a consideration in the Standards. The 'grandfathering' of Standards has posed and continues to pose an unacceptable risk to life.

Positively, in Australia the feedback provided to the TGA based on the studies involving Tracy and Hinder et.al, have prompted a presently underway review of the ISO standards. However the audit of devices can not wait for this update while lives

Figure 2.1: A tiny premature infant. Image property of Jamie and John Florio

are at stake and while technology continues to advance. If there were a bench test apparatus that could interact with devices like a neonate lung and output data from this interaction for analysis, it would be possible to assess respiratory device effectiveness and inform the regulatory bodies and the clinical community. Though there are various bench tests and lung models that have been used in the past, the lack of standardized equipment and testing procedures have made it difficult to establish normative values for the various parameters of interest that are independent of the measurement devices used, or to use these tests as reliable clinical tools [25]. Furthermore, there are concerns regarding the validity of these models [26, 19, 27]. Herein lies the motivation for the research undertaken in this thesis.

The intention of this literature review is to educate the reader on the requirements that a neonatal lung model for testing respiratory devices must meet in order to be a valid and meaningful tool. It is divided into four parts. Part 2.3 brings the reader up to date with the present theoretical understanding of neonatal respiratory physiology and how it is a challenging system to model. Part 2.4 educates the reader on proposals of how this system can be modelled mathematically and mechanically and concludes that a simple single compartment model with compliance is the best option for a bench test. Part 2.5 introduces devices that we at the Westmead NICU believe require thorough testing with an emphasis on mechanical ventilators, and outlines the present risks and clinically observed problematic phenomenon. To conclude, Part 2.6 provides examples of how in light of all of this information, current regulatory standards fall short of where they need be.

2.2 Definitions

Medical terms that are referred to throughout the literature review are defined below.

Neonate An infant within their first 28 days of life [22].

Neonatal Lung An umbrella term to encapsulate the vast range of lungs (with varied physiology and mechanics) that exist across all neonates.

Corrected Gestational Age Refers to a persons age from birth, corrected for prematurity by subtracting how many weeks before 38 weeks gestation the person was born.

Gestational Age Refers to the length of time a premature infant spent in gestation.

Term Birth is classified as 37 to 42 weeks gestation [24].

Preterm Birth is classified as less than 37 weeks gestation [24].

Very Preterm Birth is classified as 28 to 32 weeks gestation [24].

Extremely Preterm Birth is classified as less than 28 weeks gestation [24].

CPAP Continuous Positive Airway Pressure involves the delivery of pressure to sustain inflation of the airways to maintain a PEEP.

PEEP Positive End Expiratory Pressure is the pressure remaining in the lungs that prevents alveolar collapse at the end of a full exhalation [28]. This is often provided by a mechanical ventilator.

PIP Peak Inflation Pressure is the maximum pressure encountered in the thoracic cavity at the end of inhalation [28].

FRC Functional Residual Capacity is the volume of air remaining in the lungs at the end of passive expiration [29].

WOB Work of Breathing is the amount of effort, or work (J) a person must put in energetically to inhale.

Surfactant An oily secretion containing phospholipids and proteins. It reduces surface tension in the liquid coating of the alveolar surface. Therefore is has the role of helping the alveoli stay open [30].

Dead Space The volume of a breath that does not participate in gaseous exchange, usually because it remains in the conducting airways.

2.3 Theoretical Basics of Neonatal Respiration

In this section, the anatomy of the respiratory system is described. An overview of the physiology and mechanics of breathing in the healthy adult is provided as a point of comparison for neonatal respiration. The purpose of this section is to set the scene of the defining parameters of a neonatal lung, to begin the discussion of an ideal neonatal mechanical lung model for bench testing. Furthermore, the below subsection provides the reader with no background in anatomy the necessary education to understand how and why some of the therapeutic devices used can be dangerous.

2.3.1 Functional Anatomy of the Healthy Adult Respiratory System

The primary purpose of the respiratory system is to facilitate gaseous exchange. Cells throughout our body require oxygen for the cellular respiration process that creates energy. The primary by-product of of cellular respiration is carbon dioxide, which needs to be removed from the bloodstream and tissue to prevent acidosis. The exchange of oxygen for carbon dioxide occurs at the alveolar interface, where haemoglobin in arterial blood takes inhaled oxygen to be delivered to cells, and venous blood returns dissolved carbon dioxide to be expired [31]. The respiratory system also has other important roles including defending the body from inhaled pathogens, producing sounds for speaking, facilitating odour detection via the olfactory receptors in the nasal cavity, and indirectly assisting in the regulation of blood pressure via the conversion of angiotensin I to angiotensin II.

The respiratory tract can be divided into two sections: Upper and Lower. The Upper Respiratory Tract (URT) is responsible for large-particle filtering, warming and humidifying of air to protect the lower respiratory structures. It is made up of structures including the external nares (nostrils), nasal conchae, oral cavity, soft palatte, pharynx, epiglottis and larynx. The Lower Respiratory Tract (LRT) has a dendriform morphology: from the trachea it branches into two primary bronchi which channel air to and each lung. The branching continues from primary bronchi through secondary and tertiary bronchi, through bronchioles which finally end in alveoli; the tiny (200 - 500μm diameter) [32] air filled tissue pockets where all gas exchange occurs [30]. Refer to Figure 2.2 for diagrams of the airway and alveoli structures.

The following sections provide further detail regarding the most relevant functions of the respiratory system when it comes to modelling. First, the mechanical prop-

erties of lung tissues are detailed as a basis for understanding normality. Then, defence mechanisms of the respiratory system are briefly described as these are important therapeutic equipment design considerations.

2.3.1.1 Respiratory Mechanics

First, we shall explore the basic mechanics of inspiration and expiration. The lungs are essentially a structured 'sponge' of epithelial tissue, lined with mesothelium tissue and sitting within the pleural cavity [33]. At the base of the pleural cavity and below the lungs is the diaphragm muscle. The active contraction of the diaphragm and intercostal muscles, triggered involuntarily by the brainstem during unforced breathing, increases the volume of the pleural cavity and decreases pleural pressure. This pressure drop causes the lung surface to be pulled outwards, opening up the airways inside. When the alveolar pressure is lower than the atmospheric pressure, the pressure differential results in air being drawn down through the airways, causing inhalation and facilitating gaseous exchange. In a healthy person at rest, exhalation is passive. When contraction of the diaphragm and muscles ceases, the elastic recoil of lung tissue causes the airways to reduce in diameter again and airway pressure increases above atmospheric pressure, forcing air out of the respiratory tract [34].

In a healthy lung, exhalation should not cause complete collapse. Instead, the system should reach a state of force equilibrium whereby lung elasticity favours further emptying but the chest wall favours expansion (see fig. 2.4, (i)). The remaining volume is known as the *functional residual capacity* (FRC) and the remaining pressure in the alveoli is known as the *positive end expiratory pressure* (PEEP). When exhalation is actively forced as far as possible, the remaining volume which prevents the alveoli from collapsing on themselves is the *residual volume* (RV). Maintenance of the RV is assisted by surfactant, a soap-like fluid which is released by cells in the alveolar duct. Surfactant reduces the surface tension within the delicate alveolar sacs, preventing the tissue from sticking to itself and allowing the easy passage of air [33, 34]. When it comes to inhalation, mechanical forces prevent inflation beyond the *total lung capacity* (TLC) in order to prevent tissue damage. TLC is determined by the point where the elastic recoil of the lung overcomes the declining strength of the muscles as the lung expands. To reach TLC would require considerable effort. Therefore a normal breath will only inflate the lung to *tidal volume* (TV) [34]. Refer to Figure 2.4 for simplified diagrams representing the forces involved in inspiration and expiration.

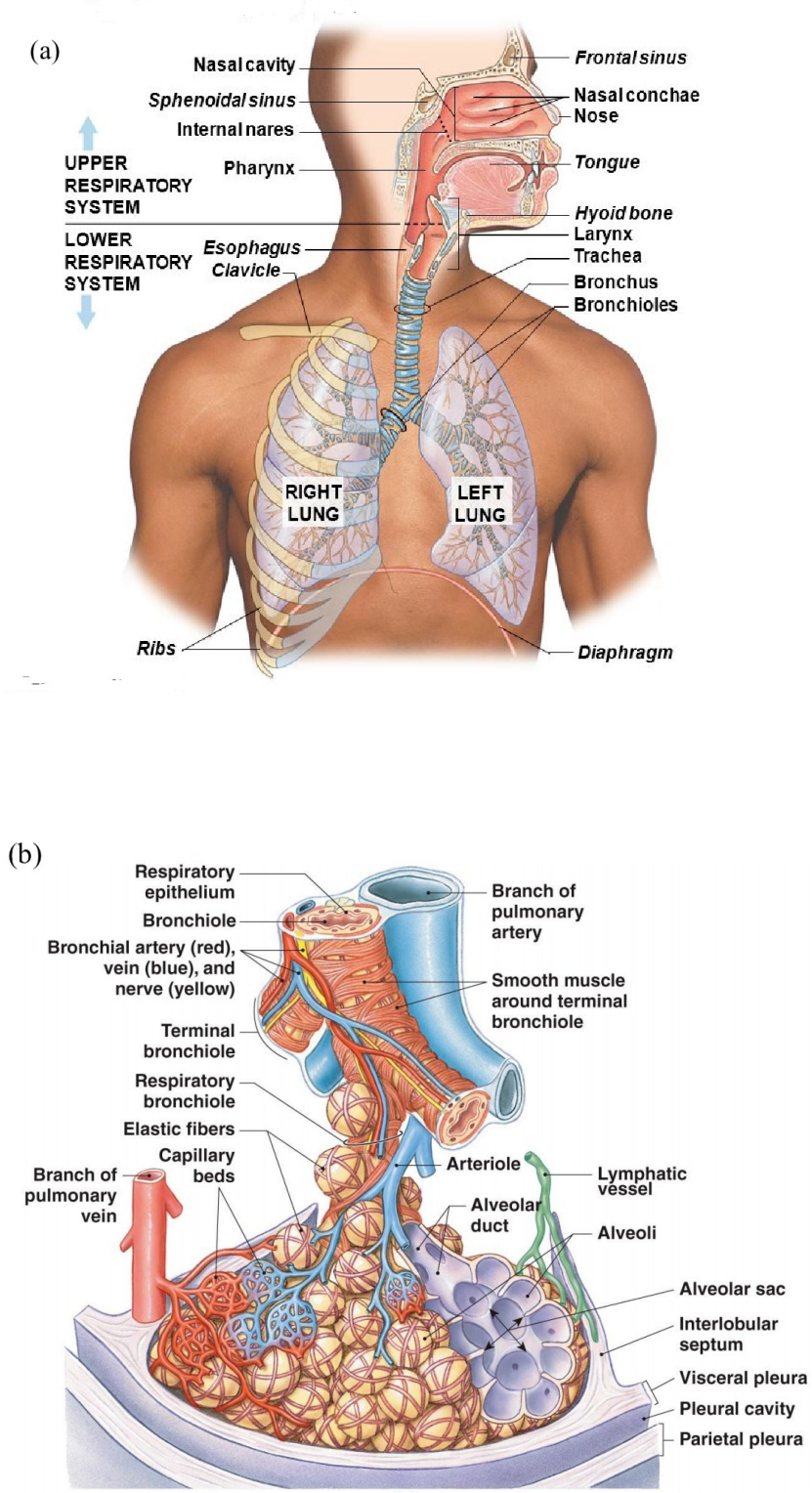


Figure 2.2: Organization of the adult respiratory system. (a)The Upper and Lower Respiratory Systems (b)Anatomy of the terminal bronchiole and alveolar sac. [30]

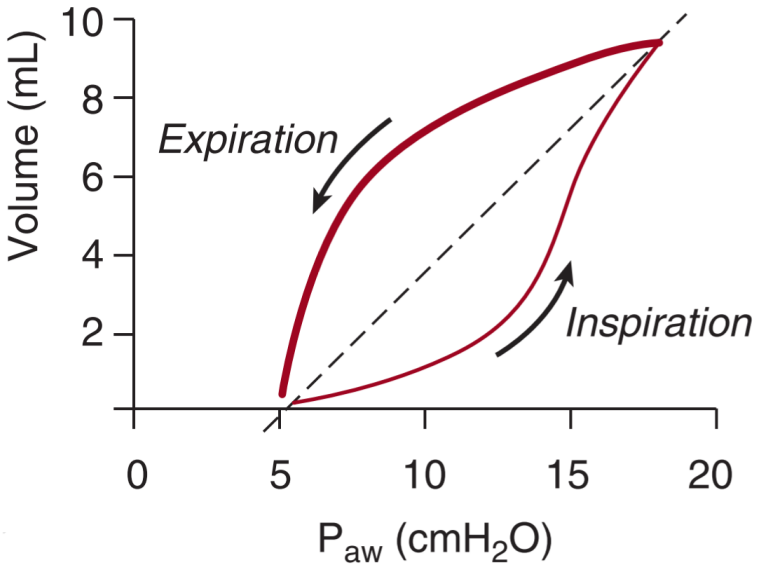


Figure 2.3: A typical volume pressure loop of a ventilated lung, demonstrating hysteresis [29]

In respiratory science, *compliance* refers to the ability of the lung to expand when a force is applied - a lower compliance essentially means the lung is stiffer. Compliance is separated into two different measurements, *static compliance* and *dynamic compliance*. *Static compliance* is a measure of the elastic properties of the lung under static conditions and is measured as the change in lung volume for any given applied pressure. *Dynamic compliance* represents compliance during gas flow, meaning that the pressure measurement is also changing. *Elastance* refers to the elastic recoil of the lung tissue and is approximately inverse to static compliance. *Resistance* refers to the resistance to airflow in the airways. It is well established that the respiratory system is a non linear system and that *compliance*, *elastance* and *resistance* are impacted by the properties of the lung including tissue cellular composition and characteristics, secretions, surface tension, plasticity, viscoelasticity and airway size and proportions, as well as by properties of respiratory activity including breathing frequency and flow [29, 34, 35]. Therefore, the volume-pressure curve of a typical lung during breathing will exhibit a well known property known as *resp*, where the locus of inflation and deflation are different due to the effects of dynamic compliance (see fig. 2.3). The following paragraphs describe the tissue properties of the healthy lung which will be used as a point of comparison for neonatal lungs.

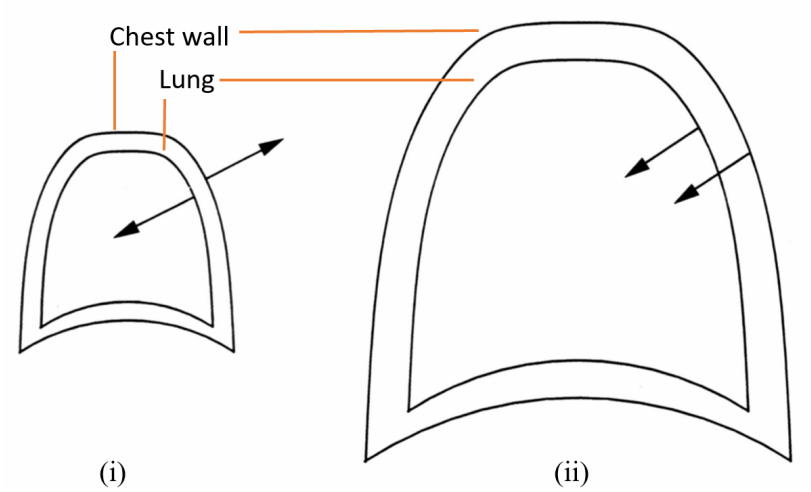
The cellular make-up of respiratory tissue changes at different levels, and therefore the mechanical properties change too. The upper airways - trachea and bronchi

- consist of psuedostratified columnar epithelium supported by cartilage. These walls are also supported by smooth muscle tissue which regulates airway diameter and therefore air distribution throughout the lungs [33]. As one descends the approximately 24 levels of branching of the airways, the thickness of the walls, fraction of cartilage and the amount of cellular pseudostratification decreases. Beneath level 20 in the lower bronchioles and alveoli, there is no cartilage and epithelial cells begin to take on a more squamous shape [36]. By the alveolar sacs, tissue consists of very thin squamous epithelial tissue which forms walls to separate alveoli. These walls are double layered and lined with capillaries. In this way, gas can easily be transported across the tissue into and out of the bloodstream. These delicate sacs remain open thanks to the combined effect of the PEEP and surfactant. Cuboidal surfactant-secreting cells provide the fluid which reduces alveolar surface tension, preventing the flaccid tissue from sticking to itself and allowing the easy passage of air [33].

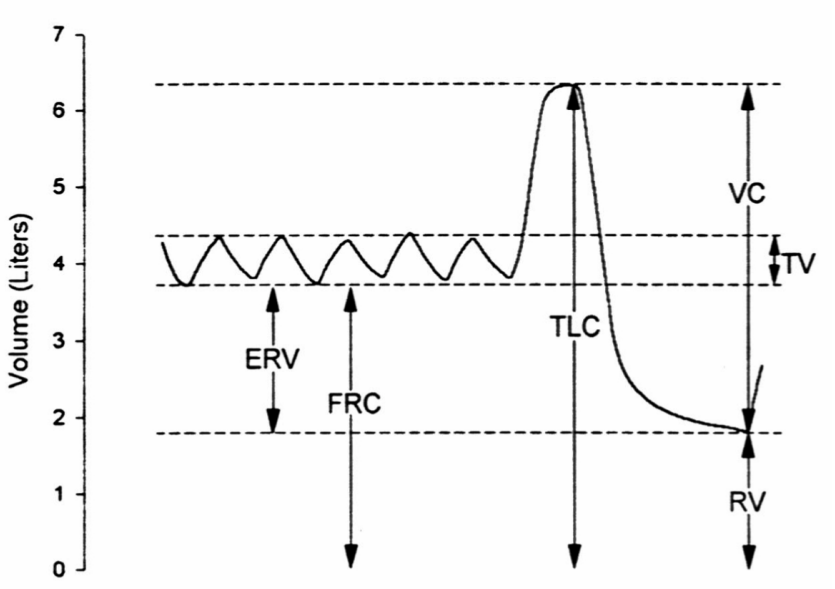
Lung tissues, particularly the lower levels of the airway structure are known to be highly viscoelastic [37]. Additionally, at a microstructural level, the components that make up airways (intraparenchymal connective tissue, surface film, contractile elements) appear to have individual signatures of energy dissipation per expansion-contraction cycle. Despite these factors inducing non-linearity in the nature of energy dissipation across the lung, the fact that synchronous expansion of the alveolar duct exists suggests that the hysteresivities of airway components are relatively well matched [38]. Clearly, if pathology or underdevelopment were to affect the ratio or organization of tissue cells, the lungs would have very different mechanical properties and the patient may not be able to breathe on their own.

2.3.1.2 Defence Mechanisms

The *conducting portion* of the respiratory tract extends from the nasal cavity to the larger bronchioles and is responsible for maintenance and defence. The tissue of this portion is lined with *respiratory mucosa* which consists of a pseudo-stratified ciliated columnar epithelium, and a layer of areolar tissue (*lamina propria*) which contains stem cells for repair and structurally supports the epithelium. Respiratory mucosa is vital for defence of the respiratory system. Mucus released from goblet cells in the bathes the tissue and entraps inhaled debris and larger pathogens (above $5\mu m$ in diameter). This mucus concoction is pushed by beating cilia towards the pharynx, where the mucus and its contents swallowed and dissolved by the acids and enzymes of the digestive tract [30]. Smaller particulates (1 - $5\mu m$) are destroyed by alveolar macrophages, with only particulates smaller than $0.5\mu m$ remaining trapped in the alveoli [30, 39].



(a)



(b)

Figure 2.4: (a) At lower volumes (i), the lung favours emptying but the chest wall favours expansion. At higher volumes (ii), both the lung and the chest wall favour emptying. (b) Spirometry of lung volumes. [34]

2.3.2 Neonatal Respiratory Physiology

The neonatal lung is a variation of the lung described. Particularly, compliance, elastance and resistance are very different compared with a healthy adult lung. It is important to qualify that the term "neonatal lung" (NL) is arbitrary. There is no such single, uniform system that is the "neonatal lung": it is an umbrella term for a wide range or respiratory systems belonging to neonates. The reader is implored to keep in mind this definition of the NL as an umbrella term as they read this thesis.

The traits of the infant's lungs are mediated by three main factors. These are the degree of pulmonary development, the nature of foetal lung fluid clearance at birth, and, pathology. These are described briefly in the following subsections.

2.3.2.1 Stages of Development

In the pulmonary system, functional maturity does not correlate to structural maturity. Therefore in order to treat neonates effectively, it is important to understand to what extent their airways are developed. This can be estimated according to the infant's GA. There are three primary stages of fetal pulmonary development: (1) Pseudoglandular stage (5-17) weeks, Canalicular stage (17-26 weeks), and Saccular stage (24 weeks to birth).

Premature infants in the NICU may be anywhere from 24 weeks old (GA) and therefore may only just be completing the Canalicular phase [4]. At this age, the key developmental markers include: (i) Partial development of pulmonary parenchyma, (ii) Existence of primary, secondary and tertiary bronchi and up to three orders of bronchioles, and, (iii) Creation of airspaces in mesenchyme, which begin to be surrounded by cuboidal epithelium and capillaries. These thin walled saccules are situated at the end of the airways and are not alveolar ducts [4].

At birth the infant's pulmonary system is still not structurally mature. The key developmental markers at the end of the Saccualar stage are: (i) Saccules have developed into smooth walled channels, which transition into alveolar ducts after birth (see fig. 2.5), (ii) Capillaries have increased in density. They are embedded in a cellular interstitual layer that has low elastin content. Elastin content increases as alveoli form, though only 10% of alveoli have formed by 38 weeks GA. (iii) The surfactant secretion system has matured.

Postnatally (or after 38 weeks GA) sacs of cuboidal epithelium and elastin continue to develop into capillarised alveoli. More than 90% of alveoli are formed after birth [5]. It is unknown exactly when in childhood the lung is fully formed [4], though recent estimates tend towards age 8 years old [5]. Furthermore, there are general proportional differences between an infant compared with a child or adult that affect airflow through the lungs. Due to the relatively large head to body ratio in a neonate, there is greater anatomical dead space. The epiglottis is also situated higher in the pharynx and is relatively large, which results in a lower airflow resistance in the nasal passage and explains why infants breath preferentially through their nose [40].

Given this description of the developmental changes at the tissue level, less developed lungs tend to have higher compliance, lower elastance and higher airway resistance. High compliance and low outward elastic recoil are caused by thinner airways with low elastin content, and less alveoli septa separating alveoli. For the infant this means that forceful inspiration can cause collapse of the upper airways. After exhalation the lower airways may be too weak to hold the FRC and collapse due to low transpulmonary pressure. This is partially compensated for by the tendency of the infant to take its next breath before elastic equilibrium volume (EEV) is reached [41]. The smaller airway diameters in neonates causes higher airflow resistance as resistance is inversely proportional to the fourth power of the airway radius. For the same reason, the narrowing of the endotrachea by secretions has a far higher resistive impact in the neonate than the adult. Efficiency of breathing is also reduced in infants due to the diaphragm and intercostal muscles containing less type 1 muscle fibres, which control slow endurance [40].

Ultimately, these impedences to breathing mean that the infant's work of breathing (WOB) is much higher than a healthy adult. Additionally, the levels of the CNS responsible for controlling breathing pattern begin to develop early in gestation are incomplete until weeks to months after term birth. Between fatigue or lack of neural communication, preterm infants often have irregular breathing patterns and may experience apneas (periods of non-breathing) which can be life threatening or cause neural tissue death. The ventilatory response to hypercapnea and hypoxia are also impaired due to incomplete formation of the systems required for peripheral and central chemoreceptor response [40].

2.3.2.2 Respiratory Transition at Birth

At birth, an infant is required to rapidly clear its air spaces of fetal lung fluid that has been occupying its lungs throughout its entire development and effectively self-

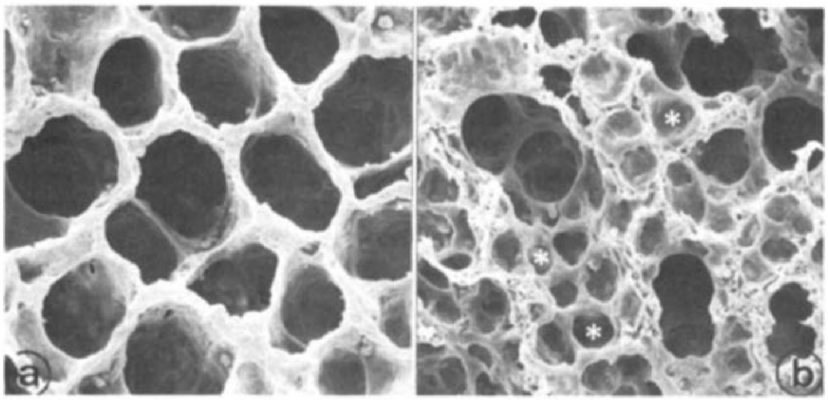


Figure 2.5: Postnatal alveolar formation in the rat lung, by SEM, magnification 230x. a) Age 4 days: large and smooth contoured airspaces. b) Age 8 days: crests and ridges have appeared dilineating shallow alveoli (asterisks). [4]

resuscitate. This complex process is mediated by changes in the hormonal state of fetus and mother that occur during labour which induce rapid maturation of the infant's pulmonary system, as well as the mechanical forces applied to the infant during vaginal delivery. If this process is disrupted, such as due to emergency induced premature labour or birth by ceasarian section, this transition may not be successfully completed. Deprivation of the necessary hormonal changes or premature birth will result in an underdeveloped pulmonary system. The absence of the mechanical forces contributed by vaginal delivery can result in retention of fluid in air spaces, leading to alveolar hypo-ventilation [8]. In either case, the infant will require resuscitation and will likely be admitted to the NICU for ongoing surfactant therapy and ventilation assistance via mask or intubation. The most comprehensive model of the transition of an infant's lungs between the womb and their first breath is a three stage process, described by Hooper et al [42]. These stages are:

1. *Stage one.* Liquid clearance from the airways. Gas exchange cannot occur.

2. *Stage two.* Temporary accumulation of liquid from the airways within the interstitual tissue compartment. Gas exchange is compromised because the increase in interstitual pressure means a high probability of liquid re-entry into the lungs if the fluid is not rapidly cleared via respiratory activity or application of positive pressure (CPAP)

3. *Stage three.* Liquid clearance from the tissue via the bloodstream, tissues and lymphatic system. Regular gas exchange can occur.

A neonatal lung may exist in any one of these stages. As a result, modelling and treatment designs and decisions must take these drastically different systems into account.

2.3.2.3 Common pathologies and their impact on the mechanics of the respiratory system

Naturally, pathologies impacting the respiratory system will change the physiology and mechanics of it. There are many pathologies that neonates may have, and often they may have many simultaneous conditions interacting. Some pathologies are chronic lung diseases whilst others, such as acidosis, impact the whole body and though quickly rectifiable, are deadly. Some can be caused by invasive ventilation procedures, including bronchopulmonary dysplasia and pneumothorax. Seven of the most common pathologies experienced by neonates, their pathogenesis (if known) and the impact on the respiratory system are listed in the table 2.1. There is some further detail on conditions that are caused by intubation and ventilation in the section titled *Case Study One: Interfacing ventilators with neonatal patients*.

Table 2.1: Common Respiratory Pathologies in Neonates [1]

Condition	Pathogenesis	Impact
Bronchopulmonary Dysplasia	Damaged alveolar sacs and airways, usually caused by prolonged ventilation. Can be chronic.	Usually results in less compliant lungs, requiring a greater WOB. Also greater risk of infections.
Transient Tachypnoea	Excessive lung fluid. Non chronic.	While the fluid is in the lungs the infant will require assistance getting proper oxygenation.
Respiratory Distress Syndrome	Deficiency of surfactant. Can cause chronic hyaline membrane disease. Can be chronic.	Dead cell membranes coat the alveoli, preventing gaseous exchange.
Pneumonia	A lower respiratory tract infection. Non chronic.	Fluid on the lungs and the resultant ruptured cell membranes can cause difficulty aquireing proper oxygenation, and difficult treatment due to the extra sensitivity and weakness of the tissue.
Meconium Aspiration Syndrome	Aspiration of milk, blood or faeces during or after birth. Non chronic.	Can cause pulmonary hypertension. Inhaled meconium can cause airway obstruction, infection leading to swelling, and chemical pneumonitus.
Persistent Pulmonary Hypertension	The failure of pulmonary vascular resistance to decrease after birth. A primary or secondary cause of respiratory distress. Non chronic.	Can cause acidosis due to the demand for extra oxygenation to help lower vascular resistance.
Pneumothorax	Air presence in the pleural space	Inability to regulate pressure gradients needed for inspiration and expiration.

2.4 Neonatal Lung Modelling

When it comes to the mechanically modelling the lung, there is a dichotomy between the accuracy of a complex, multi-compartment model (MCM) and using a simple single-compartment model (SCM). A MCM can include the multiple levels of the bronchial tree and their mechanical characteristics. With an MCM, the flow and pressure characteristics throughout the lung will be more realistic and therefore any interaction between the MCM and a device will be more similar to an interaction with a real infant and give a better indication of whether the device is functioning correctly. The problem with an MCM is that the more moving or interfacing parts in a machine, the greater the risk of dysfunction. Additionally, as the tube components that represent airways in the machine will not be made of tissue, and also because of the large variation between individual lungs, it is unlikely that an MCM would contribute significant additional accuracy to a mechanical model. For this reason, MCMs have remained in the domain of computational modelling and SCMs have been the design of choice for bench-top models.

Even though real lungs are by no means single compartment, a good mechanical SCM should replicate the properties of the neonatal lung as best as possible. Internal tube diameters should be diameters that reflect the neonates airways. For example, an extreme premature infant may have an endotracheal diameter as small as 2.5mm, while a term infant's may be up to 5mm [43]. These material of these tubes should be similar in terms of elastance, toughness and internal roughness to bronchial tissue. The volume of the internal compartment must be variable and should be able to deliver appropriate volumes. Perhaps most importantly, the model should have a changeable non linear compliance. Finally, it should be able to replicate the breathing patterns of an infant.

Many papers have been published which detail the design of neonatal specific lung models. These models tend to take two forms: (1) simple bellows or piston models, and (2) hybrid mechanical-software models. A survey of the literature on models determines that hybrid models tend to have more use as the user can program the breathing parameters, but it is the robustness of the mechanical design which really determines the model accuracy [44, 45, 46, 25, 47, 48, 35]. Some of the key considerations include:

- Materials used must have enough placticity such that the pressure volume curves exhibits hysteresis [47].

- The model must be leak free and capable of having PEEP and PIP induced without pressure baseline drift [48].
- The rate of inflation and deflation must be accurate for high and low frequencies and high and low volumes. This can become problematic, for example, when using stepper motors.
- Gas compressibility must be taken into account, and the thermodynamic gas reference model should be adiabatic [44].
- The dead space within the machine should reflect that of an infant lung [26].

There are some available neonatal lung simulators on the market which can be divided into three categories: puppet-like, passive, and active simulators. Puppet-like simulators are realistic appearing models of an infant which can be ventilated via mask, mouth-to-mouth, intubation etc. They are able to imitate active breathing and changes in breathing rate and depth, but not airway resistance, lung compliance or complex breathing patterns and therefore can not be used to evaluate ventilator interaction with pathological states [48, 47]. Examples include the SimNewB®(Laerdal Medical, Wappingers Falls, NY, USA) and Newborn HAL®S3010 (Guamard, Miami, FL, USA). Passive simulators tend to be small machines with a singular or double compartment which allows the user to alter airway resistance and compliance but not simulate spontaneous breathing patters. Examples include the Adult & Infant Test Lung (Michigan Instruments, Grand Rapids, MI, USA) or SmartLung™Infant Test Lung (IMT Medical, Buchs, Switzerland). Finally, active lung simulators are capable of simulating spontaneous breath and interacting with ventilators, in addition to all the features of passive simulators. The most commonly used model on the market is the ASL 5000 Adult/Neonatal Breathing Simulator (Ingmar Medical, Pittsburgh, PA, USA) [47].

2.5 Therapeutic systems

Testing therapeutic systems with an infant model is essential for evaluating performance. This evaluation translates directly into hypotheses on the effects of the therapeutic device with a real infant. The purpose of this section is to briefly introduce some therapeutic systems that we at the Westmead NICU believe require testing. First, mechanical ventilators as a whole are explained to the reader. Mechanical ventilators have vastly improved patient outcomes but even the best new modes appear to still have issues and the patient risks (or lack thereof) have not fully been realised. Itagaki et al. and Krieger et al. have shown that there is significant variability between the performance ventilator models, even when used with

the identical settings. It is likely that these issues are caused by algorithm differences between devices [18, 17].

We in particular want to test the Volume Guarantee mode which is meant to deliver volume based on feedback of the patients own effort to reduce injury. However, it has been clinically witnessed at Westmead Hospital NICU that it may deliver too high pressures with the smallest infants. Second, rPAP pieces are mentioned, including how testing on a mechanical infant will help clinicians and technicians understand whether it is a worthwhile technology.

2.5.1 Mechanical Ventilators

This section begins with an overview of modern mechanical ventilation systems, common ventilation modes and the risks associated with ventilation systems. The aim with automated ventilation is to provide the minimum assistance necessary to keep the patient healthy. Over-ventilation can cause permanent injuries including volutrauma and lung over-distension [49] and prevent the patient from developing their respiratory muscles in order to function independently. The majority of premature infants particularly those very and extremely preterm, require breathing assistance due to the incomplete formation of their airways and lungs.

Automated long term mechanical ventilation systems used in conjunction with surfactant therapy has been the preferred method of treatment since the 1960's when it was employed to treat infants with hyaline membrane disease [23]. These ventilators supercede a long history of resuscitation procedures and devices, with mouth to mouth and alcohol being used since the middle ages, to the first bellows type ventilator with pressure limiting valves being built by Hunter in 1755. In 1907, Draeger and Blume released the Pulmonator, a mobile short-term respirator that applied alternating positive-negative pressure [50]. All of these progressions to or from ventilation were made according to prominent scientific understanding at the time.

Modern mechanical ventilators provide oxygenation to the patient via silicon face mask, nasal prongs or insertation of an endotracheal tube (ETT). The ventilator machine itself contains a pump which delivers air from a tank of compressed oxygen or mixed gas, depending on the patient needs. The air is humidified via a humidifier and warmed to body temperature before delivery. Clinicians select appropriate modes of delivery via a touch screen computer interface. In addition to the mode, the user may select important parameters including breath frequency, delivered volume, PEEP and PIP. The interface also displays patient breath data and

waveforms.

Due to advancements in technology and understanding of neonatal physiology, mechanical ventilators save more lives than ever before [51]. Treatment modes include biphasic positive airway pressure (BiPAP), continuous positive airway pressure (CPAP), high flow nasal cannula therapy (HfNC), nasal high-frequency oscillation CPAP (hfCPAP), nasal intermittent positive pressure ventilation (NIPPV), synchronized intermittent positive airway pressure using fluidic flip flow driver (SIPAP), intermittent mandatory ventilation (IMV), synchronized intermittent mandatory ventilation (SIMV), synchronized intermittent positive pressure ventilation (SIPPV, also known as Patient Triggered Ventilation, PTV, or Assist Control (A/C)), high-frequency oscillation ventilation (HFOV), Volume Guarantee (VG) and pressure support ventilation (PSV) [12].

The most common mode of ventilation is IMV, especially in centres outside of the first world. During IMV, mechanical breaths of fixed duration and volume or pressure are delivered at predetermined time intervals. This leads to asynchrony because the frequency of spontaneous breaths by the infant is random. *Inspiratory asynchrony* occurs when a mechanical breath is delivered at the end of and extends beyond spontaneous inspiration. It can produce an inspiratory hold that limits the spontaneous respiratory rate or results in excessive lung inflation. Expiratory asynchrony occurs when a mechanical breath is delivered during exhalation. It can delay lung deflation and elicit active expiratory efforts against positive pressure producing large fluctuations in intrathoracic pressure. Asynchrony can affect gas exchange, and has been linked to increased risk of air leaks and intraventricular haemorrhage (IVH) [23].

The synchronized modes (SIPPV, SIMV, VG, PSV) use complex software control algorithms that allow ventilators to respond in real time to feedback from the infant has meant that patients pulmonary systems can grow without ventilator induced injuries, in particular, overdistension injuries. There substantial evidence of the benefit of synchronised ventilation over conventional ventilation. Patients can be removed from ventilation (extubated or mask removal) sooner which is associated with lower rates of sustained ventilator induced injury or disease [23, 49, 52]. Ventilators are required to deliver these treatment options to a wide patient weight range (approx. 500gms to 6Kgs for newborns, up to 35Kg infant in some combined paediatric models) and often in a difficult thermal environment (closed incubators, open care platforms with high humidity patient circuits) [12].

Today, as with throughout history, our technological abilities are ahead of our understanding of the intricate physiology of the infant's lungs. This is one of the primary reasons why the discovery of negative side effects of new delivery modes of ventilation are often not discovered until clinicians notice adverse trends in the health of their patients. The other primary reason is the dysfunction of these modes. Thankfully this is far easier to control for and avoid through rigorous regulatory testing before the modes are introduced into practice. Unfortunately these controls currently are not rigorous enough and lives are still at risk to avoidable injuries [12, 27, 15, 52], which will be explained in detail under *Shortfallings of Current Regulatory Standards*. It has been observed that with the recent update of the Fabian ventilator (Acutronic, Hirzel Switzerland), VG fails with a sudden drop in PIP on the smallest infants. Problems have also been observed in the VN500's (Drager, Lubeck Germany). This has meant that often VG can't be used which is not ideal. For this reason, VG mode is one of the first technologies that we intend to test after completion of this thesis, using the improved design for testing neonatal equipment. VG will be explored in further detail in the following section.

2.5.2 Respiratory mode of interest: Volume Guarantee

Volume Guarantee mode falls under the umbrella of Volume Targeted Ventilation (VTV). These modes aim to maintain tidal volume and thereby reduce injury causing issues that are common with Pressure Limited Ventilation (PLV). PLV works by delivering a fixed PIP and has traditionally been used to control the arterial carbon dioxide (PaCO2). During this mode, the tidal volume (V_{Td}) fluctuates due to the babies own attempts at breathing, changes in lung mechanics (eg, the addition of surfactant) and variable ETT leak. PLV has led to high inflation pressures or losses in alveolar volume due to insufficient ventilation. A systematic review of randomized controlled trials of VTV versus PLV found that VTV significantly reduces the rates of death, bronchopulmonary dysplasia (BPD), pneumothorax, hypocarbia and brain injuries [53]. A contributing factor to this is that VTV seems to be associated with faster weaning than other modes of ventilation [23].

The premise of VG is that the machine will monitor a patients own breathing efforts and deliver the required amount of air required such that the expired V_{Td} equals a set volume. Specifically, the peak inflation pressure (PIP) is adjusted according to whether the V_{Td} of the previous inflation met the set V_{Td}. Used in conjunction with SIPPV mode, VG supports spontaneous breaths [53]. It aims to ensure that the patient is ventilated to the set volume whether the patient does or does not make any effort to breathe, according to a set back-up rate frequency (BUR) which delivers inflations when the infant's breathing rate falls beneath it and auto-adjusts when

they make a breathing attempt [49]. The advantage of using expired V_{Td} is that it is less affected by ETT leak than inspired V_{Td}. This is important because it avoids erroneous changes to the delivered V_{Td}. VG can be used with a leak of up to 50%. Beyond this, the ventilator will increasingly underestimate the delivered V_{Td}, causing an increased PIP and potentially hypocarbia.

So far, there is little published evidence of any major concerns associated with the use of VG. This could be associated with how relatively new VG is and how it is still not used in many NICUs due to lack of understanding of it [12]. There is some published clinical evidence of some inappropriate PIP's being delivered with VG. McCallion et al (2005) found that after 3% of inflations, interruption of expiration (most likely due to diaphragmatic braking) caused an increased pressure for the next inflation of $\approx$ 4.9 cm H_2O compared with normal VG inflation [52]. In their 2007 study, Mccallion et al found that in the Draeger Babylog 8000 ventilator the PIP was 4cm H_2O lower during triggered inflations (those caused by the baby making a breathing effort) than untriggered inflations (those made according to the BUR) [49].

It is not yet known how common these errors are and why they particularly are seen in the smallest infants. To obtain data of VG interraction with a valid infant lung simulator would be very useful for clinical decisions and to induce technological changes and changes to the Standards.

2.5.3 rPAP System

The rPAP technology is posed as a replacement for the T-piece (TPR) which is the piece used to permit exhalation and help regulate flow when delivering CPAP and other ventilation therapies. During spontaneous breathing, the rPAP has a lower resistance to breathing, is pressure stable and reduces the imposed work of breathing compared with the TPR. During positive pressure ventilation, the rPAP has higher expiratory flows and shorter time constants. This probably reduces the risk of inadvertent PEEP, but it may also increase the risk of unequal ventilation in inhomogenous lungs [26]. In order to guarantee the rPAPs value over the TPR for sure and bring it to the point of clinical usage, it would benefit from obtaining data from being tested on a valid infant lung model .

2.6 Shortfallings of Current Regulatory Standards

In Australia, the Therapeutic Goods Administration (TGA) is the government body responsible for setting the regulatory guidelines of devices used in the medical in-

dustry. New medical devices must go through procedures to meet these guidelines and approved before being permitted for use with patients. This can often take many years. The six general principles that must me met are [13]:

- Use of medical devices do not to compromise health and safety.
- Design and construction of medical devices must conform with safety principles.
- Medical devices must be suitable for intended purpose.
- Long-term safety. That is, the device is safe to use within the period indicated by the manufacturer so long as it is maintained and no subjected to stressors beyond manufacturers recommendations.
- Medical devices are designed and packed in such a way that they are not adversely affected by transport or storage.
- The benefits of medical devices outweigh any undesirable effects.

The TGA, as well as the United State's equivalent body, the Federal Drug Authority (FDA) and the International Electrotechnical Commision (IEC) are informed by the International Organization for Standardization of the minimum requirements devices must meet. These requirements are decided by a panel of "technical experts" in a given field who design the standards for a device, or device category, based off clinical testing different brands devices. Automated neonatal ventilators fall under *ISO 80601-2-12:2011 Particular Requirements for basic safety and essential performance of critical care ventilators* [54]. Manufacturers and sponsors present medical devices to the TGA by application for market approval. The use of the standard is not mandatory, especially in cases where there is no relevant standard. In these cases, the TGA takes into consideration clinical evidence and a medical device risk category: Class I low risk, II, IIa, III and high risk active implantable medical devices (AIMD). It is also possible in cases where there is no clinical evidence available examining device efficacy, the device can be registered as low risk and therefore used on patients via expert opinion alone [12].

The TGAs lack of thoroughness, transparency and their apparent inadequacy to respond in a timely fashion to findings of major risks of devices are critiques made by Tracy and Hinder at Westmead NICU, as well as other clinicians in a recent presentation to the TGA. Furthermore, the pace of technological innovation in the field of neonatal ventilation, especially with regard to new complex algorithms and modes,

means that there possibly are unknown safety concerns that the TGA cannot account for under the current structure [12]. It is for this reason that one of the purposes of this thesis is to provide a foundation for gathering data to present to the TGA so that the process of any necessary legislation changes can begin.

A number of recent papers have identified consistent errors induced by automated ventilators that can cause severe damage to patients. Hence,the *ISO 80601-2-12:2011* is presently under review. Nonetheless, until the reviews completion, new devices on the market must only meet the present conditions. Some of the major published findings and observations that have lead to this review are:

- The scope of standards for critical care ventilators does not cover a number of delivery modes used with devices in neonatal intensive care units [12]. *ISO 80601-2-12:2011* Section 201.1.1 states [54], "This International Standard does not specify requirements for medical electrical equipment that is intended solely to augment the ventilation of spontaneously breathing patients within a professional healthcare facility", and, "This International Standard is not applicable to continuous positive airway pressure (CPAP), ventilatory support, transport ventilators, and high frequency oscillatory ventilators".

- Ventilators in CPAP mode regularly experience partial to complete expiratory limb occlusion due to water accumulation, causing dangerous pressurization and inappropriate alarms. They fail to display correct delivered airway pressure during simulated partial and total occlusion [15]. Therefore, most neonatal ventilators that have been permitted by the TGA for patient care fail to meet requirements for 'Obstruction Alarm Condition' (201.12.4.107) and 'Partial Occlusion Alarm Condition' (201.12.4.108) [12] in the *ISO 10651.4-2004*. This standard has since been withdrawn and it is expected that the appropriate revisions will be made in the revisions of *ISO 80601-2-12:2011*.

- The ISO standard presently does not advise testing criteria for patients that weigh up to 10kg with regard to minimum PEEP valve settings in TPR devices [55].

Given the shortcomings of the present Standards in the area of neonatal resuscitation, it is clear that independent studies in NICUs such as the one being undertaken in this thesis are essential to ensure the appropriate and safe therapy of some of our most fragile patients.

3 | Core Study One: Verification and Validation of a Neonatal Active Lung Model

3.1 Background

3.1.1 Motivation

Presently, the ASL 5000 Adult/Neonatal Breathing Simulator is considered by many to be the best bench-top model that is available for purchase. However, its validity is questionable. Some of the major concerns include:

- The scale of components does not permit high precision measurements of ventilator performance. The piston volume and some tube diameters are adult sized, which impacts the airflow dynamics. There is a far higher amount of dead space in the model than a real infant's lungs, which will cause some distortions when there are high pressure swings from gas compression [26].
- Volumetric flow is not calculated by an external pneumotachograph, but rather from piston movements and compressible volume. Therefore measurements of volumetric flow and pressure can not be considered accurate. Under conditions such as an internal leak, flow was turbulent, or gas had undergone significant expansion/contraction, then the calculated value would be significantly inaccurate [26].
- Though its motor is strong, its mass and internal friction cause it to respond too slowly at a high respiratory rate [26].
- The software is difficult to use [26].
- It can not replicate any lung compartment inhomogeneity [48].

In 2018 a new neonatal active lung model became available for purchase known as the GINA (Dr. Schaller Medizintechnik ™, Dresden, Germany). Clinicians at West-

mead Hospital, Australia and researchers at the Karolinska Instituet, Sweden, took interest in it. This is because it was designed by an expert in the field of neonatal respiration, Dr. Peter Schaller. It was the first model designed specifically and singularly to model the neonates airways. As it was so new and quite unknown, no papers or data had been published assessing its validity. In the interest of finding better ways of testing neonatal respiratory equipment, a GINA was purchased by the team at Westmead Hospital NICU with the first task of testing its validity.

3.1.2 GINA's design

Figure 3.1: The NALM box

The GINA is a hybrid mechanical-software model and an active simulator. It consists of two parts: the mechanical model which will be referred to here-out as the NALM (neonatal active lung model) box, and the GINA software. The NALM (fig. 3.1) is a single compartment model which uses a microchip to drive a linear voice coil motor (VCM) which modulates air movement in and out of a cylinder. Inspiration is active and expiration is passive, which is in line with with the majority of the literature. The following parameters are adjustable and cause physical changes in the machine:

Tube resistance is realised by endotracheal tubes (2mm, 2.5mm, 3mm and 5mm) which are manually switched by a dial on the NALM. The smaller 3 tubes are made of Vygon (a PVC composite regularly used as an ETT), the 5mm is made from silicon. These materials are known to have similar mechanical characteristics to endotracheal tissue.

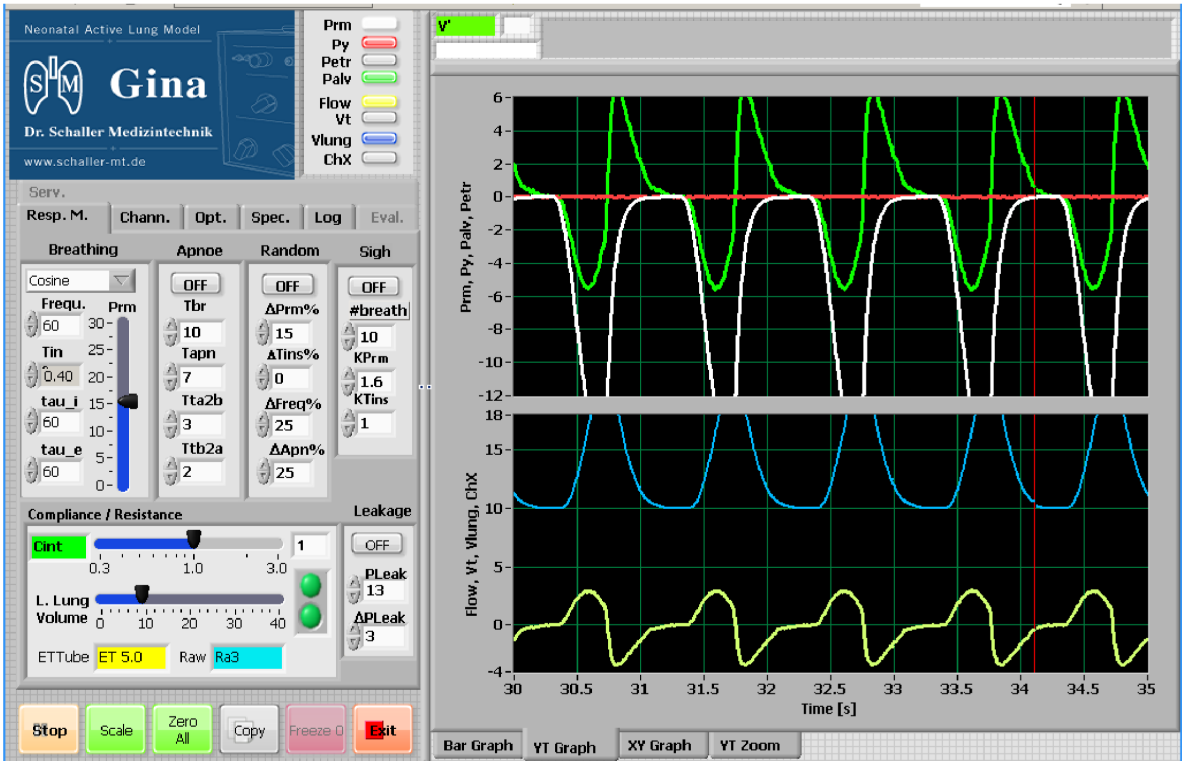


Figure 3.2: GINAs interface (full screen is trimmed)

Airway resistance (cmH2O) is realised through seven thin walled stainless steel tubes of differing internal diameter which are connected in parallel, which leads to dominating linear behaviour (ie. resistance is most affected by tube diameter and not by frictional resistance.) These are manually changed via a dial on the NALM.

Compliance can be realised internally or externally. The internal compliance (C_int) represents dynamic compliance. The value is set on the GINA in the range of 0.3 to 3 ml/hPa, and the user may select whether it is modelled linearly or sigmoidally. This setting causes a proportional counterforce to be applied to the piston cylinder via a voltage difference. External compliance is realised by attaching one of two sizes of isothermal lung model (wide neck flasks filled with steel wool). External compliance represents non dynamic compliance.

Pressure of the respiratory muscles (Prm) is set by the user on GINA to simulate the spontaneous breathing effort of the patient. This parameter adjusts the start and end points of the piston and controls volume output.

Leak is realised through a luer-lock valve which can be triggered to open and close at certain pressures, set by the user on GINA.

Breath frequency and inspiratory and expiratory time constants are set in GINA are alter the rate of piston movement [56].

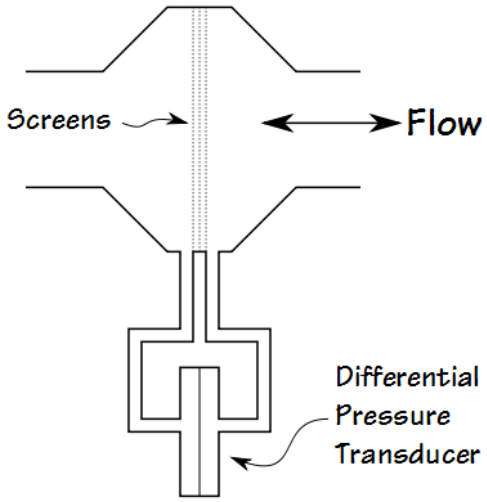


Figure 3.3: A pneumotachograph determines flow rate by calculating the pressure differential across a screen of known resistance [57]

The GINA software is a LabView based program (fig. 3.2). In addition to controlling the parameters mentioned above, the GINA interface displays real-time signals of parameters including:

Pressure at the (i) y piece outlet (P_y), (ii) endotrachea (P_{etr}) , and (iii) alveoli (P_a). These pressures are each measured by the differential pressure transducer part of a pneumotachograph.

Volumetric flow rate (f) which is measured by the pneumotachograph (fig. 3.3) at the breath outlet.

Tidal volume (V_t) which is calculated according to the integral of flow.

Lung volume (V_p) which is calculated as the volume of air moved by the piston plus cylinder dead space.

Pressure of the respiratory muscles (Prm) which is calculated as the force required to move the piston.

Channel X which is a function which can plot many parameters including work of breathing or the inspiration step function.

The user also has control over breathing frequency ($freq$), inspiratory time (t_{in}), inspiratory time constant (τ_{in}) and expiratory time constan (τ_{exp}). GINA has built in functions to simulate apnea, sigh, or random breathing. GINA has the option to record and output data at a sample rate of the users choice. The continuous data for each of the signal parameters listed above is output as a TDMS file.

3.2 Aims

The aims of this series of experiments are as follows:

1. The primary aim of this study is to determine whether the GINA/NALM is a valid active mechanical model of a neonatal lung. Specifically, there will be two criterion for assessing model validity. These are (1) Accuracy of measurement, and (2) Realistic imitation of neonatal respiratory behaviour.

2. The secondary aim is to evaluate the model for its usability as a bench-top model for testing ventilation equipment. This aim is not experimentally assessed but is discussed in the discussion section.

3.3 Methodology for aim 1: Assessing GINA measurement accuracy

3.3.1 Experimental Design and Outcomes

The physical, non-calculated measurements of GINA are take via pneumotachograph. These are volumetric flow f_v and pressure P. Therefore, these were the parameters that were chosen for accuracy testing. For the pressure recording, outlet pressure P_y was chosen because: (i) it can be tested without taking apart the NALM, (ii) the pneumotachographs in NALM are all the same kind [56], and, (iii) it is used far more frequently in equipment testing than endotracheal pressure and alveolar pressure.

The FlowAnalyzer™(imtmedical, Buchs (SG). Switzerland) device was chosen as the benchmark measurement device, as it is capable of measuring low flow with high accuracy [58]. The experimental synopsis was to hermetically connect the FlowAnalyzer and GINA and have each device simultaneously data record while GINA output simultaneous breaths. The data sets were then compared using a purpose built software, the *GINA Validator* which determined the accuracy of the GINA data. The experimental apparatus and software design are described in the following sections.

3.3.1.1 Experimental apparatus

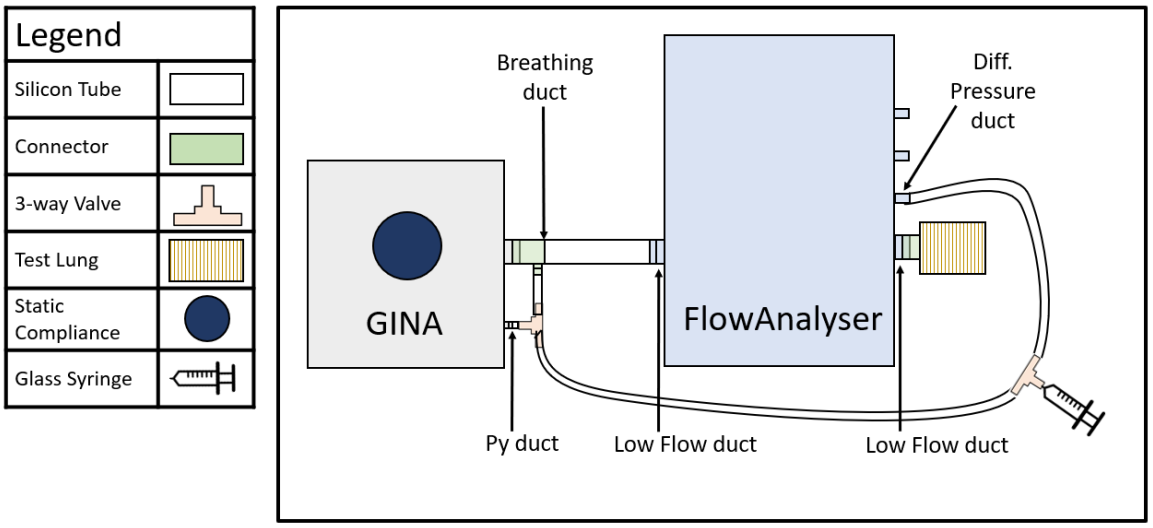
Before conducting the experiment, all measures were taken to minimise systemic error. The following protocol was adhered to every time.

1. Ensure that the NALM-to-computer and FlowAnalyser-to-computer USB connections are set up.
2. Calibrate each device according to the procedures described in their manuals.
   (a) NALM calibration required a specific set up in which a small pressure (30-50hPa) was manually delivered to it's pressure outlet. We (myself and Murray Hinder) conducted this using a small glass syringe. We determined the pressure delivered with the FlowAnalyzer. The apparatus is shown in fig. 3.4a.
3. Set up apparatus according to fig. 3.4b. Ensure seals are as airtight as possible, the system is closed using a test lung, and that the silicon tubes between the NALM breath outlet cut to the minimum possible length and is not kinked.
4. Switch on both devices. Set GINA to breathe for 10 minutes to ensure devices are warm.
5. Check that the sample rate on each machine is set to 200Hz.

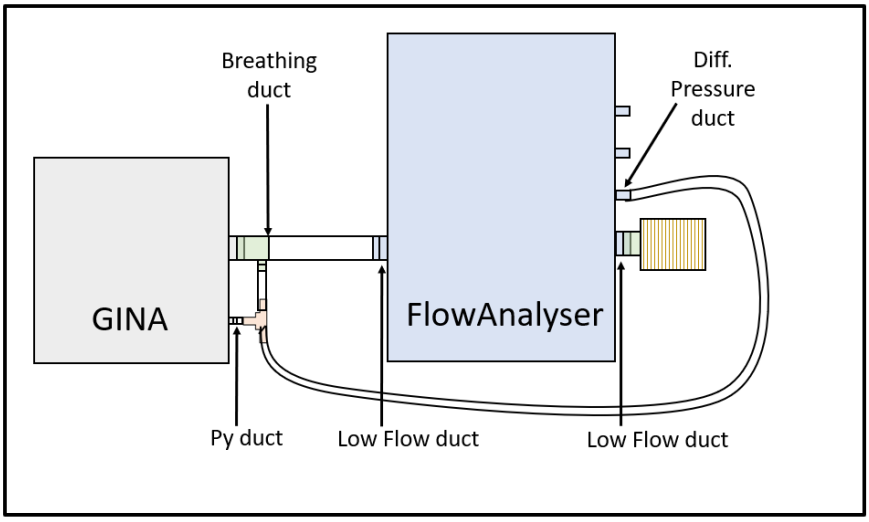
3.3.1.2 Assessing the accuracy of GINA

Following this process, the experimental breathing parameters were set on GINA. The independent variables (tested separately) were Prm (20, 35 N) and $freq$ (60, 80 breaths per minute). Values were selected to represent the spectrum of breathing. Constant values were $C_{int} = 1.5$, $Prm = 20$, $freq = 60$, $\tau_{in} = 60$, $\tau_{exp} = 60$ and $t_{in} = (0.4, 0.3s)$ for the respective frequencies. The endotracheal tube was set at 2.5mm diameter and airway resistance was set to 3 cm H2O. For each setting profile, 2 tests were run. Therefore there was a total of 10 tests. Before the start of each test, we checked that the test lung did not fully inflate at the test settings. This was done to avoid pressure build up in the system that could cause erronous results. Then, one minute of simultaneous continuous data logging on each machine was commenced.

The GINA output file was converted from TDMS format to excel format using a simple excel plugin. Then, the output data was assessed using the *Gina Validator* software which uses Bland-Altman (BA) analysis to compare signal profiles and



(a)



(b)

Figure 3.4: Aparatus for GINA validation. (a) Calibration set-up, (b) Aparatus for testing

determine whether the signals are significantly different. Each machine has an error allowance of 0.05 for each parameter. The combined acceptable error was 0.1 (mbar, l/min). A non-significant difference was determined as when the linear regression of the differences did not intercept the 95% confidence intervals, and the mean of differences value was within 0.1(mbar for pressure, l/min for flow) of zero. In addition to this measure of validity, I decided to run a further BA analysis in SPSS comparing the peaks and troughs of the flow and pressure signals between the GINA and FlowAnalyser to determine whether the mean of differences was within the the acceptable error of 0.1 (mbar, l/min). This test was performed on one case only ($freq = 80, Prm = 20$).

3.3.2 Design of GINA Validator software

The *GINA Validator* software was written in Matlab code. This language was selected for its speed with data processing and its usability, as the software may need to be amended to in the future by others and many clinicians and engineers are well versed in Matlab. An object-oriented code style was selected as it used less RAM than spaghetti code, is less corruptible, is easily read and can have functions added in a straightforward way.

The software design attempted to meet the Software Sustainability Institute's software evaluation criterion [59]. This criterion includes *usability*, which encompasses understandability, comprehensive and appropriate documentation, installability and learnability. It also includes *sustainability and maintainability* which includes testability, evolvability and changeability. Some *sustainability and maintainability* criterion such as governance and licencing are presently beyond the scope of this stage of development.

GINA Validator has a simple graphical user interface (GUI) which the user interacts with (see fig. 3.5). This was designed using Matlab's GUIDE feature. The software design is divided into the following three sections: User interface, Error handling, and Code flow and function design.

3.3.2.1 User interface

The user interface was designed to be easy to use. Refer to figure 3.5 for the following. The user is able to browse for and select the input files from GINA and FlowAnalyzer and write in the output file name. Selecting 'Go' runs the data analysis. Depending on the validity test (the BA analysis previously described), the message "Validation passed" or "Validation failed" will fill the box on the app. Selecting 'Re-

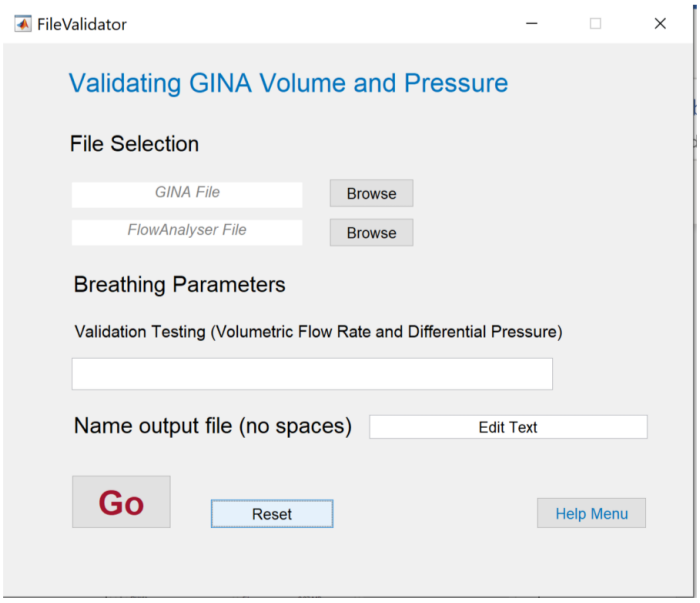


Figure 3.5: Gina Validator user interface

set' will restore the apps original settings. Selecting 'Help Menu' will cause the pdf instruction manual to be opened. See appendix A.2 for the instruction manual.

3.3.2.2 Error Handling

A necessary component of any software is that it can catch errors so that the progam doesn't crash or get caught in an infinite loop, wasting RAM. Once an error is caught it should be handled. This means the user should be informed of what the issue is and prompted to submit any needed information. The program will then either continue with the provided information or stop and reset. In any case, the program should tell the user exactly what is happening. Errors that are caught in this way should only be those that occur as a result of aberrant input, not bugs. Figure 3.6 shows the flow process of how errors are caught including the problem, the prompts/error messages, response handling, and final action of the software

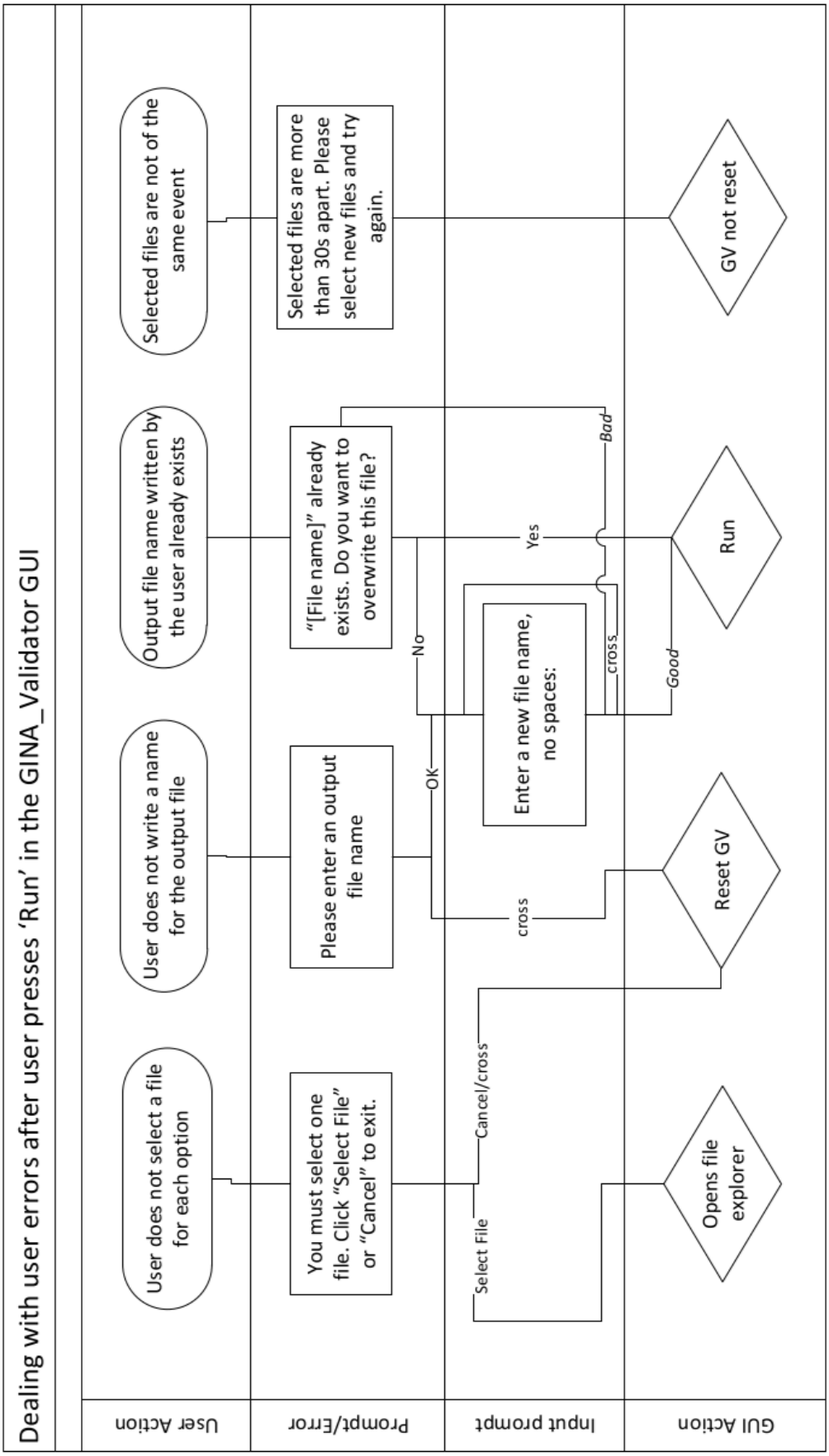


Figure 3.6: Handling errors in GINA Validator software

3.3.2.3 Code flow and function design

There are three primary matlab files: *FileValidator*, *main* and *Tester* . The full code for the first files is contained in appendix A.1. This is the code flow: *FileValidator* collects user input, *main* tests that the files are of the same event and then transforms data into tables for analysis, and *Tester* is fed this data and analyses it to determine validity.

- FileValidator.m creates the GUI application. The functions code the layout and the capabilities of each item (button, text box etc) on the interface. For example, the browse buttons opens the file browser into the directory named $Tests_{Data}$, a subdirectory of the software directory and allows the user to select files of the relevant file time (.xlsx for GINA, .log for FlowAnalyzer) Data is collected, tested for errors (see section *Error Handling* and sent to the main.m file as global variables.
- main.m imports the files, tests that the signals are of the same event and then transforms the data into tables for analysis.
- Tester.m runs data analysis to determine signal validity and saves appropriate data and plots into a directory named according to provided by the user to the GUI. This directory is saved under the $Test_{Output}$ directory.

This is the code flow and function design of *Tester*. Refer to the full code in appendix A.1

function: obj = Tester

Code line reference 53-167

Input datasets from GINA and FlowAnalyser, output folder name, signal time stamps.

Description This is the object initialization function. The GINA output xlsx file is read in as a table. The output file folder is made in the Test_Results subdirectory of the software directory. Input parameters are set as object properties. All other functions are called.

function: filetrimmer

Code line reference 415-494

Input object properties

Description This function trims the data to start from the same moment. This is done twice. First, by comparing time stamps. Secondly, by trimming both datasets to the first moment of monotonically increasing flow where the flow wave transitions from negative or zero to positive (see fig. 3.7b).

function delayCalc

Code line reference 493-539

Input object properties

Description This function calculates the linear regression of the peak-to-peak delay between time signals to calculate the scaling factor (see fig. 3.7a.

Points of interest It was interesting to discover that there was a delay between signals that could not be attributed to phase shift. The delay was such that the signals appeared to be scaled differently in the time domain (see fig. 3.8. Therefore, this could not be due to the apparatus set up. Instead, it had to be because of a sample-rate error, meaning that in one or both machines there was a delay between signal trigger and signal recording.

function: labDataScaler

Code line reference 407-413

Input object properties

Description This function multiplies the time domain of the FlowAnalyser data by the scaling factor to rectify the difference.

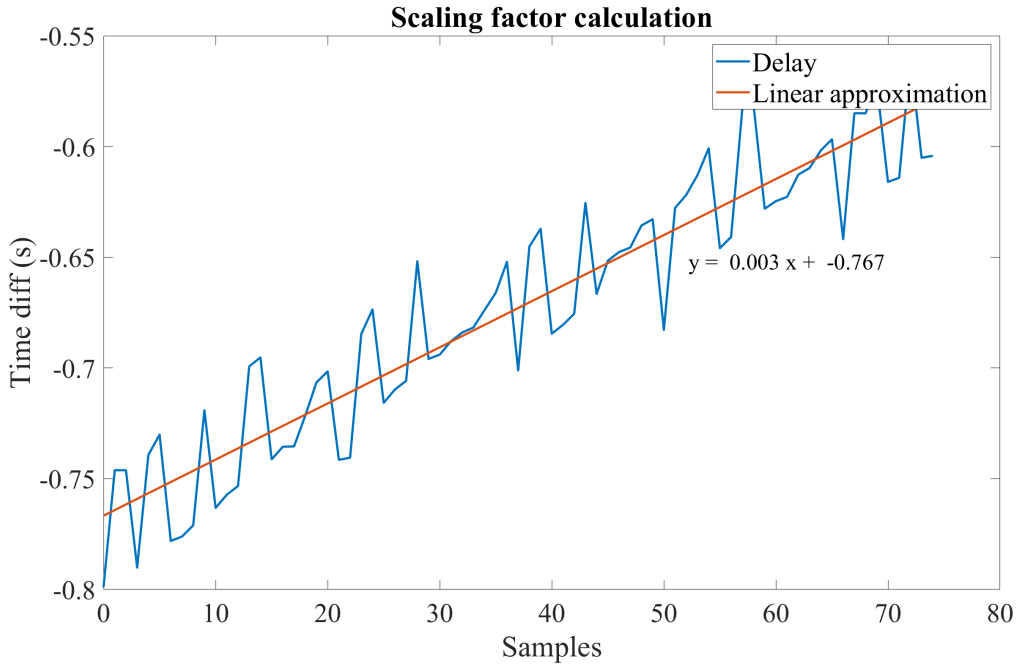
Points of interest

function: flowPlotter

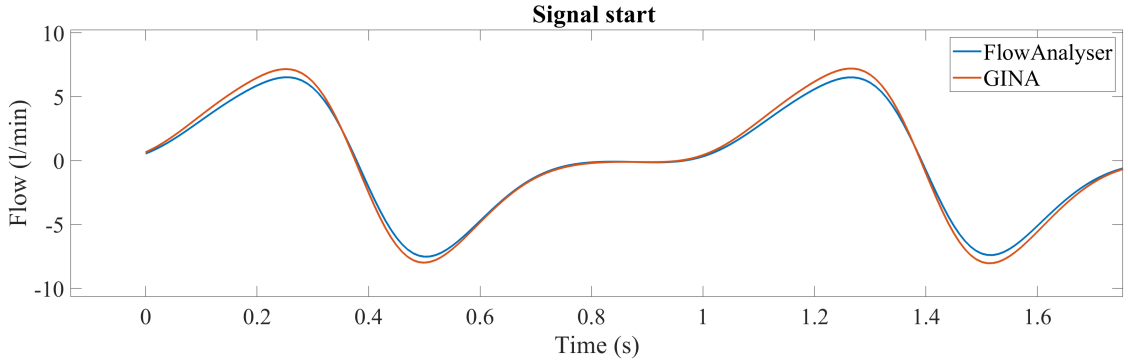
Code line reference 179-343

Input object properties

Description This function uses interpolates the flow signals for both devices and plots the signals against each other. BA analysis is run.



(a)



(b)

Figure 3.7: Data amendment processes in the GINA Validator. (a) There is a signal processing lag between the GINA and the FA. The slope of the peak-to-peak differences used to scale the FA signal and correct for sample rate error (b) Data is trimmed to the exact same start signal according to time stamp, phase shift, and then matching the first positive monotonically increasing slopes.

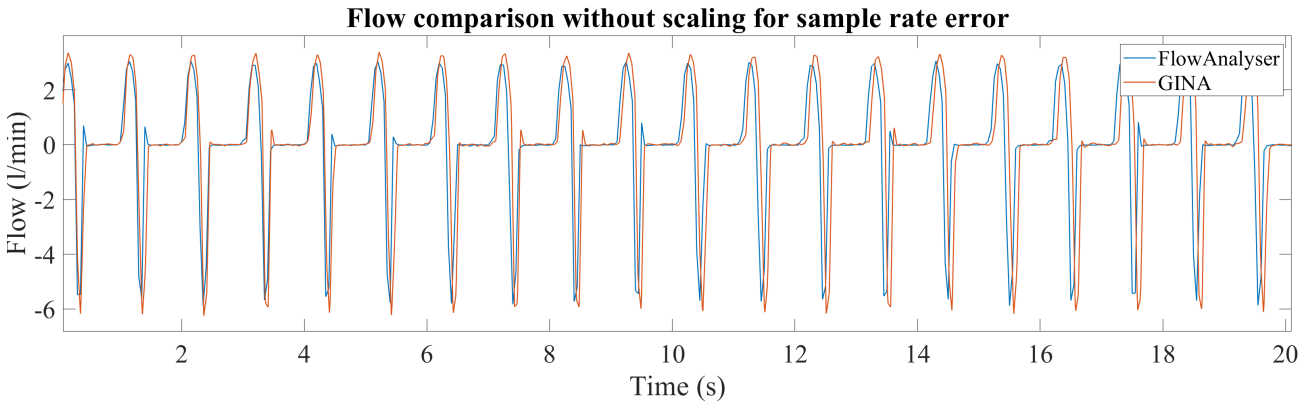


Figure 3.8: Unscaled flow data from the GINA and FlowAnalyser

function: pressurePlotter

Code line reference 259-345

Input object properties

Description This function uses interpolates the flow signals for both devices and plots the signals against each other. BA analysis is run. Depending on the whether the regression of differences intercepts the 95$ confidence intervals, the parameter 'valid' is set to 1 (valid) or 0 (invalid). This determines the message validation message that will be displayed by the application.

3.3.2.4 GINA Validator software assessment criterion

The two test criterion for assessing the GINA Validator software are:

1. It is valid, that is, it does what it's meant to do. Specifically, it should determine whether GINA data is valid and alert the user, and save figures and analysis to the chosen directory.

2. It passes the conditions of the Software Sustainability Institute's criterion based assessment [59] that are relevant at this pre-distribution stage of development.

3.4 Methodology for aim 2: Assessing realistic behaviour

3.4.1 Experimental design and outcomes

The assessment criteria for realistic behaviour were that:

- Breathing waveforms maintained correct morphology across the measurement spectrum for independent variables ($f_v, freq, Prm$), see fig. 3.9 for example of good morphology.
- Interactions could occur between the GINA and external devices and produce realistic outcomes.
- Changes in internal compliance were proportional to changes in work of breathing.

Details of the former two assessments are described:

3.4.2 Assessing breathing waveforms

Realistic breath flow waves do not have any flat spots. A flat spot would indicate no change in flow, which in a dynamic system would never happen. As flow is dependent on volume changes, we determined that the simplest way of finding any flat spots was by looking for non-linearity in the T_p vs. T_v curve, and strange morphology in P_a vs. T_v curves. CPAP was applied to the GINA while GINA was spontaneously breathing. Pressure levels were 2.5 mbar, 5 mbar, 7.5 mbar, 10 mbar and 12.5 mbar. GINAs endotracheal tube was set at 5mm, Prm was 5 and Airway resistance was 3 cmH2O.

3.4.3 Assessing interactions

Interactions were assessed in two simple ways. The first was by attaching a ventilation machine (Babylog VN500, Draeger Medical, Lubeck, Germany). GINA was set to not breathe. We visually assessing whether the flow wave display on the ventilator matched GINA. The second was by delivering CPAP to the GINA and testing whether GINA could maintain a PEEP.

3.5 Results

3.5.1 Assessment of the measurement accuracy of GINA

Visual comparison of the flow and pressure signals from GINA and FlowAnalyser show that they are closely matched especially in terms of frequency and morphology (see fig. 3.10, though there was a trend of the GINA signal having a greater amplitude. Using the BA assessment, it was determined that GINAs output for volumetric flow rate and for pressure was valid across all 10 test cases. In all cases ($freq = 40, Prm = 20; freq = 60, Prm = 20; freq = 80, Prm = 20; freq = 60, Prm = 5; freq = 60, Prm = 35$) the absolute average value of the mean of differences for

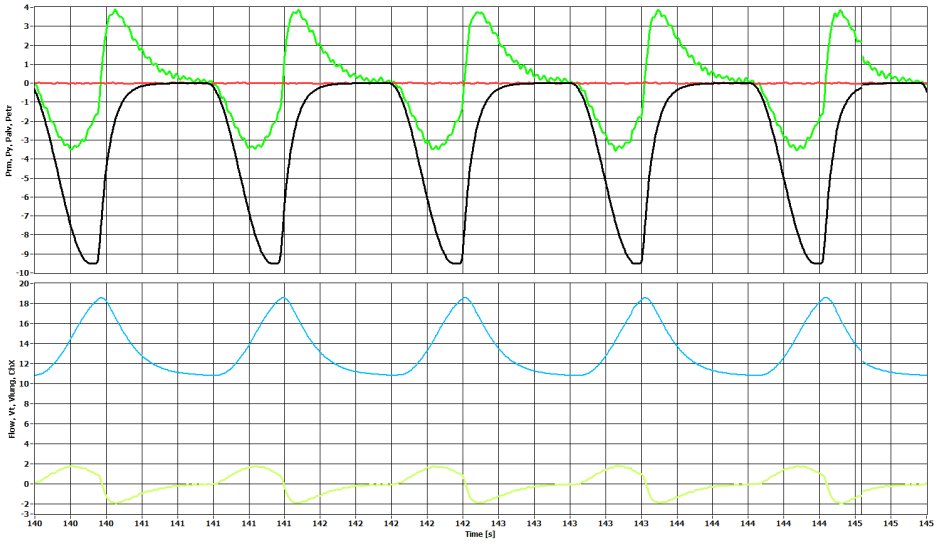


Figure 3.9: Graphical output from GINA of spontaneous breathing with good morphology. There is no test lung or device attached to the breathing outlet. Compliance is internal, linear. Signals from top to bottom: (i, green) Alveolar pressure, (ii, red) Pressure at the y piece, (iii, black) Pressure of the respiratory muscles, (iv, blue) Tidal volume, (v, yellow) Flow.

flow (-0.01, -0.02, -0.02, -0.02, 0, -0.03, -0.03, -0.04 , respectively) and for pressure (-0.01, -0.01, -0.03, -0.02, 0.01, respectively) were not greater than acceptable error of 0.1 (l/min, mbar). In all cases, the linear regression of signal differences did not intercept the 90% confidence intervals. Linear regressions consistently had a negative gradient. There was a trend towards increased variance of differences during expiration. See fig. 3.11 for analysis samples. Therefore, GINA meets the validity criterion for measurement accuracy.

There were some trends in the distribution of data in the BA plots. For flow, it appeared that increasing breathing $freq$ increased the variance, but increasing Prm decreased the variance. Regression lines were similar (y=-0.09x - 0.2 for $freq = 60$ $Prm = 20$, y=-0.08x - 0.03 for $freq = 60$ $Prm = 35$, y=-0.08x - 0.02 for $freq = 80$ $Prm = 20$). For pressure, both $freq$ and Prm increased the variance. Regression lines were identical for all test cases (y= -0.01x -0.05).
The BA analysis on the maximums and minimums for flow and pressure (see fig. 3.12) showed that for flow, the mean of differences was -0.31 for maximums and 0.25 for minimums. These cases fell outside of the tight error allowance (± 0.10). This trend confirms that GINA recorded a larger amplitude for flow. There is a greater spread of data for flow minimums with many falling above the upper leniency limit. For pressure, the BA analysis shows a mean of differences of -0.001 for maximums and 0.048 for minimums. These both fall close to zero. However, there is quite a spread of differences in both cases, with some points falling outside

of the upper and lower leniency limits for pressure maximums.

3.5.1.1 Assessment of GINA Validator software

The GINA Validator software is mostly valid as it does what is required of it, but some bugs remain. It performs analysis to determine validity of the GINA signals vs. FlowAnalyser signals. Data is saved in the subdirectory named by the user. The GINA Validator handles most errors, though at this stage not all user errors have been accounted for and therefore some bugs remain. Given that GINA Validator was not designed to be used outside of this experiment, fixing these remaining bugs was a low priority.

When it comes to the criterion based assessment, GINA Validator passes some but not all conditions. Under the *usability* criterion, the software passed understandability, documentation and learnability as the application interface is simple and easy to understand, and clear documentation of how to implement the software is included. Under the *Sustainability and maintainability* criterion, the software passes changeability, testability and evolveability, but not portability. This is because when the software is installed on different computers, although it runs, the app interface features (buttons, words etc) do not properly oriented. This is a problem with the GUIDE Matlab plug-in.

3.5.2 Assessment of how realistic GINA's behaviour is

In the first experiments of CPAP on the GINA to asses signal morphology, changing PEEP levels with CPAP caused the morphology of the pressure-volume curves to change. Curves shrunk as PEEP increased. Volume (tidal) vs. Volume (piston) curves were not linear as expected and peak volume being chopped at the highest and lowest PEEPs. This experiment was run a second time after it was discovered that the GINA software had corrupted on the first computer we were using. The second experiments did not show any aberrant morphologies. It was noted that approximately once every 10 minutes, morphology would be interrupted as a result of what we assume was the piston getting stuck and then hitting the back of the cylinder. This caused large spikes in the data (see fig. 3.14).

Visual analysis of the signal of the ventilator and the GINA when attached indicated that the signals appeared to match. GINA was able to maintain a PEEP when attached to the CPAP machine. The spontaneous breathing pressure (p_y) signal shifted upwards according to the applied constant pressure. It was observed that over the course of 10 minutes the pressure shift slightly drifted downwards.

Compliance was deemed as being effective by visualising the change in WOB as with compliance changes. As compliance increased, the peak of the work of breathing function did also.

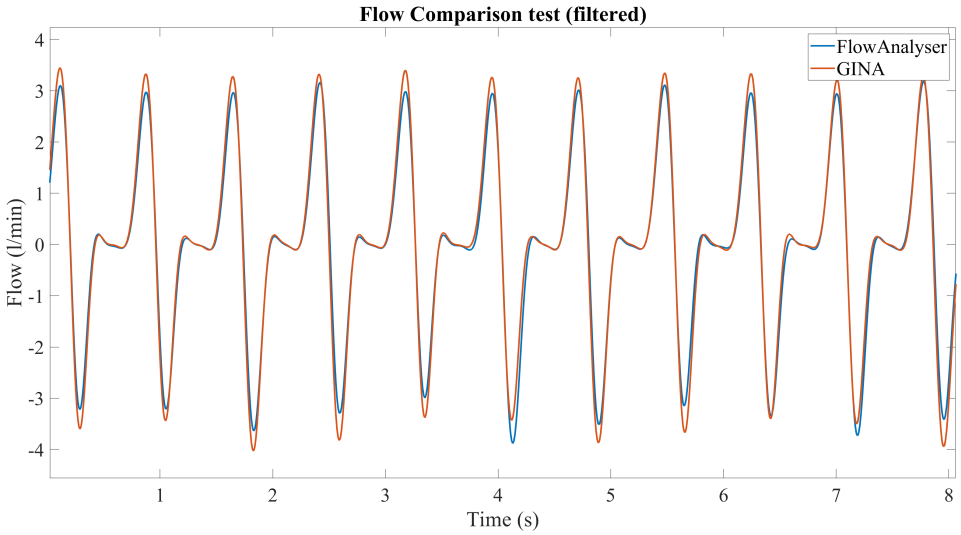
3.6 Discussion

Experimental tests have been carried out in order to determine if the GINA (NALM) is a valid model of a neonatal lung which can be used to test respiratory equipment. In the first set of experiments, we assessed the accuracy of GINA's measurements of volumetric flow and pressure at the outlet by comparing recordings from GINA and the FlowAnalyser of the same respiratory events. This was done using a purpose built software called the *GINA Validator*. We considered the FlowAnalyser to represent the standard of accuracy. Bland-Altman analysis was used to compare signals. The standard that needed to be met in order for GINA's data output to be considered accurate was that the linear regression of the difference did not intercept the 95% confidence intervals, and that the mean difference did not exceed ± 0.10. Additionally, the signals should be in phase and have the same frequency and overall shape. In the second set of tests, we assessed GINA's ability to mimic realistic behaviour. The criterion for validity included breathing signals were naturally shaped, that devices could interact with it and induce relevant changes, and that internal compliance was proportional to WOB.

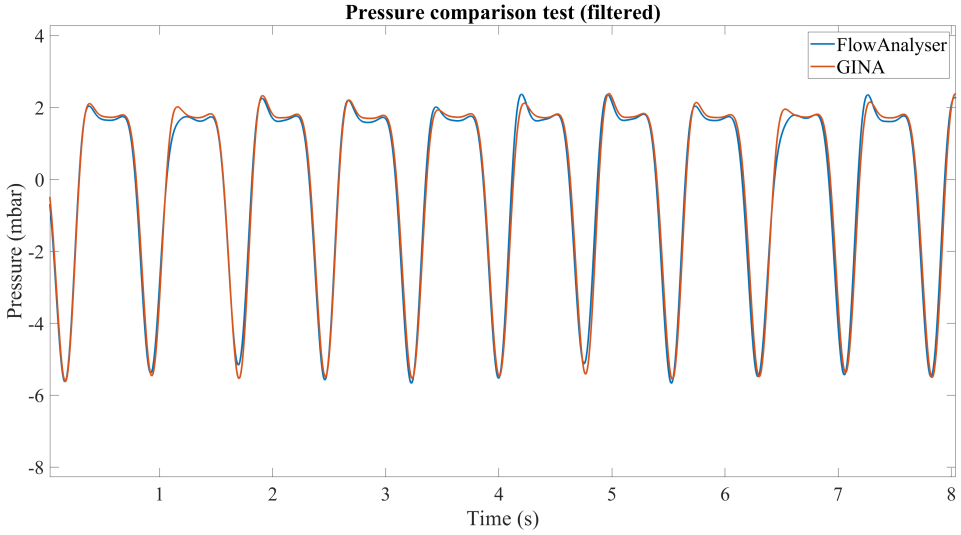
For clarity, the following discussion is divided into four parts. The first two sections concern the experiments described, each presenting an analysis of the results and an acknowledgement of experimental limitations. The third section regards the GINA Validator software including its usability, limitations and future development. The final section is a general discussion of the usefulness of GINA as a testing tool.

3.6.1 Analysis of the accuracy of GINA

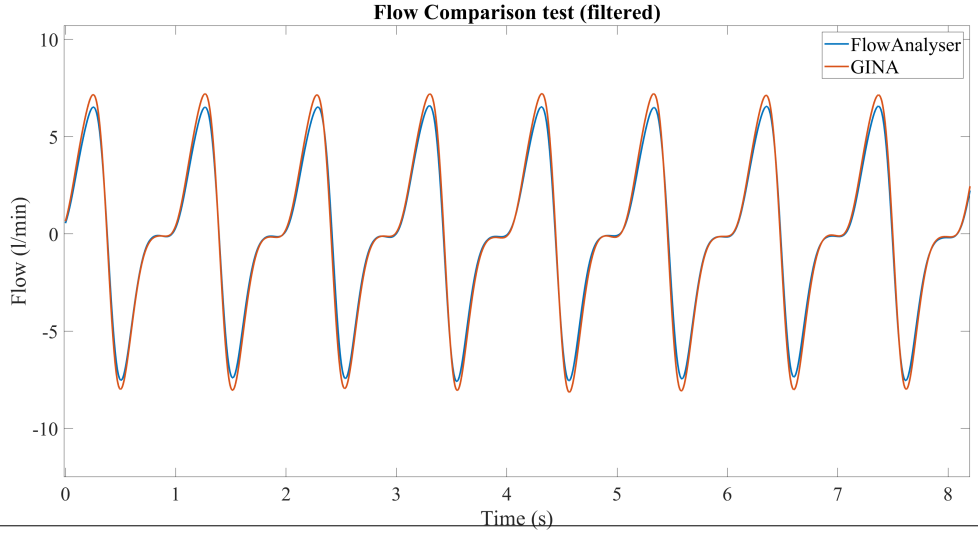
When we look at the flow and pressure signals and the BA analysis which comparing the whole flow and pressure signals, we see that GINA passes accuracy testing according to the criterion set. The time domain signals have very good synchrony; frequency is the same, turning points are aligned, and the general shapes are very similar. There are, however, some unexpected trends which will be discussed below.



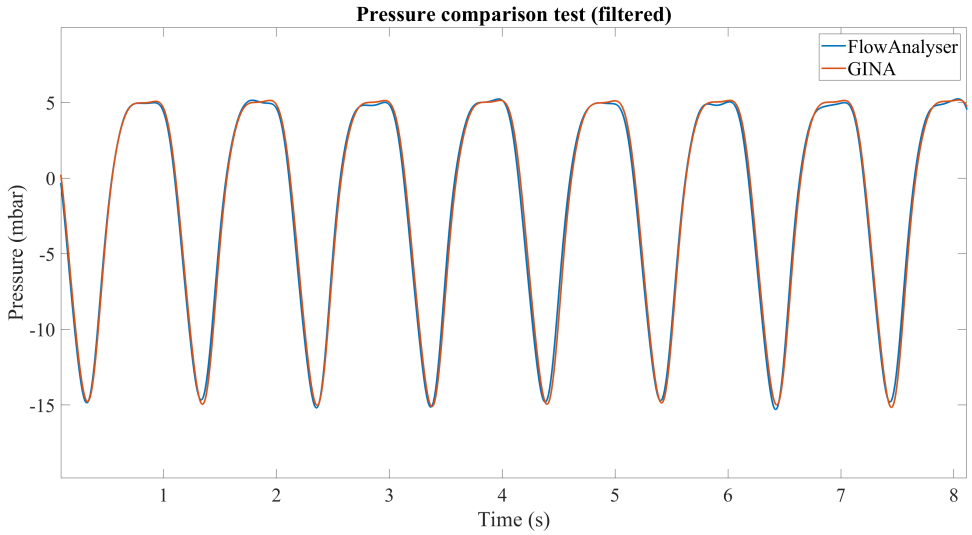
(a) Comparison of flow signals on spontaneous breath data with parameters: freq=80, Prm=20



(b) Comparison of flow signals on spontaneous breath data with parameters: freq=80, Prm=20



(c) Comparison of flow signals on spontaneous breath data with parameters: freq=60, Prm=35

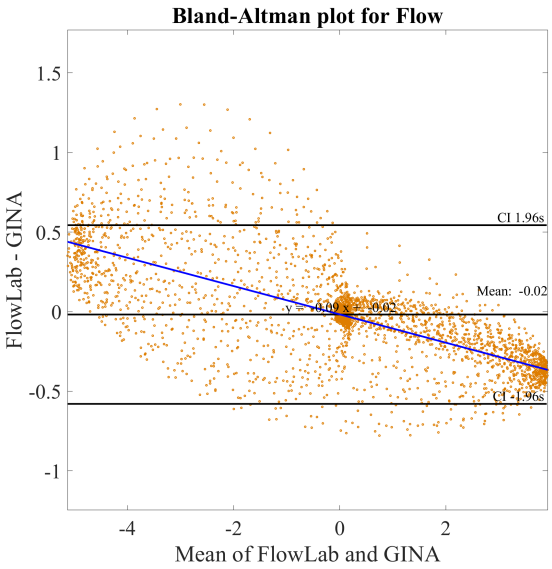


(d) Comparison of flow signals on spontaneous breath data with parameters:freq=60, Prm=35

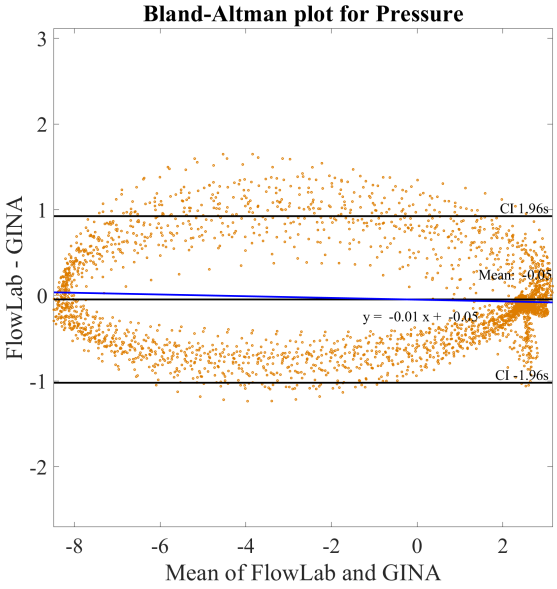
Figure 3.10: Flow and pressure signal comparison between GINA and FlowAnalyser

The GINA flow signal tends to have a larger amplitude that the FlowAnalyser signal: peaks are higher, troughs are lower. This trend causes the linear regression on the BA plots to have a negative slope. We can see that there is an accumulation of data points at the mean difference boundaries and around zero. Variance appears to be maximised at around the half way marks between these points of interest. The cause of this bow-tie shape becomes obvious when one considers the nature of the experimental apparatus. Because air is being pushed out of the GINA and into the FlowAnalyser then being sucked back in, there will be a slight lag in the rate of flow change in the FlowAnalyser compared with the GINA. The dynamic nature of fast flowing air means that it is unlikely that the lag profile will be the same every time. This is why when the rate of flow change is highest, there are the largest variances. On the other hand, the difference between signals at the turning points should always be similar. The turning points represent the maximum flow, minimum flow and the change of flow direction. Given that no variables were changed in each test, it is expected that these features will always remain similar with only slight variation due to noise from vibration, electrical equipment and any eddies in the flow at the boundaries between tubes.

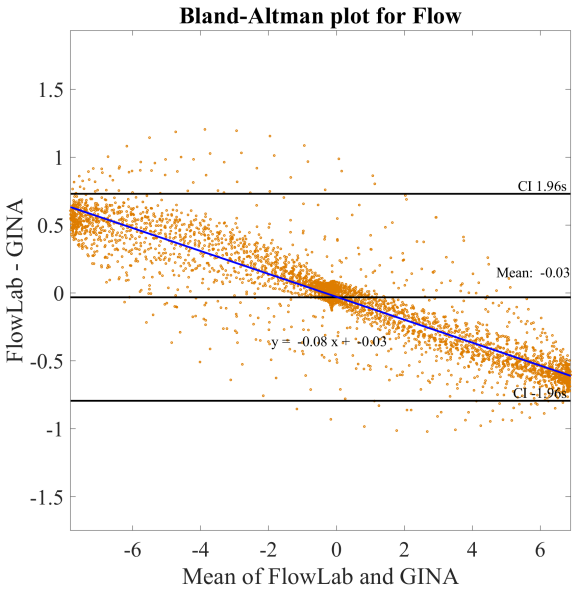
The GINA pressure signal tends to have a similar amplitude to the FlowAnalyser signal. This is to be expected because our apparatus is a closed system. There does appear to a small amount of leak which is indicated by the slight downward drift of



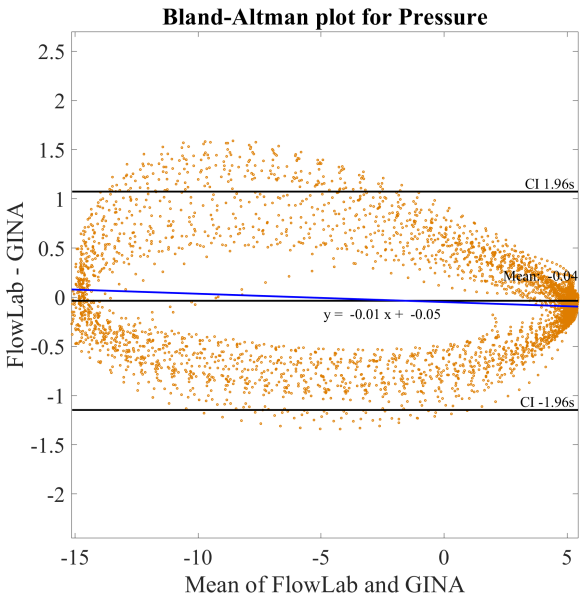
(a) BA test on spontaneous breath data of flow with parameters: freq=60, Prm=20. Mean = -0.02. Regression: y=-0.09 - 0.02



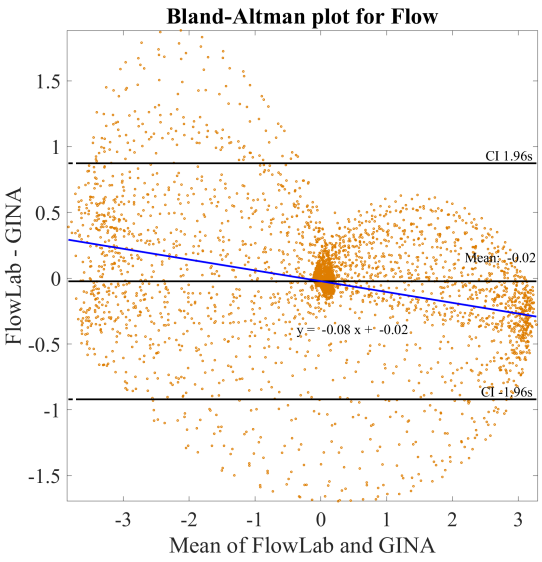
(b) BA test on spontaneous breath data of pressure at the y piece with parameters: freq=60, Prm=20. Mean = -0.05. Regression: y=-0.01 - 0.05



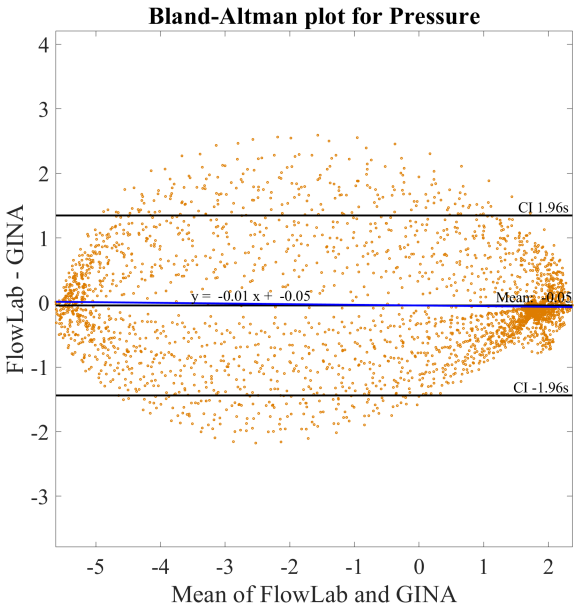
(c) BA test on spontaneous breath data of flow with parameters: freq=60, Prm=35. Mean = -0.03. Regression: y=-0.08 - 0.03



(d) BA test on spontaneous breath data of pressure at the y piece with parameters: freq=60, Prm=35. Mean = -0.04. Regression: y=-0.01 - 0.04



(e) BA test on spontaneous breath data of flow with parameters: freq=80, Prm=20. Mean = -0.03. Regression: y=-0.08 - 0.02

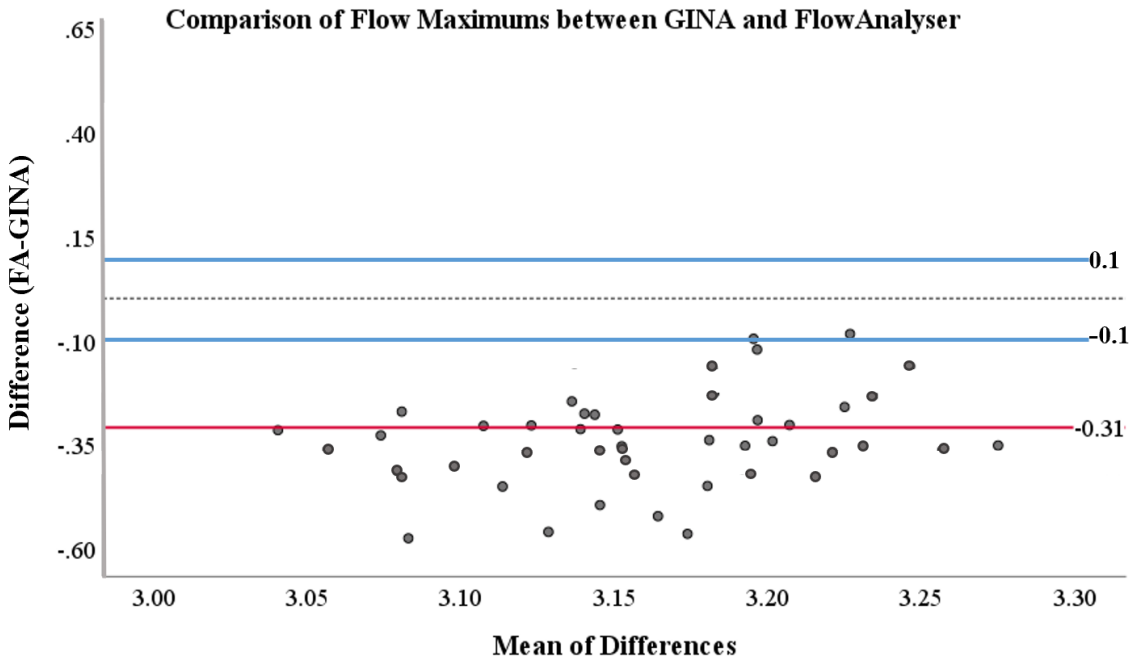


(f) BA test on spontaneous breath data of pressure at the y piece with parameters: freq=80, Prm=20. Mean = -0.05. Regression: y=-0.01 - 0.05

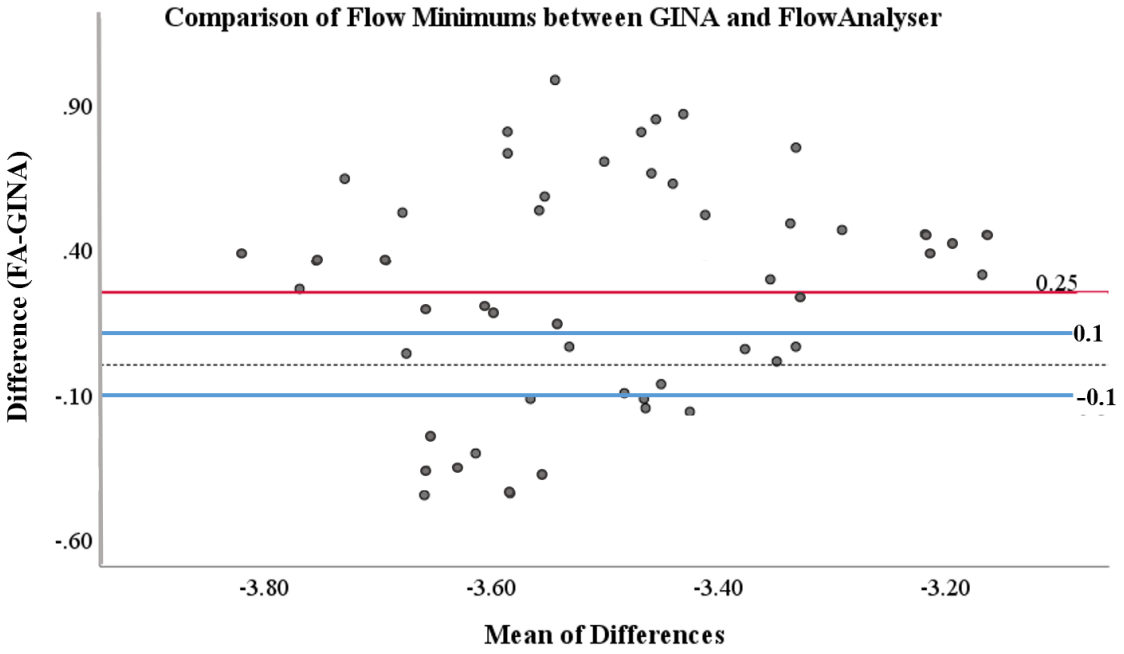
Figure 3.11: BA analysis of signal differences for flow and pressure

both pressure signals over time. This is likely as a result of the pipe connections in the apparatus, though it could be because of an internal leak in GINA. This will be discussed more below. In the BA plots, the data points accumulate at the pressure maximums and minimums and the variance is highest in between. This loop shape appears to have hysteresis. This makes sense because GINA is designed to be compliant and the silicon connection tubes also have some elasticity. Similar to with flow, there will also be some lag in the rate of pressure change in the FlowAnalyser compared with the GINA.

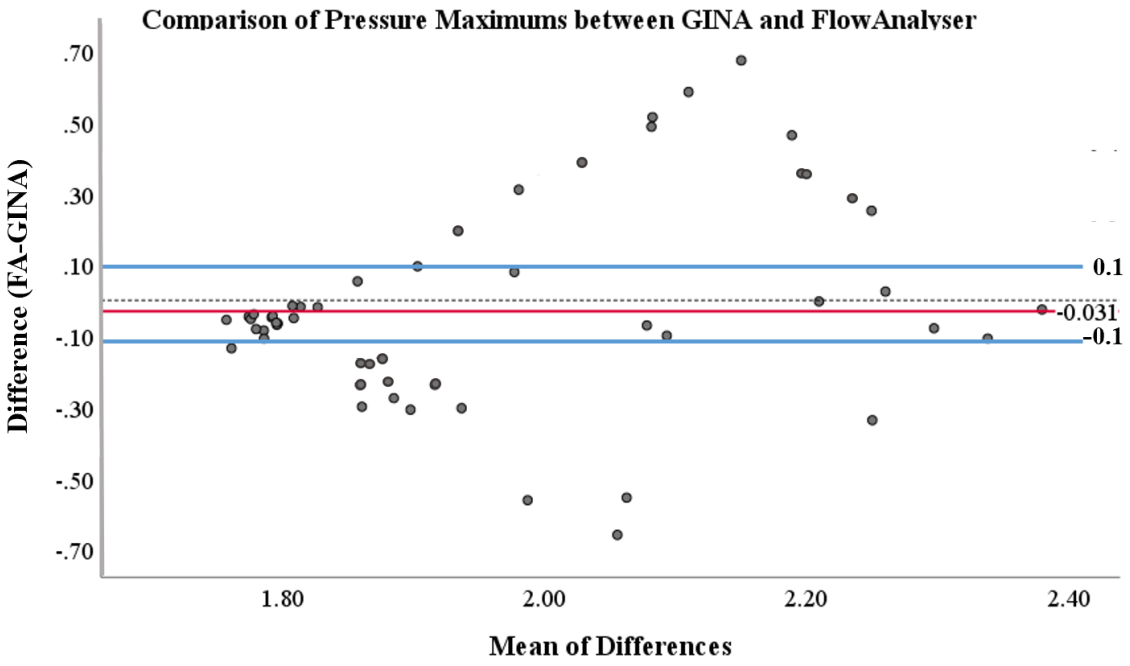
For $flow$, increasing frequency appears to increase the variation of differences while increasing Prm appears to decrease it. For pressure, increasing $flow$ and Prm both increase the variance. There are a few reasons why this may be the case. When Prm increases but $freq$ remains the same, a larger volume of air must be shifted at the same amount of time. This may cause greater mixing of air because of turbulence which ....
Additionally, it is interesting to see that for both flow and pressure, the variation of the differences is more pronounced during the exhalation phase. The simplest explanation for this is, as previously described, the lags in the rate of flow change and pressure change, respectively as during exhalation the rate of change is at its



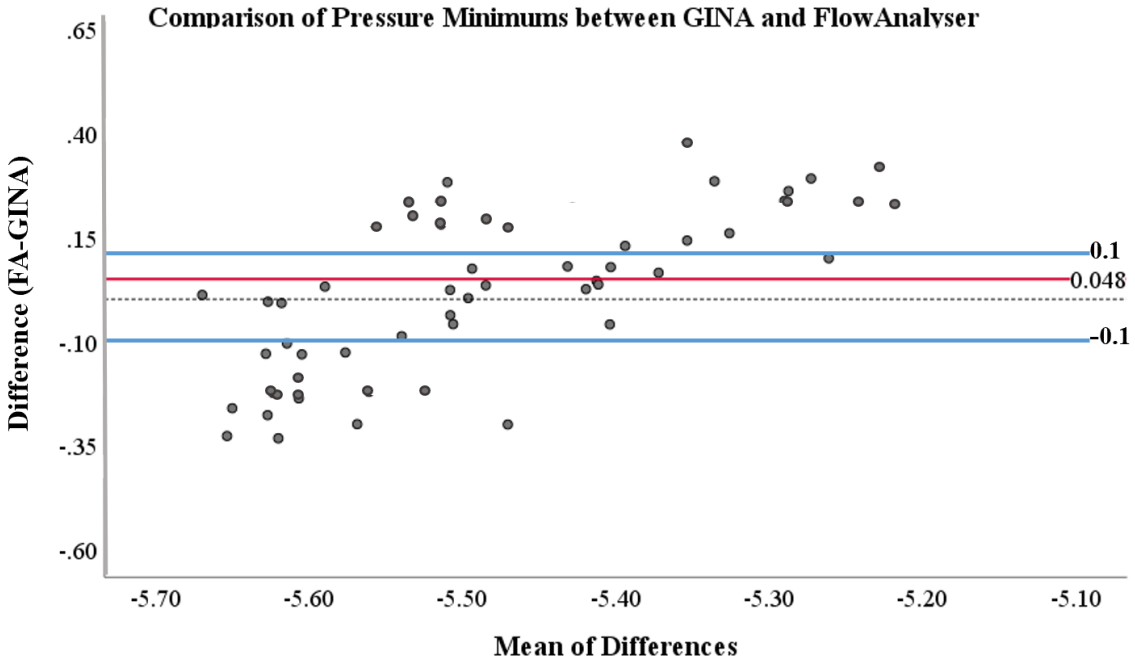
(a) Comparison of flow maximums on spontaneous breath data with parameters: freq=80, Prm=20



(b) Comparison of flow maximums on spontaneous breath data with parameters: freq=80, Prm=20

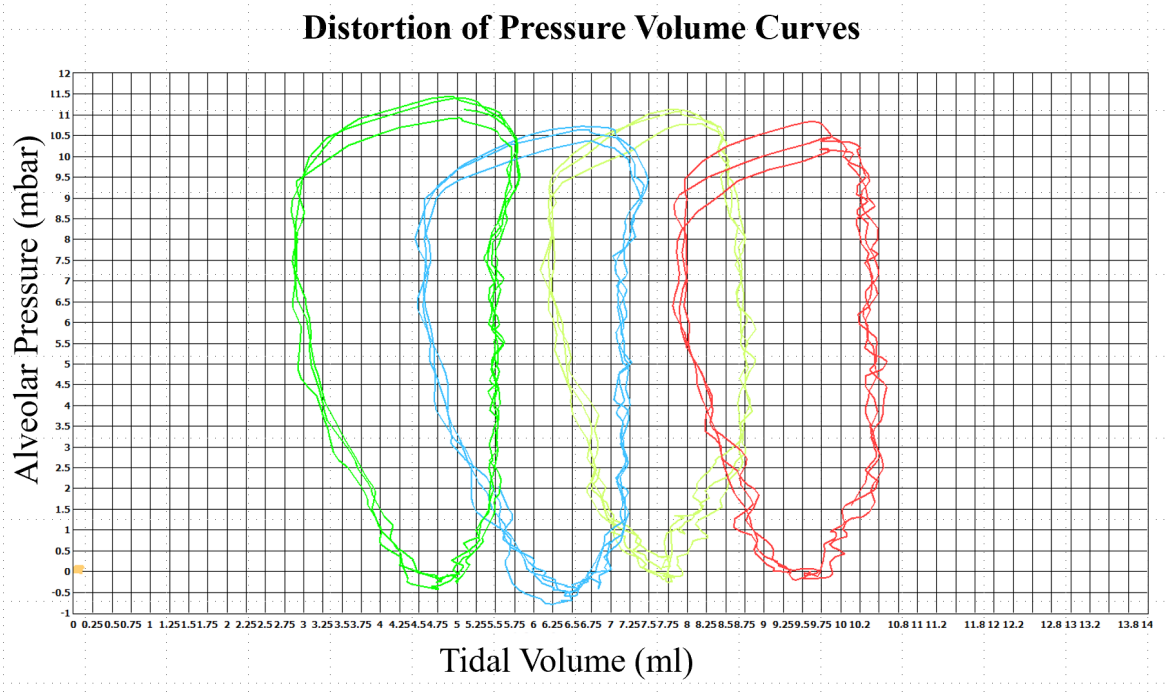


(c) Comparison of flow maximums on spontaneous breath data with parameters: freq=80, Prm=20

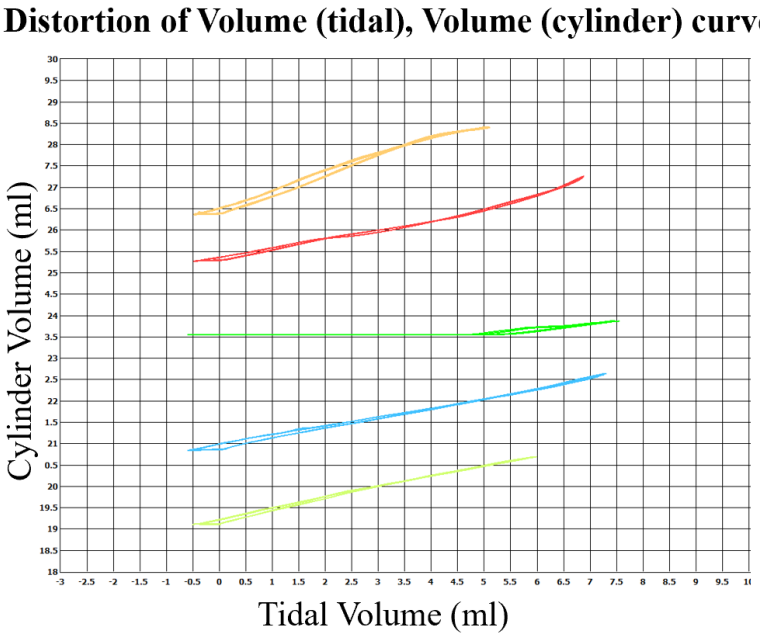


(d) Comparison of flow maximums on spontaneous breath data with parameters: freq=80, Prm=20

Figure 3.12: BA analysis comparing flow and pressure maximums and minimums between GINA and FlowAnalyser



(a) PEEP = 2.5, 5, 7.5, 10



(b) PEEP = 2.5, 5, 7.5, 10, 12.5

Figure 3.13: Bad morphology of spontaneous breaths in GINA. These plots show evidence of distortion across a range of applied pressures via CPAP and TPR. (a) Distortion in pressure volume curves, (b) distortion in volume (tidal) volume (cylinder) curves.

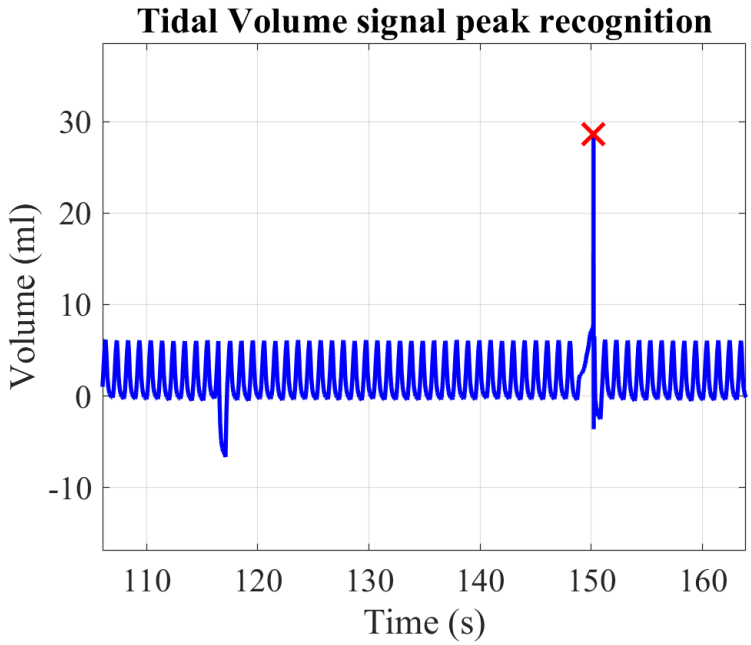


Figure 3.14: Improper breath due to NALM mechanical error

maximum.
The second set of BA analysis which looks at peaks reveals what we already know - that the amplitude of the GINA flow signal exceeds the FlowAnalyser, and the pressure signal amplitudes are very similar. However, it turns out that the mean of differences for flow is beyond the accepted error rate. While the mean of differences for pressure is in the acceptable range, it is surprising to see such variation. Although the linear regressions of the whole-signal BA analyses stay within the 95% confidence interval bounds, the signal amplitude differences are an indication of systemic error. Taking into consideration the second BA's, it is likely that we have a sytemic error. It is possible that this is simply a calibration error, especially because there appears to be no systemic change in the means of differences between in the whole-signal BA tests. There are a number of other factors that may have contributed to the inaccuracies that have not bee quantified, which is a limitation of this study. These are worth investigating in the future for a better determination of how accurate GINA truly is. For example:

- It has been shown that with increased frequency of piston movement, the phase delay between the volume recordings at the piston and the lung model outlet increases [46].
- The piston movement in GINA causes a vibration which is likely to be causing fluid oscilations and therefore turbulence, interfering with pressure and flow readings.

- An ideal mechanical lung model has either adiabatic or isothermal internal conditions. This is not the case in the GINA. Therefore, according to the ideal gas law, as the air heats up due to the work of the piston, the pressure and volume in the model increases.
- Bernoulli's effects are particularly impactful in tubes with small diameters.
- Having explored the GINA in great depth over the last year, we have noticed that pressure signals always exhibit a gradual downward drift. This indicates that GINA has an internal leakage, which reduces its accuracy.

Therefore, although GINA passed the accuracy tests prescribed by our methods, we would recommend further testing to determine the impacts of the above-mentioned and to fully ascertain GINA's accuracy.

3.6.2 Analysis of how realistic GINA's behaviour is

It will never be really real heat

3.6.3 Evaluation of the GINA Validator software

The GINA Validator software is simple to use and processes data quickly. It fulfilled its purposes for this experiment. Before it can be used in the future however, there are some limitations that should be addressed. Limitations - installability, filters,

3.6.4 General discussion

Can't preset simulations clunkiness

4 | Core Study Two: Software development for useful breath-by-breath output

4.1 Background

Although the ASL 5000s software has been described as "difficult to use" [26], one of its great advantages is that it calculates and outputs breath-by-breath data for over fifty parameters. The GINA does presently have this capacity. Additionally, GINA does not record parameter changes that are made during the signal recording. Therefore, it is presently useless to run any test simulations where the GINAs breathing changes due to internal stimuli (eg, user changing frequency, Prm etc). This seriously caps her usability. Personal communications with the manufacturers revealed that presently, they are not looking to develop such capabilities in the near future. Therefore, if the GINA is to be used soon for testing equipment, these features needs to be built in-house.

4.2 Aim

The aim of this study is to create a software for clinicians and researchers wanting to use the GINA to test devices. The software should read in GINA's output and transform it into a log of breath-by-breath parameters which can be used in statistical analysis.

4.3 Methodology

4.3.1 Design of the GINA Analyser software

The *GINA Analyser* software was written in Matlab code. This language was selected for its speed with data processing and its usability, as the software may need to be amended to in the future by others and many clinicians and engineers are

well versed in Matlab. An object-oriented code style was selected as it used less RAM than spaghetti code, is less corruptible, is easily read and can have functions added in a straightforward way.

The software design attempted to meet the Software Sustainability Institute's software evaluation criterion [59]. This criterion includes *usability*, which encompasses understandability, comprehensive and appropriate documentation, installability and learnability. It also includes *sustainability and maintainability* which includes testability, evolvability and changeability. Some *sustainability and maintainability* criterion such as governance and licencing are presently beyond the scope of this stage of development.

Respiratory clinicians and researchers, Thomas Drevhammer, Mark Tracy and Murray Hinder, and myself decided that at this stage and within the available timeframe, we required the following features:

- A unique identifier for each test
- Accurate calculations of the number of breaths and, for each breath: V_t, V_p, t_{in}, t_{ex}, P_y maximum, minimum and mean, P_{etr} maximum, minimum and mean, P_a maximum, minimum and mean.
- The ability to comment in additional details about each test run.

GINA Analyser has a simple graphical user interface (GUI) which the user interacts with (see fig. 3.5). This was designed using Matlab's GUIDE feature. The software design is divided into the following three sections: User interface, Error handling, Code flow and function design, and, Output.

4.3.1.1 User interface

The GINA Analyser application was designed for easy and simple usage. Full instructions are included in appendix B.2. Figure 4.1 shows the application interface. By clicking the 'browse' button, a search in the directory named GA_TestData, a subdirectory of the software directory opens. Here the user can select multiple excel files. The user can choose to create a new log of the calculated breath-by-breath data, or add to an existing log. They can choose to calculate breaths using the work of work of breathing signal or the flow signal. Selecting 'Go' runs the data analysis. Selecting 'Reset' will restore the apps original settings. Selecting 'Help Menu' will cause the pdf instruction manual to be opened.

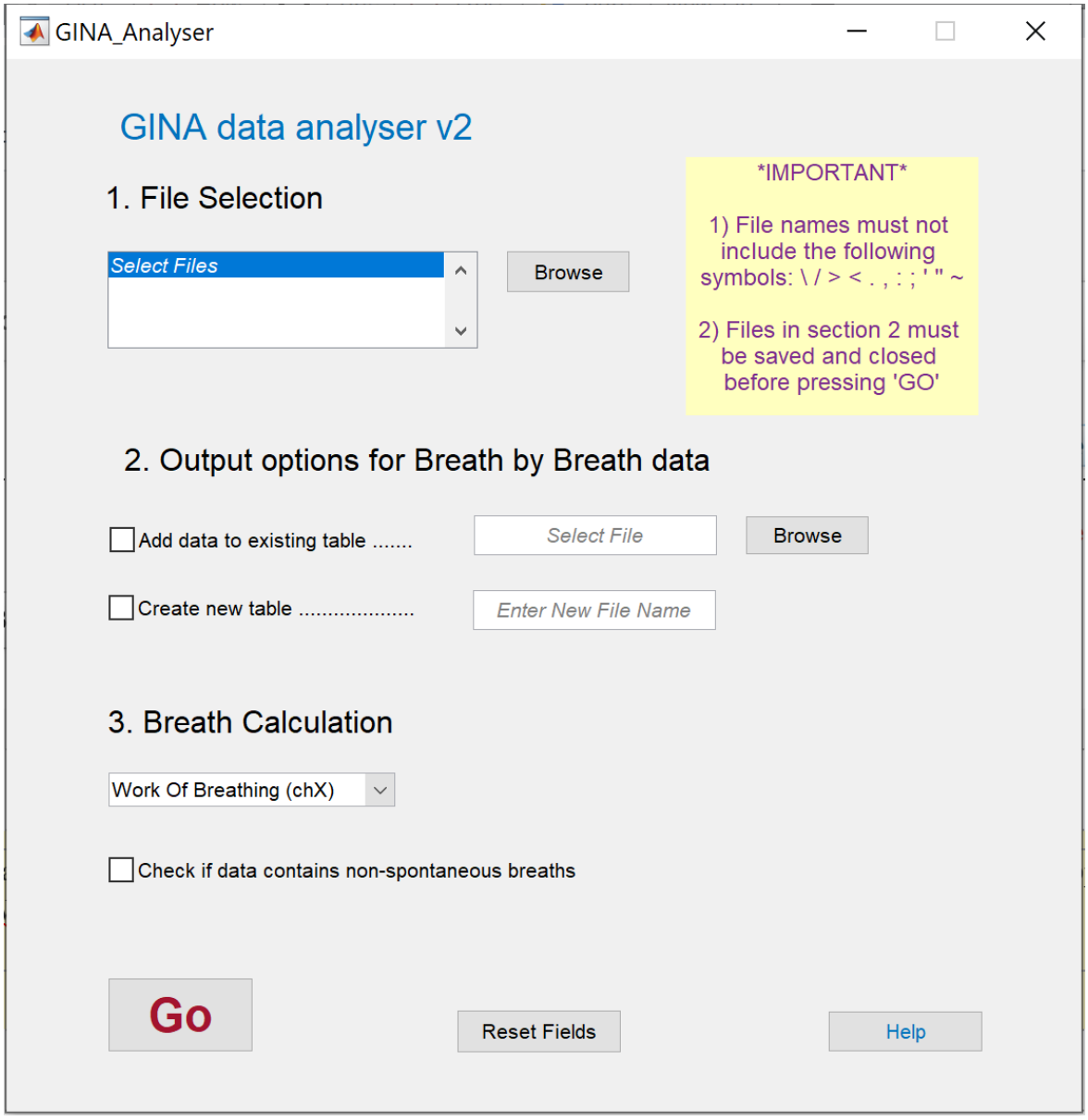


Figure 4.1: GINA Analyser app interface

4.3.1.2 Error handling

Figure 4.2 shows the flow process of how errors are caught including the problem, the prompts/error messages, how the user response to this is handled, and what action occurs.

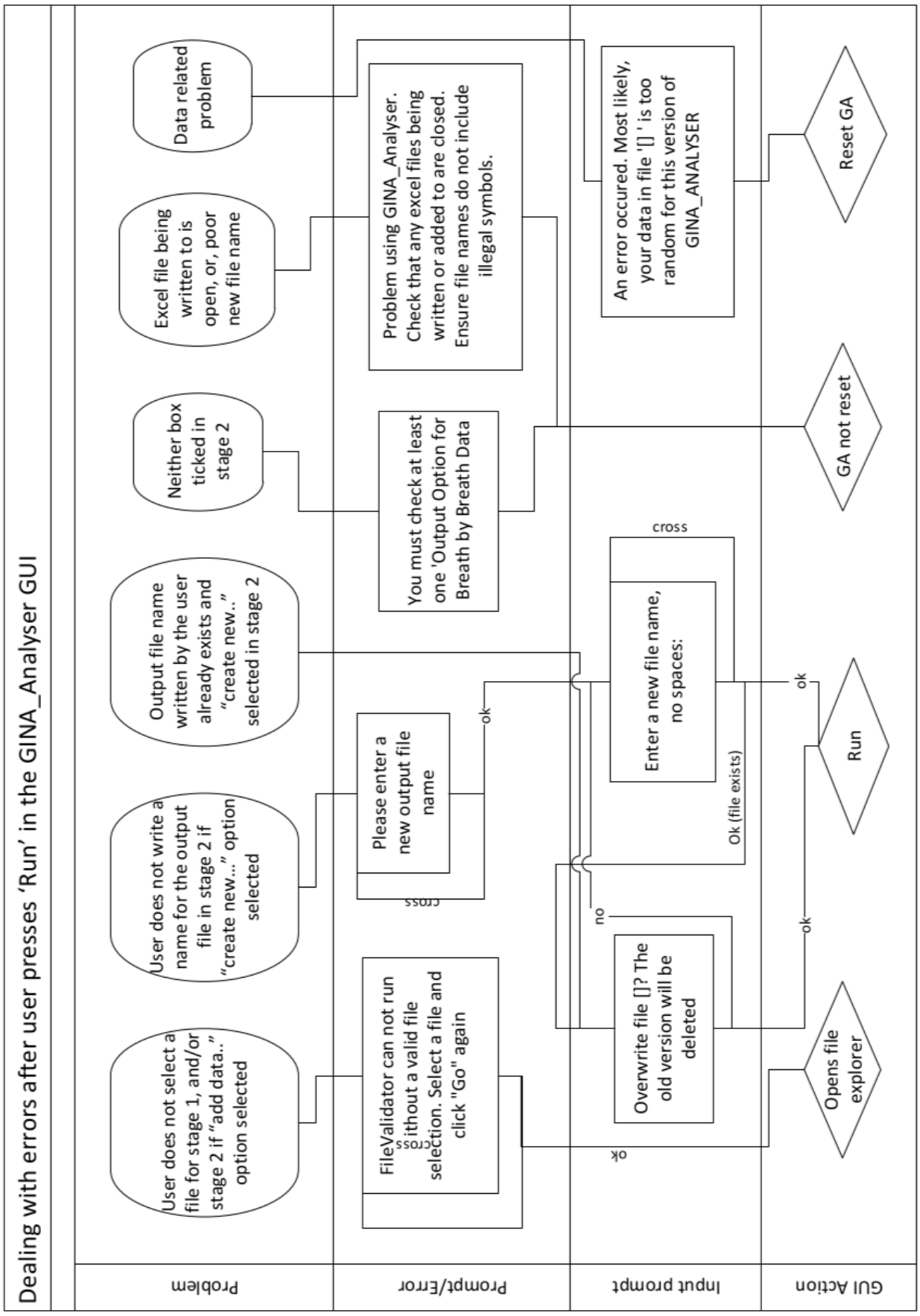


Figure 4.2: Handling errors in GINA Analyser software

4.3.1.3 Code flow and function design

There are three primary matlab files: *GINA_Analyser, ga_main* and *GA_Tester*. The full code for the first files is contained in appendix B.1. This is the code flow: GINA_Analyser collects user input, ga_main... GA_Tester ...

- *GINA_Analyser* creates the GUI application. The functions code the layout and the capabilities of each item (button, textbox etc) on the interface. Data is collected, tested for errors (see section *Error Handling* and sent to the *ga_main* file as global variables.
- *ga_main* imports the files. If the user has chosen to add to an existing log file, ga_main checks whether any of the files have already been added and prompt the user to decide to add it again or not by looking at the files own log of added files (fig. 4.4). Remaining filenames and the flag representing the users choice of breath calculation by WOB or Flow are sent to GA_Tester.
- *GA_Tester* runs the analysis on the imported data and tabularises it. The unique identifier for each test is the test name. Output is returned to the main function.
- *ga_main* Compiles the table and saves the excel output file.

This is the code flow and function design of *GA_Tester*. Refer to appendix B.1

function obj = GA_Tester

Code line reference 47-58

Input filename, flag, spontcheck

Description This is the object initialization function. The GINA output xlsx file is read in as a table. The flag indicates whether user selected breath calculation by WOB (flag=1) or Flow (flag=2). Spontcheck indicates if the box for non-spontaneous breaths was checked. These parameters are set as object properties. Function bpb_analyser is called.

function bpb_analyser

Code line reference 62-170

Input object properties, flag

Description This function tries to call all remaining functions. The directory for saving data is created and all open figures are saved to it. The log table is created: headers are set, input file name is used for the rownames, table size is set according to the number of calculated breaths. Table is populated and bad data is trimmed. If an error occurs during the process, this is caught and an error message displayed. All open figures are saved to the directory.

function freqfinder

Code line reference 174-304

Input object properties, flag

Description Function is divided into 2 if statements: if flag == 1, and if flag ==2 to guide the data analysis process.

- Flag == 1:
  - WOB signal read in from file. The WOB signal is 0 during expiration and non 0 during inspiration with a vertical drop from peak to 0 when inspiration ends. Time signal is adjusted to begin at 0 from startpoint. Data is trimmed so that it starts when WOB = 0. Peak analysis determines number of total breaths as peaks - 1.
- Flag == 2:
  - Flow signal read from file. If data is of spontaneous breaths, a strong lowpass filter is applied. This maintains peak height. If data is of non spontaneous breaths, a butterworth filter is applied. It filters the noise of the non-spont signal, but also chops peaks. Breaths are counted as every time the signal goes from negative to positive.

function inspexptime

Code line reference 308-487

Input object properties, flag

Description Function is divided into 2 if statements: if flag == 1, and if flag ==2 to guide the data analysis process.

- Flag == 1:

  - Based on when the signal is zero and non zero, inspiration starts and timelength, and expiration starts and timelength, are calculated. Data trimmed to start of first inspiration, and any signal after last full breath is chopped off.

- Flag == 2:
  - Based on where inspirations start (points of signal crossover from negative or flat to positive monotonic increasing) and expirations start (points of crossover from positive monotonic decreasing to negative monotonic decreasing), inspiration starts and timelength, and expiration starts and timelength, are calculated. Data trimmed to start of first inspiration, and any signal after last full breath is chopped off.

function tidalfinder

Code line reference 491-541

Input object properties, tnum (flag for T_v or T_p)

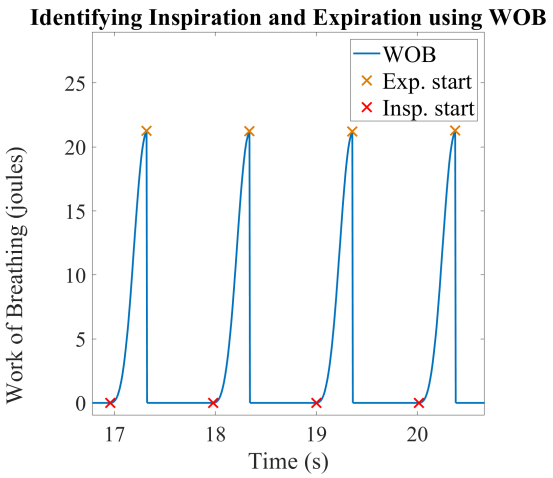
Description Function uses peak analysis to determine tidal volume and piston volume.

function pressure_data

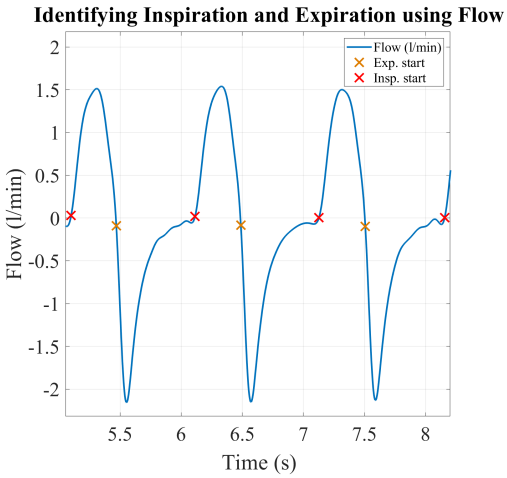
Code line reference 545-722

Input object properties, pnum (flag for P_y, P_{etr}, P_a)

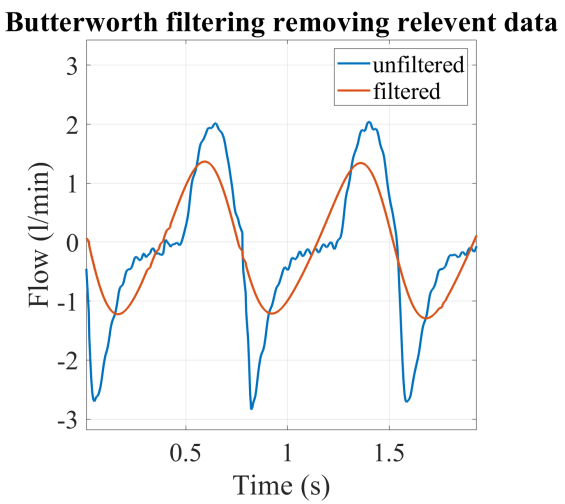
Description Function puts pressure data through a lowpass filter. Peak analysis is used to determine pressure minimums and maximums for each breath.



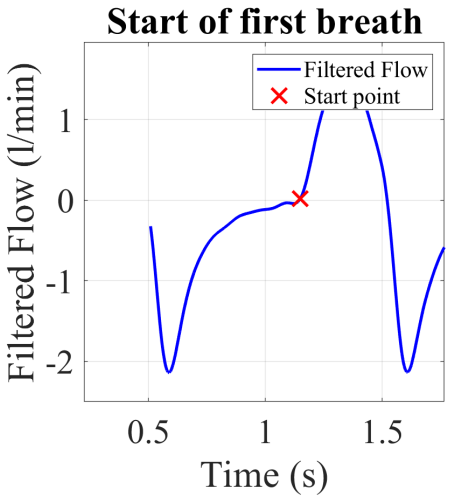
(c) Determining inspiration and expiration using WOB



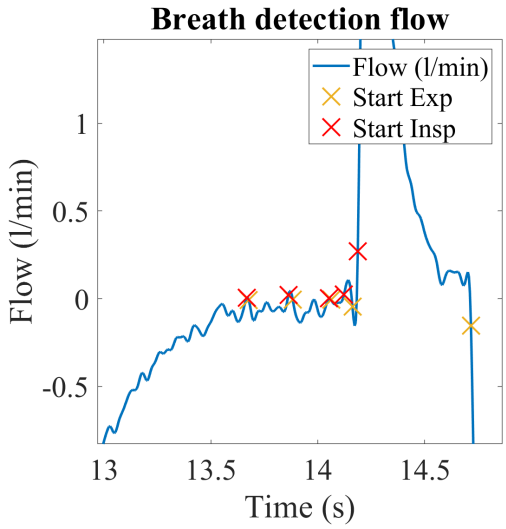
(d) Determining inspiration and expiration using flow



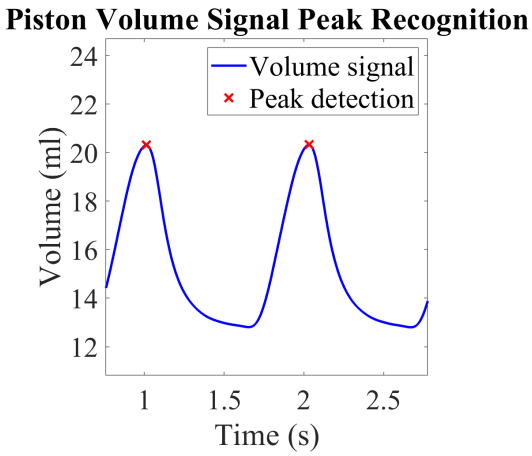
(a) Butterworth filtering on flow chops off peaks.



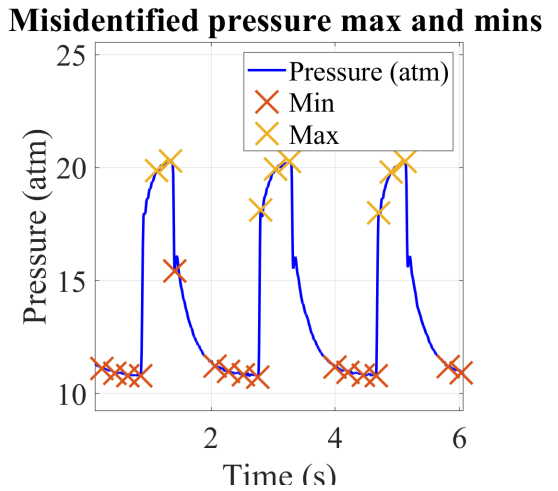
(b) Determining the start of the first breath on the flow wave. Data is trimmed to this point.



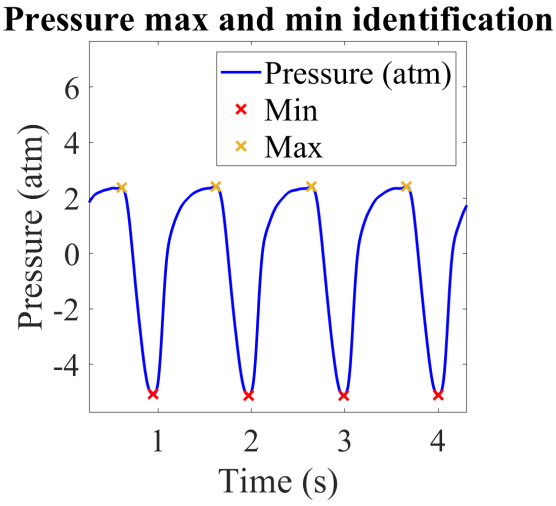
(e) Miscalculation of inspiration and expiration starts. Noisy non-spontaneous breathing flow signal requires a strong filter to remove noise.



(f) Determining volume for each breath.



(g) Miscalcution of pressure maximums and minimums on noisy signal



(h) Determining pressure maximums and minimums

Figure 4.3: GINA Analyser, GA_Tester output and code progression

4.3.1.4 Output

	A	B	C
1	**Data_Log**		
2	trim_F30_Tin0.8_Taui100_Taue260_C1.0_LL10_prm15.xlsx		
3	trim_F60_Tin0.4_Taui60_Taue120_C1.0_LL10_prm5.xlsx		
4	trim_F60_Tin0.4_Taui60_Taue120_C1.0_LL10_prm15.xlsx		
5	trim_F60_Tin0.4_Taui60_Taue120_C1.0_LL10_prm25.xlsx		
6	trim_F120_Tin0.2_Taui30_Taue60_C1.0_LL10.xlsx		
7			
8			

Figure 4.4: Example of GINA Analyser output file, sheet 2- log of files added.

Output is saved as a 2-tab excel file. The first tab has the breath-by-breath analysis (fig 4.5). Values are rounded to 2 decimal points, this was a clinician request. The second tab has the log of files added (fig. 4.4).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	TestName	Breath	Tidal_Volume	Piston_Volume	Insp_Time_s	Exp_Time_s	Py_max	Py_min	Py_mean	Ptr_max	Ptr_min	Ptr_mean	Palv_max	Palv_min	Palv_mean
2	trim_F30_Ti	1	6.15	20.41	0.73	1.28	2.25	-4.71	-0.19	2.36	-4.74	-0.10	2.37	-4.59	-0.09
3	trim_F30_Ti	2	6.11	20.42	0.73	1.28	2.25	-4.68	-0.18	2.36	-4.71	-0.14	2.37	-4.57	-0.07
4	trim_F30_Ti	3	6.15	20.40	0.74	1.29	2.24	-4.69	-0.19	2.34	-4.75	-0.11	2.37	-4.59	-0.06
5	trim_F30_Ti	4	6.14	20.41	0.72	1.28	2.26	-4.68	-0.20	2.36	-4.72	-0.10	2.40	-4.56	-0.10
6	trim_F30_Ti	5	6.08	20.41	0.73	1.29	2.26	-4.69	-0.18	2.38	-4.72	-0.10	2.40	-4.58	-0.04
7	trim_F30_Ti	6	6.24	20.41	0.73	1.28	2.26	-4.69	-0.17	2.34	-4.73	-0.10	2.38	-4.58	-0.09
8	trim_F30_Ti	7	6.20	20.41	0.73	1.28	2.25	-4.70	-0.18	2.33	-4.73	-0.09	2.38	-4.59	-0.08
9	trim_F30_Ti	8	6.08	20.43	0.73	1.28	2.26	-4.67	-0.19	2.36	-4.73	-0.13	2.41	-4.59	-0.06
10	trim_F30_Ti	9	6.04	20.42	0.73	1.29	2.26	-4.68	-0.19	2.37	-4.73	-0.11	2.39	-4.56	-0.07
11	trim_F30_Ti	10	6.14	20.41	0.73	1.29	2.27	-4.70	-0.16	2.36	-4.69	-0.06	2.41	-4.58	-0.05
12	trim_F30_Ti	11	6.15	20.43	0.72	1.28	2.28	-4.68	-0.18	2.39	-4.67	-0.12	2.39	-4.56	-0.07
13	trim_F30_Ti	12	6.13	20.42	0.74	1.28	2.27	-4.68	-0.17	2.36	-4.71	-0.10	2.38	-4.54	-0.04
14	trim_F30_Ti	13	6.07	20.42	0.73	1.28	2.27	-4.68	-0.16	2.39	-4.69	-0.07	2.39	-4.55	-0.08
15	trim_F30_Ti	14	6.15	20.44	0.74	1.29	2.29	-4.69	-0.19	2.38	-4.72	-0.09	2.38	-4.57	-0.06
16	trim_F30_Ti	15	6.08	20.43	0.73	1.28	2.26	-4.68	-0.16	2.37	-4.73	-0.10	2.39	-4.58	-0.04
17	trim_F30_Ti	16	6.12	20.43	0.74	1.29	2.26	-4.68	-0.18	2.39	-4.71	-0.07	2.38	-4.57	-0.04
18	trim_F30_Ti	17	6.08	20.44	0.73	1.29	2.28	-4.69	-0.16	2.35	-4.70	-0.09	2.40	-4.57	-0.04
19	trim_F30_Ti	18	6.13	20.44	0.73	1.29	2.30	-4.66	-0.14	2.42	-4.69	-0.08	2.42	-4.57	-0.05
20	trim_F60_Ti	1	2.16	14.66	0.38	0.66	1.01	-1.91	0.01	1.10	-1.98	0.08	1.14	-1.83	0.12
21	trim_F60_Ti	2	2.15	14.65	0.36	0.66	1.00	-1.92	0.00	1.14	-1.95	0.08	1.11	-1.82	0.12
22	trim_F60_Ti	3	2.14	14.65	0.36	0.64	1.01	-1.94	0.03	1.12	-1.96	0.08	1.13	-1.84	0.13
23	trim_F60_Ti	4	2.22	14.66	0.38	0.64	0.99	-1.92	0.00	1.09	-1.97	0.07	1.10	-1.83	0.10
24	trim_F60_Ti	5	2.15	14.65	0.37	0.65	1.00	-1.91	0.00	1.11	-1.96	0.08	1.10	-1.82	0.10
25	trim_F60_Ti	6	2.12	14.65	0.37	0.63	1.00	-1.92	0.00	1.07	-1.97	0.07	1.08	-1.83	0.10
26	trim_F60_Ti	7	2.22	14.65	0.39	0.65	1.00	-1.91	-0.01	1.09	-1.95	0.06	1.10	-1.82	0.10
27	trim_F60_Ti	8	2.13	14.64	0.36	0.66	0.99	-1.92	-0.01	1.07	-1.97	0.06	1.10	-1.82	0.10
28	trim_F60_Ti	9	2.13	14.66	0.36	0.65	0.98	-1.93	0.00	1.09	-1.97	0.08	1.11	-1.84	0.09
29	trim_F60_Ti	10	2.19	14.65	0.36	0.64	0.99	-1.92	0.00	1.13	-1.99	0.08	1.09	-1.82	0.11
30	trim_F60_Ti	11	2.16	14.64	0.37	0.66	0.98	-1.92	-0.01	1.10	-1.98	0.05	1.09	-1.82	0.10
31	trim_F60_Ti	12	2.18	14.65	0.36	0.65	0.97	-1.94	-0.02	1.09	-1.98	0.06	1.08	-1.81	0.11
32	trim_F60_Ti	13	2.12	14.64	0.36	0.65	0.98	-1.91	-0.02	1.07	-1.93	0.08	1.10	-1.82	0.08
33	trim_F60_Ti	14	2.14	14.63	0.37	0.65	0.99	-1.96	-0.01	1.10	-1.97	0.07	1.09	-1.84	0.11
34	trim_F60_Ti	15	2.14	14.65	0.36	0.66	0.99	-1.96	-0.01	1.08	-2.00	0.08	1.08	-1.82	0.11
35	trim_F60_Ti	16	2.13	14.64	0.36	0.65	0.98	-1.93	-0.01	1.08	-1.95	0.05	1.10	-1.83	0.10
36	trim_F60_Ti	17	2.18	14.64	0.37	0.65	0.97	-1.93	-0.01	1.09	-1.95	0.08	1.08	-1.83	0.11
37	trim_F60_Ti	18	2.20	14.63	0.36	0.65	0.99	-1.94	-0.02	1.09	-1.94	0.06	1.12	-1.84	0.10

Sheet1 Sheet2 Sheet3

Figure 4.5: Example of GINA Analyser output file, sheet 1 - breath by breath data.

4.3.2 Test conditions for assessing the GINA Analyser software

GINA Analyser's validity was assessed through testing:

1. that the output file meets the design requirements: a unique identifier for each test, calculations of the parameters, and the ability to comment. Additionally, that output was saved to the correct directories.

2. its ability to calculate output parameters for spontaneous breathing by the WOB technique or flow technique. The independent variables were $freq$ at 30, 60 and 120 bpm, and Prm at 5, 15 and 25 N. Dependent variables were C_{int} (1.0), endotracheal diameter (5 mm) and airway resistance (2 cmH2O). t_{in}, τ_{in}, and τ_{out} were changed as required to create ideal flow wave morphology. The output parameters tested were V_t, t_{in}, t_{ex} and P_y maximum, minimum and mean. V_p, P_{etr} maximum, minimum and mean, P_a maximum, minimum and mean didn't need to be assessed as the functions *tidalfinder* and *pressure_data* are already tested by V_t and P_y respectively. Correlation analysis was used to compare the signals calculated by each method to each other and to measurements taken manually off the original output signal, using a TDMS file viewer. Additionally, manual measurements were taken for both the filtered and unfiltered pressure signals and BA analysis was performed to determine whether the filter significantly changes the output values. The boundaries were set at ± 0.1, half of the acceptable error of GINA.

3. its ability to complete all functions for long data sequences. Five 10 minute sequences were tested with independent variables $freq$ at 30, 60 and 120 bpm and Prm at 5, 15 and 25 N .

4. its ability to complete all functions on alternating spontaneous and non spontaneous breathing. A signal of alternating 10 s spontaneous and 10s non spontaneous was tested.

5. its ability to complete all functions on random breathing sequences including apneas. The 'random' mode on GINA was used to generate this signal.

6. its ability to calculate output parameters for non-spontaneous breathing by the flow method. The independent variable was $freq$ at 30, 60 and 120 bpm. Dependent variables were C_{int} (1.0), endotracheal diameter (5 mm), airway resistance (2) t_{in} (0.4 s), τ_{in} (60), and τ_{out} (60).

The two test criterion for assessing the GINA Analyser software are that:

1. It passes the conditions of the Software Sustainability Institute's criterion based assessment [59] that are relevant at this pre-distribution stage of development,

2. It is valid, that is, it does what it is meant to do.

4.4 Results

4.4.1 Assessment of GINA Analyser validity

4.4.1.1 Accuracy of calculations by the flow and WOB techniques

As mentioned previously, the relationship between the manual and flow or WOB technique calculations represents accuracy. Therefore, we assessed whether the flow and WOB techniques each produced accurate calculations of the parameters (V_t, t_{in}, t_{ex}, P_y maximum, minimum and mean) by determining Pearson's correlations between the technique and manual measurements. These correlations are in figure 4.6. Tidal volume measurements were omitted from the table because T_v values were identical across each technique.

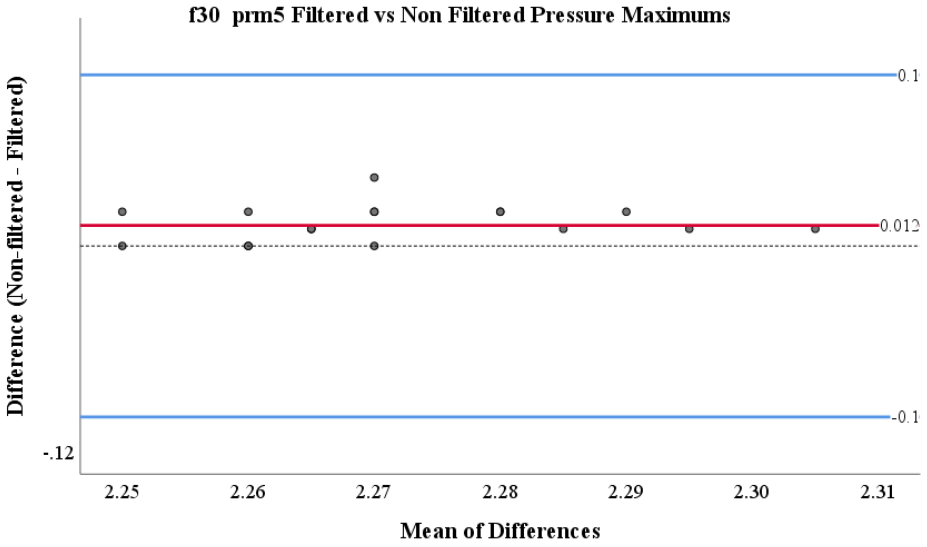
For the test cases involving higher frequencies (freq = 60 bpm and freq = 120 bpm), both flow and WOB measurements were significantly associated with the manual values, indicating a high degree of accuracy. When it comes to the lowest frequency, freq = 30 bpm, it is interesting to see that neither test was able to achieve accuracy for t_{in} or t_{ex}, but both techniques calculations were statistically identical. However, for this frequency, pressure values were significantly associated with the manual values. There was a trend towards increased accuracy of all calculations as Prm increased. There was no difference between the output from either techniques.

	Pearsons correlation											
	WOB vs Flow				WOB vs Manual				Flow vs Manual			
	t_{in}	t_{ex}	P_{max}	P_{min}	t_{in}	t_{ex}	P_{max}	P_{min}	t_{in}	t_{ex}	P_{max}	P_{min}
f30 Prm15	1.000**	1.000**	1.000**	.979**	-.139	-.125	1.000**	.979**	-.139	-.125	1.000 **	.979**
f60 Prm15	1.000**	1.000**	.532**	.237	.478*	.514**	.532**	-.003	.478*	.514**	1.000**	.464*
f120 Prm15	1.000**	1.000**	1.000**	1.000**	.512*	.550*	1.000*	1.000*	.512*	.550*	1.000**	1.000*
f60 Prm5	1.000**	1.000**	1.000**	1.000**	.420	-.299	.613*	-.488	.420	-.299	.613*	-.488
f60 Prm 25	1.000**	1.000**	1.000**	1.000**	1.000**	1.000**	1.000**	-.135	1.000**	1.000**	1.000**	-.135

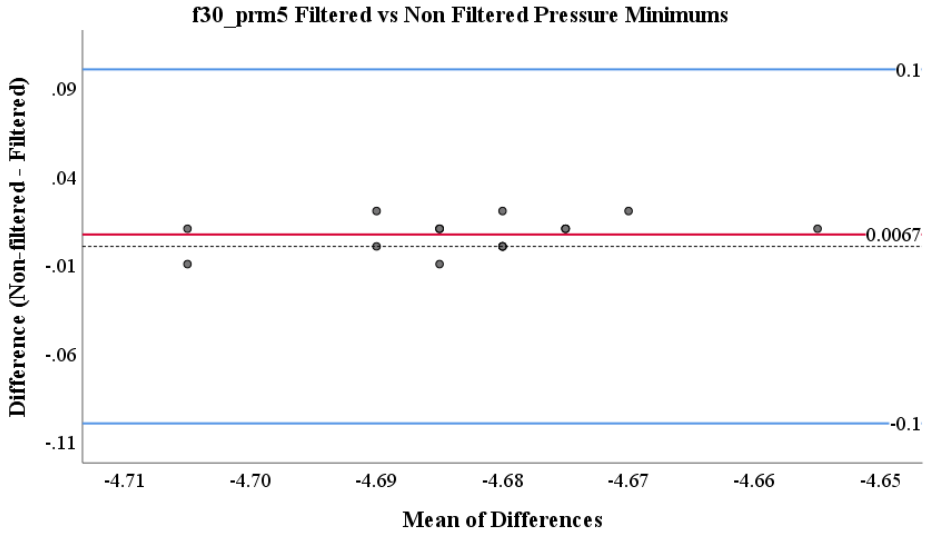
* p<.05, ** p<.01, *** p<.001

Figure 4.6: Comparison of the flow method, WOB method and manual measurements by Pearson's correlation for GINA Analyser output.

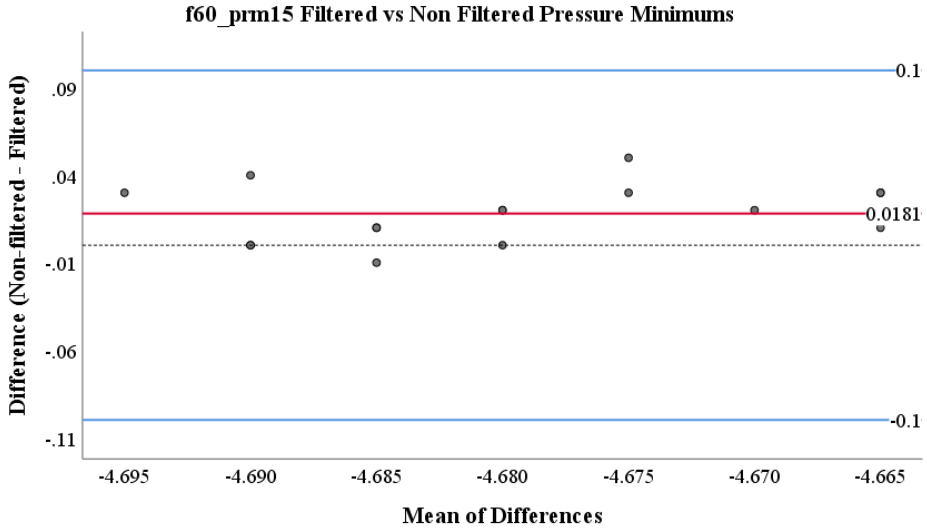
Bland Altman analysis of the difference between pressure maximums and minimums on the filtered vs. the unfiltered signal showed no significant mean of differences across all conditions (see fig. 4.7). Interestingly, in the case of minimums for freq = 120 bpm and Prm = 15, the mean of differences very closely approximated the lower boundary. This may indicate that the filter overexaggerates minimums at high frequencies but this trend is not supported by other cases.



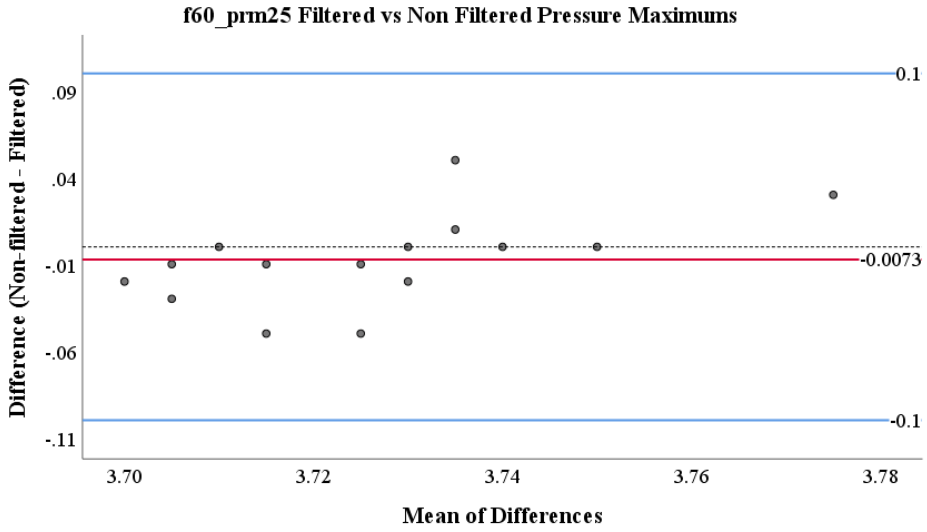
(a) Pressure maximums, freq = 30, Prm = 5



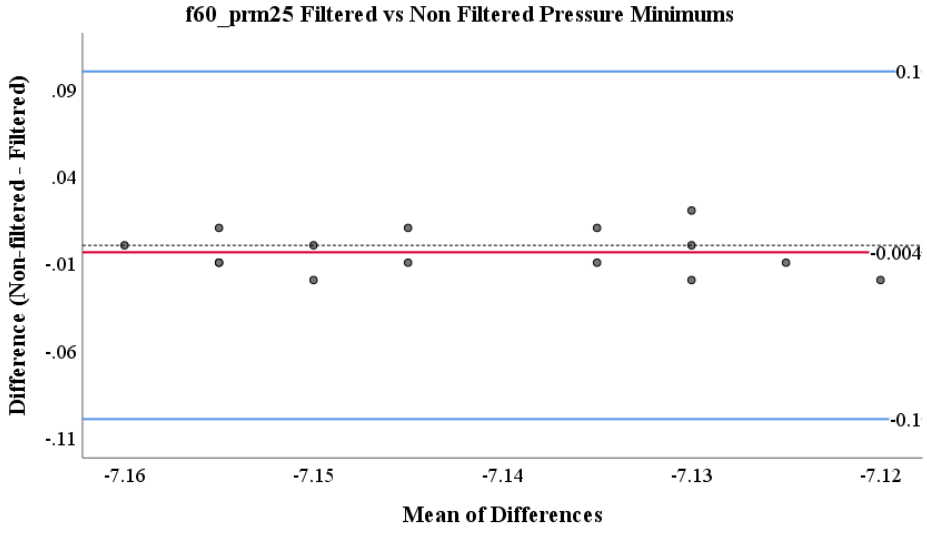
(b) Pressure minimums, freq = 30, Prm = 5



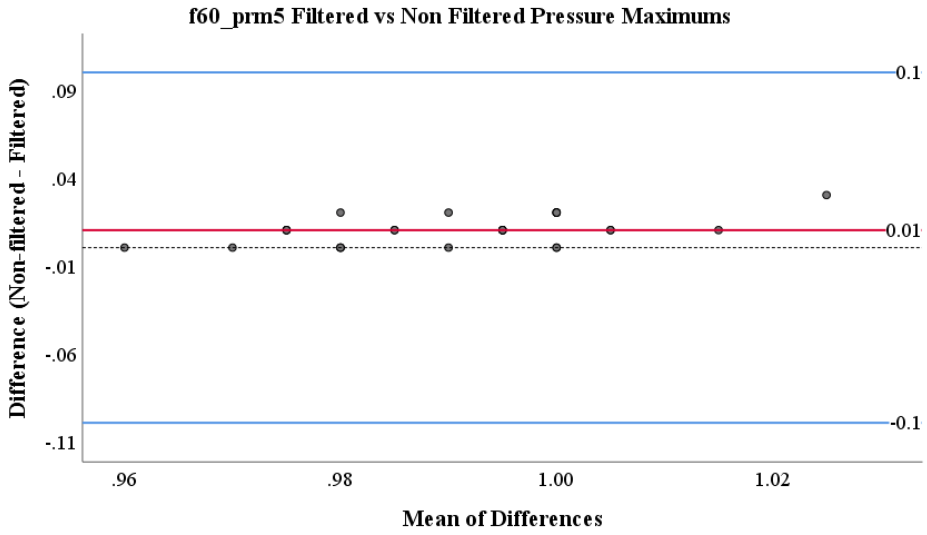
(f) Pressure minimums, freq = 60, Prm =1 5



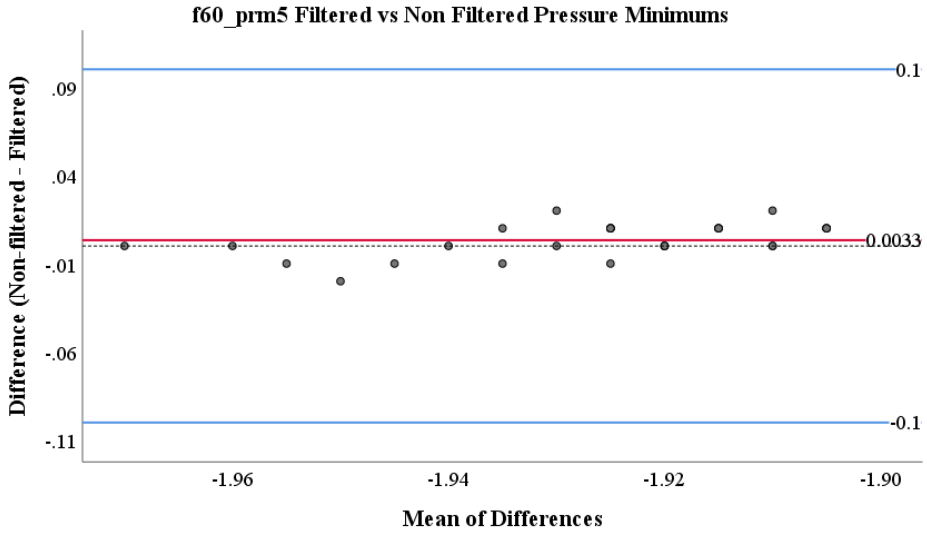
(g) Pressure maximums, freq = 60, Prm = 25



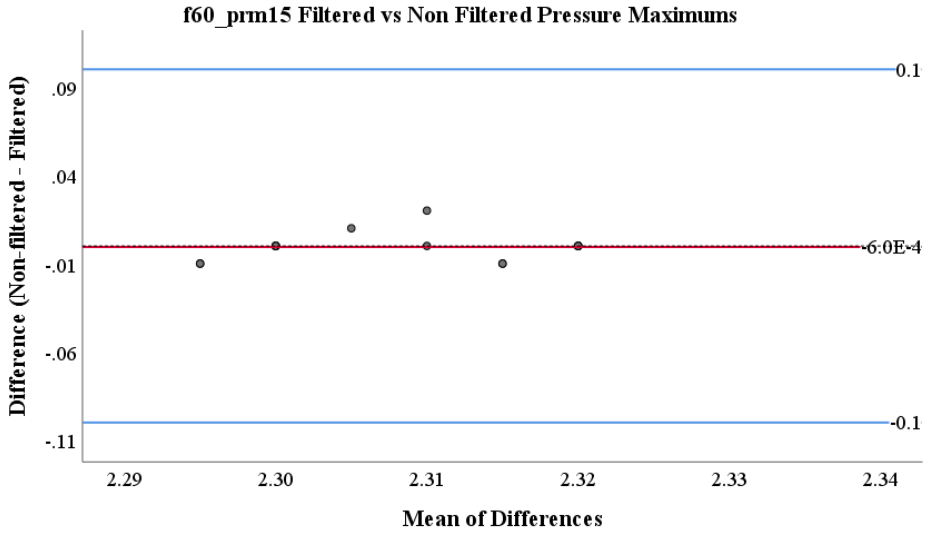
(h) Pressure minimums, freq = 60, Prm = 25



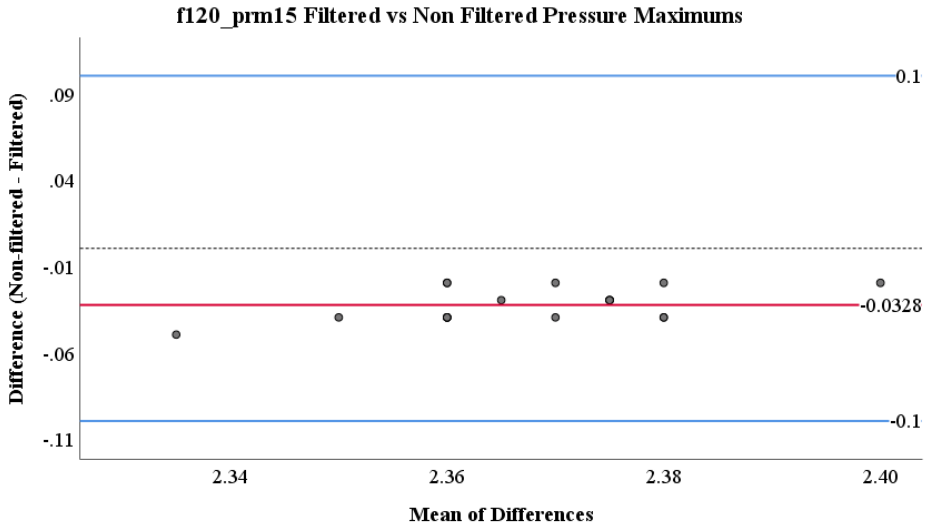
(c) Pressure maximums, freq = 60, Prm = 5



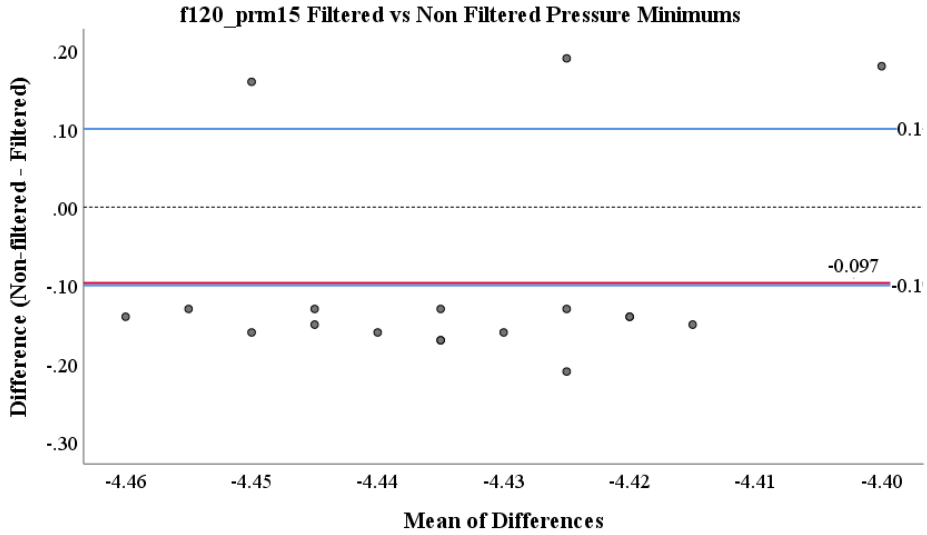
(d) Pressure minimums, freq = 60, Prm = 5



(e) Pressure maximums, freq = 60, Prm = 15



(i) Pressure maximums, freq = 120, Prm = 15



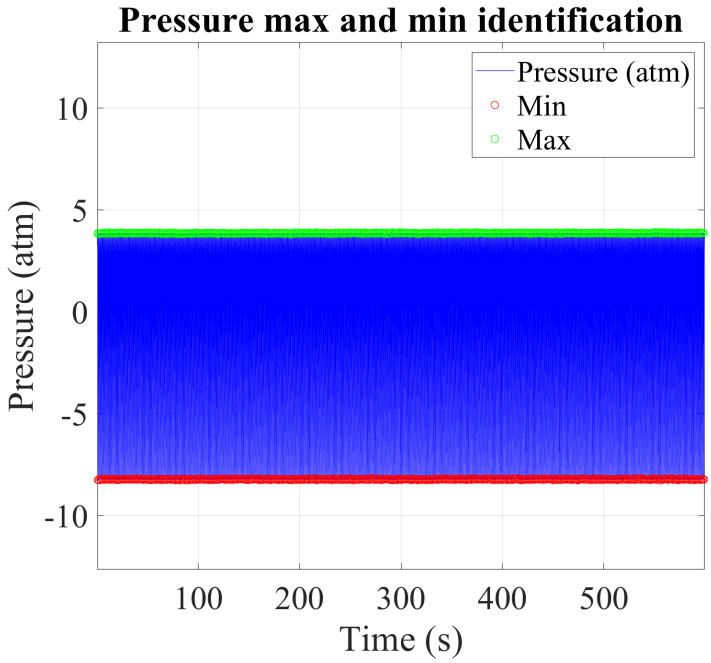
(j) Pressure minimums, freq = 120, Prm = 15

Figure 4.7: BA analysis comparing filtered and unfiltered pressure signals

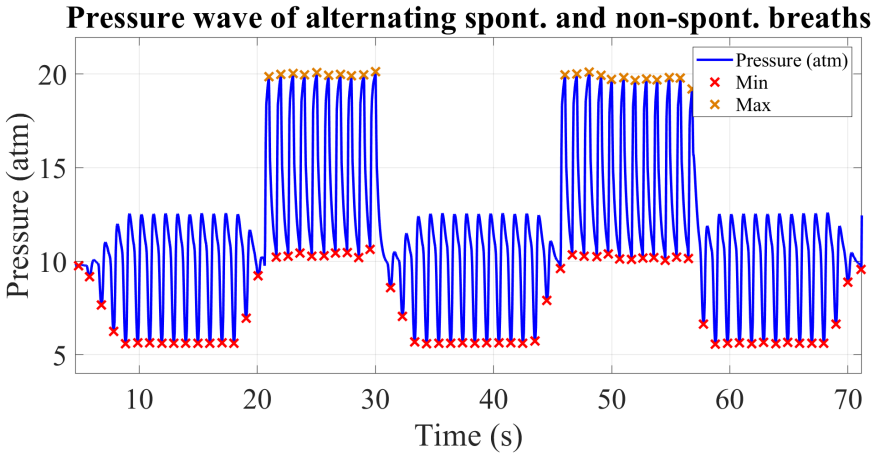
4.4.2 Assessment of GINA Analyser capabilities

The GINA Analyser software was able to run to completion for both spontaneous and non-spontaneous breathing datasets of up to two minutes long. When tested with 10 minutes of spontaneous breathing data, the software consistently could only manage to calculate breaths, inspiratory times, expiratory times, tidal volume, piston volume and pressure at the outlet, but failed when it came to endotracheal pressure and alveolar pressure. When breaths were calculated using the flow signal, software failed to calculate anything when there was a large spike or interruption

in the data as a result of a mechanical incident with the machine. When the WOB signal was used, the software could calculate breaths and breath times only. The GINA Analyser was not able to handle data that alternated between spontaneous and non spontaneous breathing, or random breathing.



(a) GINA Analyser can work on long data sets.



(b) GINA Analyser is unable to correctly calculate pressure max and min for alternating spontaneous and non-spontaneous breathing because of the noise as the signals switch.

Figure 4.8: GINA_Analyser capabilities

4.4.3 Software criterion based assessment

According to preliminary assessment before software distribution, the GINA Analyser software is valid. It does what required of it, and all errors are caught and handled. It performs the analysis to calculate breath-by-breath values for the clinician-desired parameters : V_t, V_p, t_{in}, t_{ex}, P_y maximum, minimum and mean, P_{etr} maximum, minimum and mean, P_a maximum, minimum and mean. The data output meets the requirements set out in the methods. There is a unique identifier for each test. As demonstrated in the above sections, both the flow and WOB calculation techniques produce accurate outcomes. As required, comments can be left. The second sheet of the excel output file which logs the tests performed is particularly fit for purpose. Finally, data is saved in the subdirectory named by the user.

When it comes to the criterion based assessment, GINA Analyser passes some most conditions. Under the *usability* criterion, the software passed understandability, documentation and learnability: the application interface is simple and easy to understand, and clear documentation of how to implement the software is included. Under the *sustainability and maintainability* criterion, the software passes changeability, testability and evolveability, but not portability. This is because when the software is installed on different computers, although it runs, the app interface features (buttons, words etc) do not properly oriented. This is a problem with the GUIDE Matlab plug-in.

4.5 Discussion

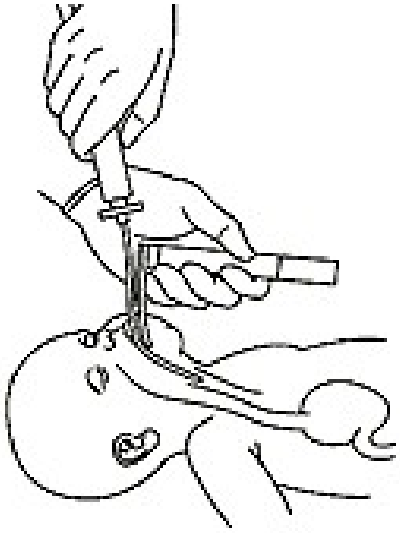
Summary of findings
Were the correlations and bland altmans acceptable? Why was there the trends?
Room for improvement with the filters? The question of what is the true value. Should peaks be preserved?
Future directions for the software: ability to filter out big spikes, Extra parameters A different language, perhaps python, C or Java. Perhaps the ability to convert tdms files.

5 | Case Study One: Interfacing ventilators with neonatal patients

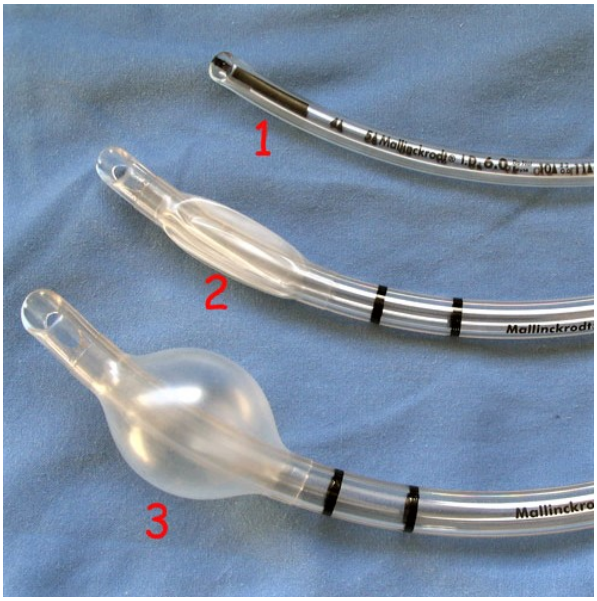
One of the most common methods of interfacing ventilators with patients for the treatment of respiratory distress syndrome is through endotracheal intubation [60]. This avoids any issues caused by upper airway collapse and there is less leak in the system than a face mask or nasal prongs (see fig. 5.1a). A recent (2016) study found that adverse events occur in 4 out of 10 intubations of neonates, with severe events occuring 8.8% of the time. Predictors of adverse events include multiple intubation attempts and emergency intubations. Adverse events that can be associated with intubation include CPR, tracheal perforation, airway bleeding or trauma, rigid-chest syndrome, emesis, pneumothorax, mainstream bronchial intubation, hypoxia and bradycardia [61]. In the first part of this chapter we explore what intubation involves, the mechanical interaction of the ETT with the endotrachea, and, the material properties of endotracheal tubes (ETT). In the second section we look at how ETT design has changed with time and current developments.

5.1 The practice and effects of endotracheal intubation

Modern ETTs are made from polyvinyl chloride composites (PVC) or polyurethane composites. These are tough but flexible polymers that are relatively bio-inert and have many biomedical applications. The ETT used on older children and adults is a cuffed tube. In these the tube diameter is smaller than the endotrachea and so a small cuff on the tube is inflated to hold the ETT in place and prevent leakage. However, the diameter of the neonates endotrachea is very small ($\approx$ 3 mm for extreme and very preterm infants [43]), and, some research suggests that it may be shaped conically, preventing proper cusp inflation [60]. Since tube resistance increases according to diameter reduction to the fourth exponent, it is most common practice to use uncuffed endotracheal tubes in neonates in order to minimise this resistance (see fig. 5.1b). Additionally, as described in section *Neonatal Respiratory Physiology*, the neonatal airways are substantially less elastic and are thin and



(a) https://www.bettersafercare.vic.gov.au/resources/clinical-guidance/maternity-and-newborn-clinical-network/meconium-stained-liquor-msl-management



(b) https://www.howequipmentworks.com/tracheal_tubes/

Figure 5.1: Endotracheal Intubation. (a) The intubation process. (b1)Uncuffed ETT (b2) Deflated cuffed ETT, (b3) Inflated cuffed ETT

weak. Although the neonatal endotrachea is supported by cartilage, the tissue lining the walls of this area relatively flimsy. There have been reports that in children under 8 years old, the pressure of the cuff on the weak endotracheal walls is associated with mucosal injury leading to subglottic stenosis [62]. Subglottic stenosis is essentially the narrowing of the airway due to the build up or scar tissue, which obviously causes airways to be stiffer, creates greater airway resistance and requires the infant exert more energy to breathe. Injury to the mucosa and cilia of the airway damages the system by which hazardous particles including germs are removed from the lungs. These injuries can cause lifelong morbidity. It is commonly held in most NICUs that uncuffed ETTs are best suited to the biomechanical environment of the infant airway.

However, in recent years the superiority of the uncuffed tube has been undermined. Although the PVC/polyurethane material itself is reasonably soft, the angled cut-off tube can be quite sharp. Because it is not prevented from moving laterally by a cuff, this cut-off can scratch the mucosal airway walls and break the cilia. Uncuffed ETTs also have a higher risk of ETT leak around the tube and deliver decreased ventilation pressures. They have also been found to be associated with more intubations in order to find the correct size, and though this data is from the child and not the neonatal population it is likely transferable [62]. With each additional intubation, there is a far higher risk of complication. Additionally, the traditionally taught 'con-

ical' shape of the airway has been called into question, with three recent studies concluding that it is shaped cylindrically, like an adults. A 2018 study found that it may be safe to used cuffed ETTs in children under 3 kgs [63], but more studies need to be done to conclude that cuffed ETTs are biomechanically safer than uncuffed ETTs.

In addition to the question of what design is safer when it comes to tube-airway mechanical interactions, there are also issues regarding the suitability of the current PVC and polyurethane composites materials of the tubes. Due to the polymer porosity, bacterial biofilm growth around and inside the tube is a consistent problem. This biofilm can cause irritation or infection of the surrounding mucosa, inducing swelling and increasing airway resistance. When the mucosal cilia has already been damaged, the problem increases as the immune system can not fully remove any infectious bacteria that drips into the airways. This has been linked to causing ventilator-associated pneumonia, a serious complication with a mortality rate of up to 48% in at risk patients [64]. When the tissue itself has been abraded there is opportunity for the bacteria to purge the layers of the tissue or enter the bloodstream, which may cause severe and possibly life risking infections [65].

5.2 Developments of material design of endotracheal tubes

There is a growing field of research in improving the material design of ETT's to mitigate some of these risks. Up until the 21st century, the primary focus in ETT design was achieving PVC composite bioinertia. The focus of the last 20 years has been bio-activity and antisepticism. The following section summarizes the upgrades of the 21th century, then goes on to introduce some of the modern materials and research concepts that are not necessarily market ready.

5.2.1 Recent history of ETT materials

In the 1950s plasticized PVC was been widely used in medical devices including ETTs, catheters and for blood transfusion. They were ideal thanks to their ductility, toughness, flexibility and translucency. However, these tubes had a short in-body life as over time they changed from a flexible state to being brittle. It was found that this phenomenon was due to the plasticizer leaching out of the tube and into the body. Research determined that this had many negative side effects. Some examples of plasticizers and their side effects include: *dioctylphthalate*

which caused contamination of blood platelets, *organotin* which caused inflammation and necrosis of muscle, and *zinc* which caused protein poisoning. Therefore, high molecular weight plasticizers became the standard as they were found to reduce this leaching and toxicity [66].

PVC continued to be the material of choice for ETTs throughout the later 20th century and polyurethane also became popular as it is highly bio-inert. Many experiments were performed to increase their biocompatibility. These included:

- **Grafting** involves grafting PVC/polyurethane to hydrophilic monomers such as acrylic acid or methacrylic acid in order to decrease the hydrophilicity of the tubes and reduce fluid build up and tube blockage.
- **Plasma treatment** involves using plasma to treat the surface of the PVC/ polyurethane to make it more bio-interactive and therefore less irritable to the body. However, this can sometimes have the unintentional side-effect of increasing opportunity for bacterial growth.[66]

5.2.2 Ongoing developments of ETT materials and design

The focus on modern research is to adapt the ETT materials or design for enhanced anti-microbial properties. Recent research developments and proposals include:

- **Incorporation of catationic steroidal antimicrobials (CSAs)** has been proposed by patent as a means of not only creating an environment where bacteria can not grow, but also because CSAs have been shown to encourage tissue healing. It is proposed that CSA compounds are mixed with the polymeric material and therefore become a part of the ETT structure [67].
- **Polyurethane-Iodine complexes** have been explored due to iodine's proficiency as an antimicrobial agent, and its ability to bind strongly to polyurethane. The new material evoked potent antimicrobial activity against gram-negative and gram-positive bacteria, fungi and viruses. It also inhibited ETT surface biofilm formation. Antimicrobial potency was proportional to iodine binding. The material was tested on rat skin cells and caused no damage. It is yet to be tested clinically [64].
- **A self-cleaning and sterilizing ETT design** has been proposed by patent. It involves combining an ETT and a suction catheter. The suction catheter decreases the tendency of mucus and bacteria to adhere to the inner surface of the ETT. It is proposed that both ETT and catheter surfaces are hydrophobic to repel moisture which would encourage outer surface bacterial growth.

Alternately, it is proposed that the ETT and catheter have an antimicrobial photocatylyst coated lumen and the ETT has a fiberoptic cable mounted in it [68].

5.3 Conclusion

This case study has provided the reader with a brief overview of the practice of endotracheal intubation in neonates. From the perspective of biomechanical interaction, the jury is still out on whether using cuffed or uncuffed tubes carry less risk. In either case, the forces applied to the mucosal cell layer of the airway can cause subglottic stenosis and inflammation which results in airway narrowing. From the biomaterial perspective, we see that ETTs also carry the risk of breeding bacteria which, in combination to the damage caused to the airway cilia, can lead to tissue infection or pneumonia. Thankfully, modern ETTs do not appear to leach dangerous plasticizers into the body as they used to. Hopefully with more research and clinical trials, ETTs may no longer pose a risk of incubating infection, and the cusp discussion may be resolved, leading to more directed research into better ETT design for the reduction of applying adverse forces to the airway.

6 | Case Study Two: Regulations in the Medical Device Industry: The GHTF and The Predicate Problem

So far, this thesis has made very it clear that regulation in the neonatal respiratory medical device industry area is in need of improvements. Unsurprisingly, many of the issues with regulatory standards described in previous sections are also relevant for other types of medical device. On a world scale, additional complexities arise due to the different regulations set by different regulatory States. Therefore, it is not uncommon that a device may pass Standards and be permitted for human use in one country but not another. In this Case Study, we first explore the World Health Organizations attempt to unify Standards across the world via the Global Harmonisation Task force (GHTF). We survey their purpose and what they identify to be important factors of medical device safety assurance. In the second part of this Case Study, we identify an issue that still plagues the medical device industry in spite of the GHTF: *predicate creep*. We survey what *predicate creep* is and examples of how it has caused major disasters for the health of patients and the reputation of the medical device industry.

The GHTF is a limb of the World Health Organization body which was formed in 2003 with the purpose of guiding the development and modification of the medical device regulatory systems of its member states. Most medical devices are used globally, therefore it is in the interest of public health to harmonize Standards. The GHTF was developed under the ideal of there being one unifying legislation across the world, though it is recognised that this ideal can not be met as “one single template will not meet the needs of every country" as some countries have “production facilities that will require goods manufacturing practice and complex quality controls", whilst others “may depend principally on the donation of equipment... and need different policies to protect their population against unsafe and inappropriate technology" [69]. The committee is formed by mainly by representatives from national medical device authorities of member states. Member states directly in-

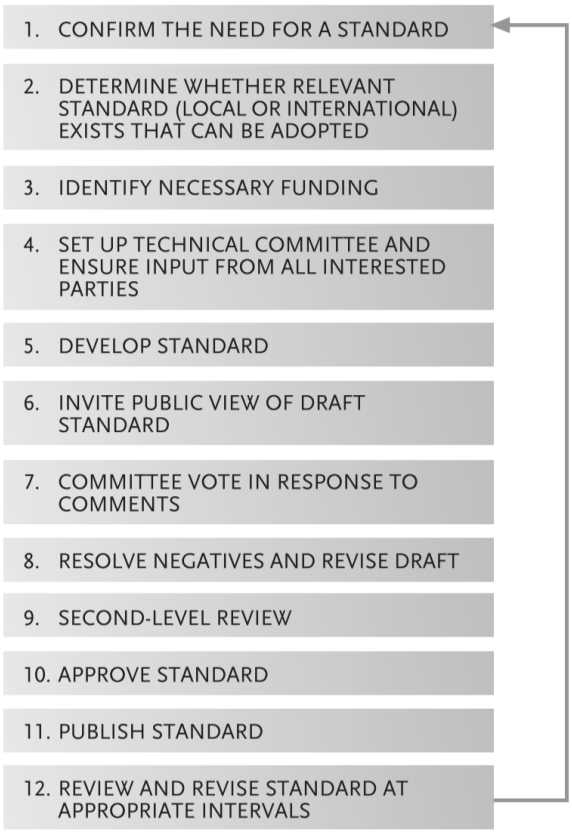


Figure 6.1: Typical Process for Standards Development [69]

volved in orchestrating the decisions of this body are the USA, European Union, Australia, Canada and Japan. Other countries including Russia, China, Brazil and India are not presently part of this body though efforts are being made to align their regulations with those set by the GHTF and ISO. The primary means by which the goals of the GHTF are accomplished are by the publication and dissemination of guidance documents for basic regulatory practices [69]. The GHTF is not the International Organization of Standardization (ISO), but they do work with them to maintain that the standards ensure the safety, reliability and performance of products. ISO Standards are not mandatory in and of themselves, but are made mandatory when they are mandated by a regulatory or legal body. Most countries do treat the ISO Standards as the basis for their own regulations. According to a products conformity to the Standards, the product may display a certification mark allowing it to be market available in many jurisdictions. Figure 6.1 outlines the typical process by which the ISO develops Standards.

As mentioned previously in this thesis, the purpose of regulations are to ensure the safe and effective treatment of patients and to give consumers access to beneficial technology as quickly as possible. According to the WHO document titled *Medical*

Device Regulations: Global overview and guiding principles, medical device safety assurance has several essential elements. These features are: risk management, assessing effectiveness of medical devices, safety across phases in the life span of a medical device, and responsibilities of stakeholders.

Medical device safety and risk management. Device safety can never be guaranteed as inherently all devices carry a degree of risk and are capable of causing harm to a patient or user under the right circumstances. These circumstances may be patient specific. For example, all tests on a pacemaker may point towards it working correctly and being bio-compatible. However, the implantation procedure for pacemakers is high risk as is housing an implant within the body, and the patient may not be able to survive it. The current approach to estimating the potential of a device to be unsafe is by means of a risk assessment. Risk assessments are inherently subjective. The GHTF recommends that therefore device risk assessments are carried out by a panel of medical specialists and other relevant experts, such as biomedical engineers. In many regulatory jurisdictions, medical devices are placed into classes according to its risk assessment. For example, in Australia medical devices are classed into the following; class I (low risk), class I-supplied sterile (low medium risk), class I - incorporating a measuring device (low medium risk), class IIa (medium risk), class IIb (medium high risk), class III (high risk) and Active Implantable Medical Devices (high risk). The United States on the other hand has classes I-III from low to high risk, and humanitarian use devices. The device class guides the stringency of tests set out by the regulatory body that a device must pass before approval [69, 70].

Effectiveness and performance of a medical device. A device is termed *clinically effective* when it "produces the effect indended by the manufacturer relative to the medical condition" [69]. *Performance* includes clinical effectiveness, but also additional features of the device (eg, an alarm) that don't impact patient outcome but do have other uses. Effectiveness and performance are required to ensure medical device safety.

Phases in the life span of a medical device. A medical device goes through the following phases. (1) Conception and Development, (2) Manufacture, (3) Packaging and labelling, (4) Advertising, (5) Sale, (6) Use, and (7) Disposal. At each of these stages, risks must be mitigated to ensure the overall safety of the device. For example, it is important to ensure biohazards are not introduced or can not be introduced during the packaging stage. Similarly, it is important to consider how a device can be disposed of so as to not be an available biohazard that can be touched by waste handlers or the public, nor can it infiltrate

soils and water etc. Advertising and labeling must be clear and truthful to mitigate the risk of improper usage.

The role of each stakeholder. There are three primary stakeholders that play a critical role in ensuring the safety of a medical device. These are the manufacturer, the vendor, and the user. The manufacturer must ensure that the device meets or exceeds the required standards for safety and performance across the development, manufacture and packaging/labeling stage. It is the responsibility of the manufacturer to not only mitigate dangers to the patient, but also the opportunity for accidental 'user error'. It is the responsibility of the vendor to sell the device within the requirements set by the manufacturer (for example, storage time and temperature), and to be truthful in advertising. The vendor is responsible for facilitating after-sale service to maintain the ongoing safety of the device. The user is responsible for ensuring that they have the qualifications and correct conditions (eg, a sterile operating theatre) for usage of the device. They are required to use the device within the manufacturers specifications.

In addition, there are two secondary stake-holders. These are the public and the government. The public are the beneficiary of devices and are responsible for complying with correct device operating procedures. Finally, the government is responsible for setting the jurisdiction for each of the stakeholders. They ensure user safety and protect the liability of the other vendors through law and punishment. They must stay on top of technology as it advances and change the laws appropriately. Ultimately, the responsibility for ensuring medical device safety is shared (see fig. 6.2 [69]).

Though the harmonization of medical device regulations is an important objective, there remain major issues with medical device regulation that must be scrutinized in order to truly protect consumer safety worldwide. Indeed, in recent times there has been increased recall of medical devices [70]. This may be an outcome of more stringent auditing and better reporting practices, but it may also demonstrate more dangerous devices are being permitted for clinical use. One of the major issues is *predicate creep*. *Predicate creep* occurs when a number of devices are considered to meet regulatory standards based on similarity to another device that exists on the market in succession to one another. For example it is possible that by the time ten devices in this series that have passed regulations without testing, there is barely any similarity between the first and the tenth device. Therein lies the danger of *predicate creep* as the final device in that series has essentially never been tested to assure safety. The most famous cases of this phenomenon and its aftermath is

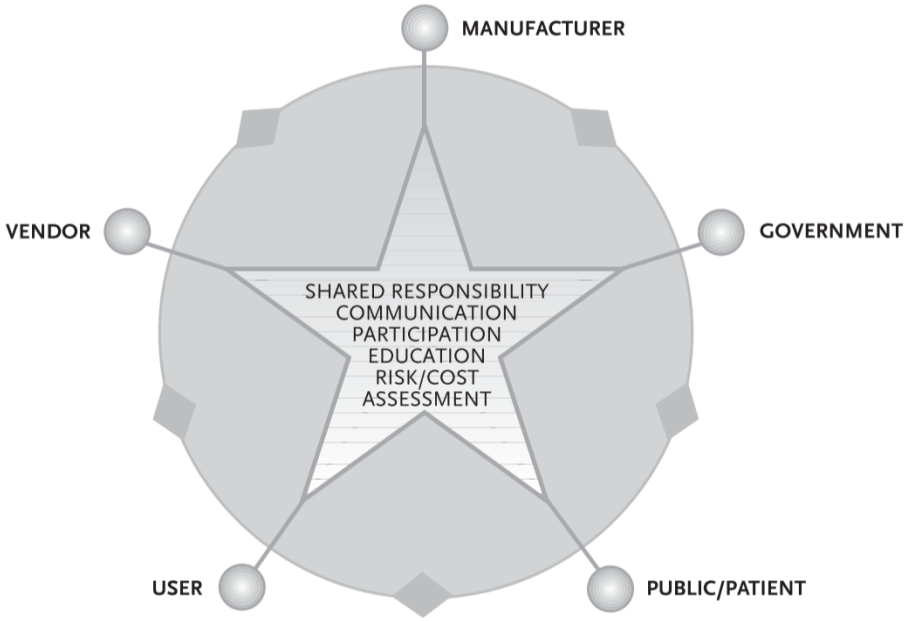


Figure 6.2: Ideal conditions for ensuring the safety and performance of medical devices [69]

that of the vaginal mesh, silicon breast implants and metal-on-metal artificial hip implants. The extremity of these cases was the impetus for changes to regulations that continue to this day. The following paragraphs explore the vaginal mesh case and how it occurred in spite of regulations. This demonstrates why and how methods of medical device regulation and jurisdiction requires drastic improvements.

Transvaginal mesh devices were initially classed as class II under the USA's FDA system and therefore were eligible for approval under the 510(k) system [71]. This system permits vendors to claim their device is "substantially equivalent" to a device already on the market, including up until 2016, recalled devices [72]. In Australia, as in Europe, vendors can apply to have devices approved via proxy of passing the USA's 510(k) system according to what the TGA describe as a "risk based approach" assessed by "experts" in the field [13]. As a result of their low class and the 510(k) process, these devices were implanted in many women for treatment of pelvic organ prolapse and stress urinary incontinence worldwide. Though there is evidence showing that mid-urethral slings, the category under which vaginal meshes fall, are effective, their have been a significant number of serious complications with the vaginal mesh that could have been avoided had they not been allowed to bypass proper regulatory testing. Serious adverse effects include vaginal erosions, infections and organ perforation [71]. The first warning about vaginal meshes was released in 2011, though devices were not recalled from Australia and other gover-

nance members of the GHTF until April 2019.

A study by Mahtani et al. [73] traced the marketing clearance for 61 mesh devices that had gained market access through a chain of equivalence and found that the only two unique originating devices that were made to pass all standards were approved in 1985 and 1996. They found no evidence for any new clinical trial data at the time of device approval for all 61 devices, with empirical evidence of effectiveness from randomised trials emerging on average five years after approval with a range of up to fourteen years. The danger of predicate creep should have first been realised when an earlier device, the Protegan sling which was made from polyester, was recalled from the market. This device was a not only a predicate for more modern devices, but modern devices were not even made from the same material. They were made from polypropylene and yet still passed the 510(k) procedures. It is worth noting that the 510(k) process is not necessarily void of any clinical testing, however, the clinical tests applied to these devices are not rigorous enough. All meshes are required to undergo 48 hours of *in vivo* animal testing to observe for biocompatability and dangerous side effects. Mahtani et al. reflect that "such studies are unlikely to reflect the benefit-harm profile of implantable devices that remain in humans for decades". Additionally, it is uncertain to what extent the movement of a device in an animal such as a rodent is indicative of how it will behave in a human. In spite of the 2016 decision of the FDA to change the class of vaginal meshes from class II to class III in order to force new devices to undergo more thorough testing, the devices that have not been recalled from the market will not be forced to be retested, leaving it to time and findings of patient harm before adverse effects can be identified. Furthermore, a three year transition period is in place before the class III rules come into force which does not expire until May 2020. Clearly, not only do current regulations permit *predicate creep*, they also do not act fast enough on existing adverse events caused by the phenomenon.

In summary, it is clear that while regulation of medical devices is moving towards unification to ensure the same standard of care for patients worldwide, pertinent issues such as *predicate creep* still plague the system. There is no point in harmonisation if the harmonised Standards permit dangerous therapies to be provided to the general public.

7 | Overall Discussion and Conclusions

Is GINA valid enough? What is good enough. Comparison to ASL. Comparison to models. Price factor.
future directions: software Intended tests Timeline

8 | Work Health and Safety Issues

8.1 Overview

The *Work Health and Safety Act, 2011* [20] details the right of an employee or contracted worker to safety in the workplace and the responsibilities and duties of workers and officers in a workplace. Health and safety risks are managed under an approved code in this act, under *Section 274*, named the *Code of Practice: How to Manage Work Health and Safety Risks* [20]. In accordance with this code, all of my undertakings within the workplace at Westmead Hospital have been considered and a risk assessment has been undertaken. The purpose of carrying out a risk assessment is to identify hazards I may encounter in my work, assess their risk, and then to control those risks before pursuing said risky activities. Control measures must be lasting and may require regular assessment to ensure their performance. Hazards may be associated with the physical work environment, equipment and materials used, and work tasks that are physical or experimental in nature.
It has been deemed by myself and safety officers that the nature of the risks involved in my work are acceptable. All actions required to mitigate identified risks have been taken and will continue to be taken as necessary.

8.2 Risk Assessment

Table 8.1: Workplace Risk Assessment

Hazard	**Harm**	**Likelihood**	**Severity**	**Risk**	**Mitigation Method**	**Actioned by**	**Actioned**
Scissors, knives, scalpels	Can slice skin	Unlikely	Negligible injury	Low	Take caution, cut away from body	Corinne Gard	ongoing
Electrical machines	If dysfunc -tional, may cause electrocution	Highly Unlikely	Major injury	Med	Ensure equipment is not beyond service date	Corinne Gard, Murray Hinder	ongoing
Sitting for long hours	Injury to back, hips and/or neck	Likely	Minor injury	Med	Take regular breaks to walk	Corinne Gard	ongoing
Working at computer	Eye strain and/or neck strain	Highly Likely	Minor injury	High	Wear computer glasses, take regular breaks	Corinne Gard	ongoing
Liquid silicon rubber	Solvent in liquid silicon can be dangerous. Accidental ingestion or bio-accumulation by breathing and/or touching has been linked to cancer and endocrine disruption.	Highly Unlikely	Major injury	Med	Wear gloves, keep face away from glue, replace cap after use	Corinne Gard	ongoing

Bibliography

[1] M. O. Edwards, S. J. Kotecha, and S. Kotecha, "Respiratory distress of the term newborn infant," *Paediatric respiratory reviews*, vol. 14, no. 1, pp. 29–37, 2013, accessed: 01-06-2019.

[2] S. N. Wall, A. C. Lee, S. Niermeyer, M. English, W. J. Keenan, W. Carlo, Z. A. Bhutta, A. Bang, I. Narayanan, I. Ariawan *et al.*, "Neonatal resuscitation in low-resource settings: what, who, and how to overcome challenges to scale up?" *International Journal of Gynecology & Obstetrics*, vol. 107, no. Supplement, pp. S47–S64, 2009, accessed: 27-06-2019.

[3] S. B. Hooper, M. L. Siew, M. J. Kitchen, and A. B. Te Pas, "Establishing functional residual capacity in the non-breathing infant," in *Seminars in Fetal and Neonatal Medicine*, vol. 18, no. 6. Elsevier, 2013, pp. 336–343, accessed: 20-06-2019.

[4] P. H. Burri, "Fetal and postnatal development of the lung," *Annual review of physiology*, vol. 46, no. 1, pp. 617–628, 1984, accessed: 13-09-2019.

[5] L. J. Smith, K. O. McKay, P. P. van Asperen, H. Selvadurai, and D. A. Fitzgerald, "Normal development of the lung and premature birth," *Paediatric respiratory reviews*, vol. 11, no. 3, pp. 135–142, 2010, accessed: 28-12-2019.

[6] C. J. Lanteri and P. D. Sly, "Changes in respiratory mechanics with age," *Journal of Applied Physiology*, vol. 74, no. 1, pp. 369–378, 1993, accessed: 28-12-2019.

[7] J. J. Morrison, J. M. Rennie, and P. J. Milton, "Neonatal respiratory morbidity and mode of delivery at term: influence of timing of elective caesarean section," *BJOG: An International Journal of Obstetrics & Gynaecology*, vol. 102, no. 2, pp. 101–106, 1995, accessed: 28-12-2019.

[8] L. Jain and D. C. Eaton, "Physiology of fetal lung fluid clearance and the effect of labor," in *Seminars in perinatology*, vol. 30, no. 1. Elsevier, 2006, pp. 34–43, accessed: 04-11-2019.

[9] G. D. Hankins and M. Speer, "Defining the pathogenesis and pathophysiology of neonatal encephalopathy and cerebral palsy," *Obstetrics & Gynecology*, vol. 102, no. 3, pp. 628–636, 2003, accessed: 27-06-2019.

[10] C. J. Morley and P. G. Davis, "Advances in neonatal resuscitation: supporting transition," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 93, no. 5, pp. F334–F336, 2008, accessed: 24-06-2019.

[11] M. Miedema, F. H. de Jongh, I. Frerichs, M. B. van Veenendaal, and A. H. van Kaam, "Changes in lung volume and ventilation during lung recruitment in high-frequency ventilated preterm infants with respiratory distress syndrome," *The Journal of pediatrics*, vol. 159, no. 2, pp. 199–205, 2011, accessed: 20-12-2019.

[12] M. K. Hinder, "Assessment of devices used to procide positive pressure ventilation for newborns and infants. technological changes, efficacy and regulatory standards." Ph.D. dissertation, The University of Sydney, 2018, accessed: 20-06-2019.

[13] *Therapeutic Goods Administration, regulatory guidelines for medical devices (ARGMD)*, Government of Australia, 2011, accessed: 27-06-2019. [Online]. Available: https://www.tga.gov.au/sites/default/files/essential-principles-checklist-medical-devices.pdf

[14] M. Walker, "Ethics and advanced medical devices: do we need a new approach," *Health Voices*, vol. 21, 2017, accessed: 04-11-2019.

[15] M. Hinder, A. Perdomo, and M. Tracy, "Dangerous pressurization and inappropriate alarms during water occlusion of the expiratory circuit of commonly used infant ventilators," *PloS one*, vol. 11, no. 4, p. e0154034, 2016, accessed: 5-07-2019.

[16] M. B. Tracy, R. Halliday, S. K. Tracy, and M. K. Hinder, "Newborn self-inflating manual resuscitators: precision robotic testing of safety and reliability," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 104, no. 4, pp. F403–F408, 2019, accessed: 03-08-2019.

[17] T. J. Krieger and M. Wald, "Volume-targeted ventilation in the neonate: benchmarking ventilators on an active lung model," *Pediatric Critical Care Medicine*, vol. 18, no. 3, pp. 241–248, 2017.

[18] T. Itagaki, C. T. Chenelle, D. J. Bennett, D. F. Fisher, and R. M. Kacmarek, "Effects of leak compensation on patient-ventilator synchrony during prema-

ture/neonatal invasive and noninvasive ventilation: a lung model study," *Respiratory care*, vol. 62, no. 1, pp. 22–33, 2017.

[19] M. Hinder, personal communication.

[20] "Work health and safety act 2011," accessed: 20-06-2019.

[21] P. Coffey, L. Kak, I. Narayanan, J. Bergeson Lockwood, N. Singhal, S. Wall, J. Johnson, and E. Schoen, "Newborn resuscitation devices," *United Nations Commission on Life-Saving Commodities for Women and Children*, 2012, accessed: 03-08-2019.

[22] W. H. Organization *et al.*, "Guidelines on basic newborn resuscitation," 2012, accessed: 03-08-2019.

[23] N. Claure and E. Bancalari, "New modes of mechanical ventilation in the preterm newborn: evidence of benefit," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 92, no. 6, pp. F508–F512, 2007, accessed: 7-09-2019.

[24] H. C. Glass, A. T. Costarino, S. A. Stayer, C. Brett, F. Cladis, and P. J. Davis, "Outcomes for extremely premature infants," *Anesthesia and analgesia*, vol. 120, no. 6, p. 1337, 2015, accessed: 3-01-2020.

[25] U. Frey, J. Stocks, A. Coates, P. Sly, and J. Bates, "Specifications for equipment used for infant pulmonary function testing. ers/ats task force on standards for infant respiratory function testing. european respiratory society/american thoracic society," *European Respiratory Journal*, vol. 16, no. 4, pp. 731–740, 2000, accessed: 16-06-2019.

[26] T. Drevhammer, personal communication.

[27] M. Tracy, personal communication.

[28] E. E. Foglia, L. S. Owen, M. Thio, S. J. Ratcliffe, G. Lista, A. te Pas, H. Hummler, V. Nadkarni, A. Ades, M. Posencheg *et al.*, "Sustained aeration of infant lungs (sail) trial: study protocol for a randomized controlled trial," *Trials*, vol. 16, no. 1, p. 95, 2015, accessed: 01-06-2019.

[29] E. M. Sivieri, K. Dysart, and S. Abbasi, "Evaluation of pulmonary function in the neonate," in *Fetal and Neonatal Physiology*. Elsevier, 2017, pp. 754–765, accessed: 3-01-2020.

[30] B. Martini, Nath, *Fundamentals of Anatomy and Physiology*. Pearson Education Limited, 2014, vol. 9, accessed: 13-09-2019.

[31] J.-O. Dunn, M. Mythen, and M. Grocott, "Physiology of oxygen transport," *Bja Education*, vol. 16, no. 10, pp. 341–348, 2016, accessed: 01-11-2019.

[32] M. Ochs, J. R. Nyengaard, A. Jung, L. Knudsen, M. Voigt, T. Wahlers, J. Richter, and H. J. G. Gundersen, "The number of alveoli in the human lung," *American journal of respiratory and critical care medicine*, vol. 169, no. 1, pp. 120–124, 2004, accessed: 04-11-2019.

[33] "Histology study guide: Respiratory tract," http://www.siumed.edu/~dking2/crr/rsguide.htm, accessed: 6-01-2020.

[34] D. M. Shade, *Design of Respiratory Devices*. Johns Hopkins School of Medicine, accessed: 24-06-2019.

[35] F. P. Primerio Jr, *Measurements of the respiratory system*, 2008.

[36] C. M. Ionescu, *The Human Respiratory System: An Analysis of the Interplay between Anatomy, Structure, Breathing and Fractal Dynamics*. Springer, ch. The Human Respiratory System, pp. 13–22.

[37] B. Suki, A.-L. Barabasi, and K. R. Lutchen, "Lung tissue viscoelasticity: a mathematical framework and its molecular basis," *Journal of Applied Physiology*, vol. 76, no. 6, pp. 2749–2759, 1994, accessed: 6-01-2020.

[38] J. J. Fredberg and D. Stamenovic, "On the imperfect elasticity of lung tissue," *Journal of applied physiology*, vol. 67, no. 6, pp. 2408–2419, 1989, accessed: 6-01-2020.

[39] N. Khajeh-Hosseini-Dalasm and P. W. Longest, "Deposition of particles in the alveolar airways: inhalation and breath-hold with pharmaceutical aerosols," *Journal of aerosol science*, vol. 79, pp. 15–30, 2015, accessed: 04-11-2019.

[40] R. P. Neumann and B. S. von Ungern-Sternberg, "The neonatal lung–physiology and ventilation," *Pediatric Anesthesia*, vol. 24, no. 1, pp. 10–21, 2014, accessed: 10-8-2019.

[41] A.-F. Hoo, S. Lum, J. Mattes, and J. Stocks, "Manual of infant lung function tests," 2014, accessed: 18-06-2019.

[42] S. B. Hooper, A. B. te Pas, and M. J. Kitchen, "Respiratory transition in the newborn: a three-phase process," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 101, no. 3, pp. F266–F271, 2016, accessed: 20-06-2019.

[43] "Intubation." [Online]. Available: https://www.bettersafercare.vic.gov.au/resources/clinical-guidance/maternity-and-newborn/intubation#goto-endotracheal-tube-ett-size

[44] S. Cecchini, E. Schena, and S. Silvestri, "An open-loop controlled active lung simulator for preterm infants," *Medical engineering & physics*, vol. 33, no. 1, pp. 47–55, 2011, accessed: 16-06-2019.

[45] P. Cappa, S. A. Sciuto, and S. Silvestri, "A novel preterm respiratory mechanics active simulator to test the performances of neonatal pulmonary ventilators," *Review of scientific instruments*, vol. 73, no. 6, pp. 2411–2416, 2002, accessed: 16-06-2019.

[46] U. Frey, B. Reinmann, and J. Stocks, "The infant lung function model: a mechanical analogue to test infant lung function equipment," *European Respiratory Journal*, vol. 17, no. 4, pp. 755–764, 2001, accessed: 16-06-2019.

[47] R. T. Scaramuzzo, M. Ciantelli, I. Baldoli, L. Bellanti, M. Gentile, F. Cecchi, E. Sigali, S. Tognarelli, P. Ghirri, S. Mazzoleni *et al.*, "Mechatronic respiratory system simulator for neonatal applications (meressina) project: a novel bioengineering goal," *Medical devices (Auckland, NZ)*, vol. 6, p. 115, 2013, accessed: 18-06-2019.

[48] B. Stankiewicz, K. J. Pałko, M. Darowski, K. Zieliński, and M. Kozarski, "A new infant hybrid respiratory simulator: preliminary evaluation based on clinical data," *Medical & biological engineering & computing*, vol. 55, no. 11, pp. 1937–1948, 2017, accessed: 18-06-2019.

[49] N. McCallion, R. Lau, C. Morley, and P. Dargaville, "Neonatal volume guarantee ventilation: effects of spontaneous breathing, triggered and untriggered inflations," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 93, no. 1, pp. F36–F39, 2008, accessed: 7-07-2019.

[50] M. Obladen, "History of neonatal resuscitation–part 1: Artificial ventilation," *Neonatology*, vol. 94, no. 3, pp. 144–149, 2008, accessed: 7-07-2019.

[51] G. A. Gregory, J. A. Kitterman, R. H. Phibbs, W. H. Tooley, and W. K. Hamilton, "Treatment of the idiopathic respiratory-distress syndrome with continuous positive airway pressure," *New England Journal of Medicine*, vol. 284, no. 24, pp. 1333–1340, 1971, accessed: 10-09-2019.

[52] N. McCallion, R. Lau, P. Dargaville, and C. J. Morley, "Volume guarantee ventilation, interrupted expiration, and expiratory braking," *Archives of disease in childhood*, vol. 90, no. 8, pp. 865–870, 2005, accessed: 7-09-2019.

[53] C. Klingenberg, K. I. Wheeler, P. G. Davis, and C. J. Morley, "A practical guide to neonatal volume guarantee ventilation," *Journal of Perinatology*, vol. 31, no. 9, p. 575, 2011, accessed: 10-09-2019.

[54] "Medical electrical equipment. Part 2-12: Particular requirements for basic safety and essential performance of critical care ventilators," International Organization for Standardization, Geneva, CH, Standard, accessed: 5-07-2019.

[55] M. Hinder, A. McEwan, T. Drevhammer, S. Donaldson, and M. B. Tracy, "T-piece resuscitators: how do they compare?" *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 104, no. 2, pp. F122–F127, 2019, accessed: 27-06-2019.

[56] D. P. Schaller, *Instruction manual Gina V3.0*, accessed: 20-06-2019.

[57] "Pneumotach accuracy," https://www.pftforum.com/blog/pneumotach-accuracy/, accessed: 6-01-2020.

[58] *User Manual FlowAnalyser*, imtmedical, accessed: 20-06-2019.

[59] M. Jackson, S. Crouch, and R. Baxter, "Software evalution: Criteria-based assessment," November 2011. [Online]. Available: https://software.ac.uk/sites/default/files/SSI-SoftwareEvaluationCriteria.pdf?_ga=2.57488931.843308093.1579298888-876001691.1578153944

[60] R. M. Strong and V. Passy, "Endotracheal intubation: complications in neonates," *Archives of Otolaryngology*, vol. 103, no. 6, pp. 329–335, 1977.

[61] L. D. Hatch, P. H. Grubb, A. S. Lea, W. F. Walsh, M. H. Markham, G. M. Whitney, J. C. Slaughter, A. R. Stark, and E. W. Ely, "Endotracheal intubation in neonates: a prospective study of adverse safety events in 162 infants," *The Journal of pediatrics*, vol. 168, pp. 62–66, 2016, accessed: 5-07-2019.

[62] R. Thomas, S. Rao, and C. Minutillo, "Cuffed endotracheal tubes for neonates and young infants: a comprehensive review," *Archives of Disease in Childhood-Fetal and Neonatal Edition*, vol. 101, no. 2, pp. F168–F174, 2016.

[63] R. E. Thomas, S. C. Rao, C. Minutillo, B. Hullett, and M. K. Bulsara, "Cuffed endotracheal tubes in infants less than 3 kg: A retrospective cohort study," *Pediatric Anesthesia*, vol. 28, no. 3, pp. 204–209, 2018.

[64] J. Luo, Y. Deng, and Y. Sun, "Antimicrobial activity and biocompatibility of polyurethane—iodine complexes," *Journal of Bioactive and Compatible Polymers*, vol. 25, no. 2, pp. 185–206, 2010.

[65] K. B. Zur, D. L. Mandell, R. E. Gordon, I. Holzman, and M. A. Rothschild, "Electron microscopic analysis of biofilm on endotracheal tubes removed from intubated neonates," *Otolaryngology—Head and Neck Surgery*, vol. 130, no. 4, pp. 407–417, 2004.

[66] J. Singh and K. K. Agrawal, "Modification of poly (vinyl chloride) for biocompatibility improvement and biomedical application-review," *Polymer-Plastics Technology and Engineering*, vol. 31, no. 3-4, pp. 203–212, 1992.

[67] C. Genberg, P. B. Savage, and R. L. Bracken, "Novel endotracheal tube for the reduction of intubation-related complication in neonates and babies," Mar. 23 2017, uS Patent App. 15/270,876.

[68] C. P. Rao and D. C. Lister, "Self-cleaning and sterilizing endotracheal and tracheostomy tube," Feb. 26 2013, uS Patent 8,381,728.

[69] M. Cheng, *Medical device regulations: global overview and guiding principles*. World Health Organization, 2003.

[70] S. K. Gupta, "Medical device regulations: A current perspective." *Journal of Young Pharmacists*, vol. 8, no. 1, 2016.

[71] C. J. Heneghan, B. Goldacre, I. Onakpoya, J. K. Aronson, T. Jefferson, A. Pluddemann, and K. R. Mahtani, "Trials of transvaginal mesh devices for pelvic organ prolapse: a systematic database review of the us fda approval process," *BMJ open*, vol. 7, no. 12, p. e017125, 2017.

[72] S. P. Ho, "Fixing a 510 (k) loophole: In support of the sound devices act of 2012," 2014.

[73] C. Heneghan, J. K. Aronson, B. Goldacre, K. R. Mahtani, A. Plüddemann, and I. Onakpoya, "Transvaginal mesh failure: lessons for regulation of implantable devices," *Bmj*, vol. 359, p. j5515, 2017.

A | Appendix for Core Study One

A.1 Code for GINA VALIDATOR

```
%FILEVALIDATOR MATLAB CODE
%       This file FileValidator creates the GUI application with which the
%       user interfaces in order to validate GINA data against
%       simultaneously taken data from the FlowAnalyser. Based on user
%       selections and input, the data is sent to the main.m file for
%       processing.


%-- function not my own work. Created through GUIDE application.
function varargout = FileValidator(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FileValidator_OpeningFcn, ...
                   'gui_OutputFcn',  @FileValidator_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before FileValidator is made visible. Not my own work.
function FileValidator_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for FileValidator
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line. Not my
% own work
function varargout = FileValidator_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

%-- The following functions were created as a result of the GUIDE app
%design I created

% --- Executes during object creation, after setting all properties.
function gina_filename_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function lab_Filename_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press to reset the gui settings
function pushbutton7_Callback(hObject, eventdata, handles)
resetgui(hObject, eventdata, handles, 1);

% --- Executes on button press in ginabrowse. Browses for appropriate NALM
% xlsx file in the directory 'Tests_data' if it exists in the current
% directory.
function ginabrowse_Callback(hObject, eventdata, handles)
```

```
65 currentFolder = pwd;
66 filename = fullfile( currentFolder, 'Tests_data', '*.xlsx');
67 global file_nalm
68 file_nalm = uigetfile({filename}, 'GINA File Selector');
69 set(handles.gina_filename,'String', file_nalm);
70 out = filegetter(handles, get(handles.gina_filename, 'String'), filename, file_nalm, handles.gina_filename);
71 file_nalm = out;
72 set(handles.gina_filename,'String', file_nalm);
73
74 % --- Executes on button press in labbrowse. This function browses for an
75 % appropriate FlowAnalyser file in the directory 'Tests_Data', if it exists
76 % in the current directory.
77 function labbrowse_Callback(hObject, eventdata, handles)
78 currentFolder = pwd;
79 filename = fullfile( currentFolder, 'Tests_Data', '*.log');
80 global file_lab
81
82 file_lab = uigetfile({filename}, 'FlowLab File Selector');
83 set(handles.lab_Filename, 'String',file_lab);
84 out = filegetter(handles, get(handles.lab_Filename, 'String'), filename, file_lab, handles.lab_Filename);
85 file_lab = out;
86
87 %-- edit5 function creates textbox for validation answer in GUI
88 function edit5_Callback(hObject, eventdata, handles)
89
90 % --- Executes during object creation, after setting all properties.
91 function edit5_CreateFcn(hObject, eventdata, handles)
92
93 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
94     set(hObject,'BackgroundColor','white');
95 end
96
97 %-- nameFile box is where the user writes the new file name
98 function nameFile_Callback(hObject, eventdata, handles)
99
100 % --- Executes during object creation, after setting all properties.
101 function nameFile_CreateFcn(hObject, eventdata, handles)
102
103 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
104     set(hObject,'BackgroundColor','white');
105 end
106
107 % --- Executes on button press in helpmenu. Brings up PDF of the manual.
108 function helpmenu_Callback(hObject, eventdata, handles)
109
110 %open HELP file
111 open GINA-VALIDATOR-MANUAL.pdf
112
113 % --- Executes on button press in 'GO'
114 function pushbutton4_Callback(hObject, eventdata, handles)
115 global nameFile
116 check = 0;
117 while check == 0
118     a = get(handles.nameFile, 'String');
119     % first check that a file name has been provided
120     f = outfilecheck(a, handles);
121     %Then check if file exists
122     currentFolder = pwd;
123     File = fullfile(currentFolder, 'Tests_Results', string(f(1)));
124
125     if exist(File, 'dir')
126         str = ['File name " ', File, ' " already exists. Do you want to overwrite this file?'];
127         newStr = join(str);
128         answer = questdlg(newStr, 'Overwrite file?', 'Yes', 'No', 'No');
```

```
129         switch answer
130             case 'Yes'
131                 nameFile = string(f(1));
132                 check = 1;
133             case 'No'
134                 set(handles.nameFile, 'string', 'Edit Text');
135                 newFile = inputdlg('Enter a new file name, no spaces: ', 'Output File Name');
136                 set(handles.nameFile,'String', newFile);
137         end
138 
139     else
140         nameFile = string(f(1));
141         check = 1;
142     end
143 
144 end
145 %Error handling for if a valid file has not been selected
146 x = get(handles.lab_Filename, 'String');
147 y = get(handles.gina_filename, 'String');
148 if strcmp(y, '0') || strcmp(y,'GINA File') || strcmp(x,'0') || strcmp(x, 'FlowAnalyser File')
149       warndlg('FileValidator can not run without a valid file selection. Select a file and click "Go" again');
150       resetgui(hObject, eventdata, handles, 0)
151 else
152     main %runs main file
153 end
154 
155 %-- outfilecheck function ensures that an appropriate outfile name has been
156 %provided by the user
157 function a = outfilecheck(f, handles)
158     if f == "Edit Text"
159         uiwait(warndlg('Please enter an output file name'));
160         newFile = inputdlg('Enter a new file name, no spaces: ', 'Output File Name');
161         set(handles.nameFile,'String', newFile);
162         g = get(handles.nameFile, 'String');
163         a = outfilecheck(g, handles);
164     else
165         a = f;
166         return
167     end
168 
169 %-- This function checks that the user has selected a file and handles
170 %the associated errors.
171 function out = filegetter(handles, filestring, filename, setfile, releventhandle) %checks that a file has been selected
172     if filestring ~= '0'
173         out = setfile;
174         return
175     else
176     % set a question
177     answer = questdlg('You must select one file. Click "Select File", or "Cancel" to exit.', 'Choose a file', 'Select File', 'Cancel', ↙
'Select File');
178     switch answer
179         case 'Select File'
180             setfile = uigetfile({filename}, 'File Selector');
181             set(releventhandle,'String', setfile);
182             next = get(releventhandle, 'String');
183             out = filegetter(handles, next, filename, setfile, releventhandle);
184         case 'Cancel'
185             out = '0';
186     end
187     end
188 
189 
190 
191 %-- resetgui function restores the gui to its original settings
```

```
192 function resetgui(hObject, eventdata, handles, erasename)
193 set(handles.gina_filename, 'string', 'GINA File');
194 set(handles.lab_Filename, 'string', 'FlowAnalyser File');
195 set(handles.edit5, 'string', '');
196 if erasename == 1
197     set(handles.nameFile, 'string', 'Edit Text');
198 end
199
```

```
%MAIN MATLAB CODE
%       This file ga_main is called by FileValidator when 'GO' is selected
%       on the FileValidator GUI. In this file, output data from GINA, and
%       from the FlowAnalyser are sent to undergo analysis in Tester.m.
%       Files are read in then timestamps are checked to ensure the signal
%       recordings started within 30s of each other. Depending on
%       whether Tester determines that the GINA data is valid, the Main
%       file determines the message to be displayed by the FileValidator
%       app.

% --Global variables
   global nameFile;
   global errorMessage;
   global file_lab
   global file_nalm

% --Import data
    % FlowLab data
    % 1) Get date and time stamp
    filename1 = fullfile('Tests_Data', file_lab);
    fid = fopen(filename1);
        i = 1;
        while (i <= 2)
            A = fgetl(fid);
            i= i + 1;
        end
        fclose(fid);
        B = split(A,'=');
        T_l = datevec(datetime(B{2})); %get date vector
     % 2) Read continuous data into a table
        lab_Data = readtable(filename1, 'FileType', 'text');
        lab_Data  = removevars(lab_Data, {'FlowHigh_l_min_'});
        lab_Data.Properties.VariableNames = {'Time_lab', 'Flow_lab', 'P_diff_lab', 'Vol_tid_lab'};
        for i = 1:height(lab_Data)
            z = (-1)*lab_Data{i,3};% invert pressure signal to match the data from GINA
            lab_Data{i,3} = z;
        end

    % GINA data
    % 1) Get date and time
    filename2 = fullfile('Tests_Data', file_nalm);
    [~,sheet_name]=xlsfinfo(filename2);
    [~, text] = xlsread(filename2, sheet_name{1}, 'G2');
    T_n = datevec(datetime(text{1,1},'InputFormat','dd/MM/yyyy h:mm:ss a'));
    % 2) Read continuous data into a table
    nalm_Data = readtable(filename2,'Sheet', sheet_name{4});
    nalm_Data = removevars(nalm_Data, {'Prm', 'Palv', 'Ptr', 'Vol', 'ChX'});
    nalm_Data.Properties.VariableNames = {'Time_nalm', 'P_diff_nalm', 'Flow_nalm', 'Vol_tid_nalm'};
    nalm_Data = movevars(nalm_Data, 'P_diff_nalm', 'After', 'Flow_nalm');

%--Run validity tests
test1 = Tester(nalm_Data, lab_Data, T_n, T_l, nameFile);
if test1.ex == 1
    uiwait(msgbox('Time interval between files can not be longer than 15s. Re-enter data and restart FileValidator'));
    return
end

%-- Set validity status message on GUI
if test1.valid == 0
    errorMessage = 'Validation failed';
else
    errorMessage = 'Validation passed';
end
myhandle = findobj('Tag','edit5');
```

```
65 set(myhandle, 'String',errorMessage);
66 p = peakPlotter(test1);
67 myhandle1 = findobj('Tag','edit6');
68 set(myhandle1, 'String',p);
69
70 close all
```

```
% TESTER MATLAB code for the GINA_VALIDATOR GUI.
%       This file Tester runs signal analysis on the continuous data from
%       GINA and FlowAnalyser. It takes in the data signals, trims them to
%       have the same start, scales data to correct for sample rate error
%       and uses Bland Altman analysis to determine signal validity.

classdef Tester < handle

    properties
        sam_rate = .05 %data sampl
        %Errors for FlowLab
        erFLflow = .05 %(sl/min),
        erFLvol = .01 %sl
        erFLdp = .1 %mbar
        erFLfreq = 1 %bpm
        erFLcomp = 1 %ml/mbar
        %Errors for GINA
        erFlow = .17
        erVol = .17
        erP = .2
        currentFolder = pwd;
        folderName
        newFile
        size
        x1
        x2
        x0
        y2p
        y2v
        int
        vent
        nalm_Data
        lab_Data
        T_n
        T_l
        scale_fac
        valid = 1; %changes to 0 if test is not valid
        ex
        period = 1; %for testing purposes only. Change period if different freq
        peaksTable
        pMaxGINA
        pMinGINA
        fMaxGINA
        fMinGINA
        pMaxFA
        pMinFA
        fMaxFA
        fMinFA
    end

    methods
        %--- Initialisation function for the class
        function obj = Tester(nalm_Data, lab_Data, T_n, T_l, folderName)
            obj.nalm_Data = nalm_Data;
            obj.lab_Data = lab_Data;
            obj.T_n = T_n;
            obj.T_l = T_l;
            obj.folderName = string(folderName);

            file = fullfile('Tests_Results', string(folderName));
            mkdir (file) %destination folder for saved files

            fileTrimmer(obj);
            if obj.ex == 1
```

```
65              return
66          end
67          setFile(obj);
68
69          %here, plotting flow before delay is calculated. Comment out
70          %when using the software.
71          %{
72          variableCompare(obj); flowPlotter(obj);
73          %}
74
75          delayCalc(obj);
76          labDataScaler(obj);
77          variableCompare(obj);
78          flowPlotter(obj);
79          pressurePlotter(obj);
80
81          %populate peaksTable and save to file
82          length(obj.fMaxGINA)
83          length(obj.fMaxFA)
84          length(obj.fMinGINA)
85          length(obj.fMinFA)
86          length(obj.pMaxGINA)
87          length(obj.pMaxFA)
88          length(obj.pMinGINA)
89          length(obj.pMinFA)
90
91          %trim data to the same length
92          while length(obj.fMaxGINA) > length(obj.fMinGINA)
93              obj.fMaxGINA(length(obj.fMaxGINA)) = [];
94          end
95          while length(obj.fMaxGINA) < length(obj.fMinGINA)
96              obj.fMinGINA(length(obj.fMaxGINA)) = [];
97          end
98          while length(obj.pMaxGINA) > length(obj.pMinGINA)
99              obj.pMaxGINA(length(obj.pMaxGINA)) = [];
100         end
101         while length(obj.pMaxGINA) < length(obj.pMinGINA)
102             obj.pMinGINA(length(obj.pMaxGINA)) = [];
103         end
104         while length(obj.fMaxGINA) > length(obj.pMaxGINA)
105             obj.fMaxGINA(length(obj.fMaxGINA)) = [];
106             obj.fMinGINA(length(obj.fMaxGINA)) = [];
107         end
108         while length(obj.fMaxGINA) < length (obj.pMaxGINA)
109              obj.pMaxGINA(length(obj.pMaxGINA)) = [];
110              obj.pMinGINA(length(obj.pMaxGINA)) = [];
111         end
112
113         while length(obj.fMaxFA) > length(obj.fMinFA)
114             obj.fMaxFA(length(obj.fMaxFA)) = [];
115         end
116         while length(obj.fMaxFA) < length(obj.fMinFA)
117             obj.fMinFA(length(obj.fMaxFA)) = [];
118         end
119         while length(obj.pMaxFA) > length(obj.pMinFA)
120             obj.pMaxFA(length(obj.pMaxFA)) = [];
121         end
122         while length(obj.pMaxFA) < length(obj.pMinFA)
123             obj.pMinFA(length(obj.pMaxFA)) = [];
124         end
125         while length(obj.fMaxFA) > length(obj.pMaxFA)
126             obj.fMaxFA(length(obj.fMaxFA)) = [];
127             obj.fMinFA(length(obj.fMaxFA)) = [];
128         end
```

```
129         while length(obj.fMaxFA) < length (obj.pMaxFA)
130             obj.pMaxFA(length(obj.pMaxFA)) = [];
131             obj.pMinFA(length(obj.pMaxFA)) = [];
132         end
133
134         while length(obj.fMaxFA)> length(obj.fMaxGINA)
135             obj.fMaxFA(length(obj.fMaxFA)) = [];
136             obj.fMinFA(length(obj.fMinFA))= [];
137             obj.pMaxFA(length(obj.pMaxFA)) = [];
138             obj.pMinFA(length(obj.pMinFA))= [];
139         end
140         while length(obj.fMaxFA)< length(obj.fMaxGINA)
141             obj.fMaxGINA(length(obj.fMaxGINA)) = [];
142             obj.fMinGINA(length(obj.fMinGINA))= [];
143             obj.pMaxGINA(length(obj.pMaxGINA)) = [];
144             obj.pMinGINA(length(obj.pMinGINA))= [];
145         end
146
147         % Designing peaksTable
148         header = {'fMaxGINA', 'fMinGINA', 'pMaxGINA', 'pMinGINA', 'fMaxFA', 'fMinFA', 'pMaxFA', 'pMinFA'};
149         %size of table
150         out = zeros(length(obj.fMaxGINA), 8);
151         for i = 1:length(obj.fMaxGINA)
152             out(i,1)= obj.fMaxGINA(i);
153             out(i,2)= obj.fMinGINA(i);
154             out(i,3)= obj.pMaxGINA(i);
155             out(i,4)= obj.pMinGINA(i);
156             out(i,5)= obj.fMaxFA(i);
157             out(i,6)= obj.fMinFA(i);
158             out(i,7)= obj.pMaxFA(i);
159             out(i,8)= obj.pMinFA(i);
160         end
161
162          k =  array2table(out, 'VariableNames', header);
163          table = fullfile(file, 'Flow_Pres_Max_Mins.xlsx');
164          writetable(k, table);
165     end
166
167     %-- variableCompare function prepares data for comparison
168     function variableCompare(obj)
169
170         seconds = 40;
171         obj.int = seconds/obj.sam_rate;
172
173         obj.x1 = obj.newFile{1:obj.int,1};
174         obj.x2 = obj.newFile{1:obj.int,5};
175         obj.x0 = 0:0.01:seconds; %Prepares equally dispersed time signal for signal interpolation later
176     end
177
178     %-- flowPlotter function plots flow signals against one another
179     function flowPlotter(obj)
180         y1 = obj.newFile{1:obj.int,2};
181         y2 = obj.newFile{1:obj.int,6};
182         y10 = lowpass(interp1(obj.x1,y1,obj.x0), 0.0001); % filtering and interpolating flowanalyser signal
183         y20 = lowpass(interp1(obj.x2, y2, obj.x0), 0.0001); % filtering and interpolating gina signal
184         bigG= max(y20);
185         bigF = max(y10);
186         %find peaks
187         [obj.fMaxGINA, ~] = findpeaks(y10, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigG*2/3);
188         [obj.fMaxFA, ~ ] = findpeaks(y20, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigF*2/3);
189         y01 = -1*y10;
190         y02 = -1*y20;
191         bigG= max(y02);
192         bigF = max(y01);
```

```
193         [g, ~] = findpeaks(y01, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigG*2/3);
194         [fa, ~] = findpeaks(y02, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigF*2/3);
195         obj.fMinGINA = -1*g;
196         obj.fMinFA = -1*fa;
197
198         figure
199         plot(obj.x0, y10, 'linewidth', 2)
200         set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
201               'FontName', 'Times New Roman');
202         title("Flow Comparison test (filtered)")
203         xlabel("Time (s)")
204         ylabel("Flow (l/min)")
205         hold on
206         plot(obj.x0, y20)
207         lgd = legend({"FlowAnalyser", "GINA"});
208         lgd.FontSize = 24;
209            lgd.FontName = 'Times New Roman';
210         hold off
211
212         filename1 = fullfile(obj.currentFolder, 'Tests_Results',obj.folderName, 'flowTest.fig');
213         savefig(filename1);
214         % Bland-Altman analysis
215         [means,diffs,meanDiff,CR,linFit] = BlandAltman(y10,y20,3, 'Flow'); %decide how you'll factor in error
216         if abs(meanDiff)> obj.erFlow
217            obj.valid = 0;
218         end
219         for i = 1:length(means)
220            if  means(i)*linFit(1)+linFit(2) > CR(1) ||  means(i)*linFit(1)+linFit(2) < CR(2)
221                  obj.valid = 0;
222            end
223         end
224         filename2 = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'Bland-Altman_flow.fig');
225         savefig(filename2);
226         %method1-method2 relationship
227         a = min(y10);
228         b = min(y20);
229         if a<b
230            c = a;
231         else
232            c = b;
233         end
234         mi = floor(c - 1);
235         d = max(y10);
236         e = max(y20);
237         if d>e
238            f = d;
239         else
240            f= e;
241         end
242         ma = ceil(f);
243         figure()
244         x = [mi: 0.05: ma];
245         y = x;
246         plot(y10, y20, 'ob', x, y, '-k', 'linewidth', 2)
247         set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
248               'FontName', 'Times New Roman');
249         title('FlowLab vs GINA for flow');
250         xlabel('GINA')
251         ylabel('FlowLab');
252         lgd = legend('FL vs GINA', 'x=y');
253         lgd.FontSize = 24;
254            lgd.FontName = 'Times New Roman';
255         filename3 = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'FLvsGINA_Flow.fig');
256         savefig(filename3);
```

```
    end

    %-- pressurePlotter function plots pressure signals against one
    %another
    function pressurePlotter(obj)
        y1 = -1*obj.newFile{1:obj.int, 3};
        y2 = obj.newFile{1:obj.int, 7};
        y10 = lowpass(interp1(obj.x1,y1,obj.x0), 0.0001);
        obj.y2p = lowpass(interp1(obj.x2, y2, obj.x0), 0.0001);
      %presscorr = corr(y10, obj.y2p) %uncomment to print correlation
        bigG= max(obj.y2p);
        bigF = max(y10);
      %find peaks
        [obj.pMaxGINA, ~] = findpeaks(y10, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigG*2/3);
        [obj.pMaxFA, ~ ] = findpeaks(obj.y2p, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigF*2/3 );
        y01 = -1*y10;
        y02 = -1*obj.y2p;
        bigG= max(y02);
        bigF = max(y01);
        [g, ~] = findpeaks(y01, 'MinPeakDistance', obj.period*2/3,'MinPeakHeight', bigG*2/3 );
        [fa, ~] = findpeaks(y02, 'MinPeakDistance', obj.period*2/3, 'MinPeakHeight', bigF*2/3);
        obj.pMinGINA = -1*g;
        obj.pMinFA = -1*fa;

        figure
        plot(obj.x0, y10, 'linewidth', 2)
        set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
              'FontName', 'Times New Roman');
        title("Pressure comparison test (Filtered)")
        xlabel("Time (s)")
        ylabel("Pressure (mbar)")
        hold on
        plot(obj.x0, obj.y2p)
        lgd = legend({"FlowAnalyser", "GINA"});
        lgd.FontSize = 24;
          lgd.FontName = 'Times New Roman';
        hold off
        filename = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'pressureTest.fig');

        savefig(filename);

        % Bland-Altman part
        [means,diffs,meanDiff,CR,linFit] = BlandAltman(y10,obj.y2p,3, 'Pressure');
        if abs(meanDiff)> obj.erP %if meandiff is more than acceptable error
          obj.valid = 0;
        end
        for i = 1:length(means)
          if  means(i)*linFit(1)+linFit(2) > CR(1) ||  means(i)*linFit(1)+linFit(2) < CR(2)
                obj.valid = 0;
          end
        end
        filename2 = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'Bland-Altman_pressure.fig');
        savefig(filename2);

        %method1-method2 relationship
        a = min(y10);
        b = min(obj.y2p);
        if a<b
          c = a;
        else
          c = b;
        end
        mi = floor(c - 1);
        d = max(y10);
```

```
        e = max(obj.y2p);
        if d>e
            f = d;
        else
            f= e;
        end
        ma = ceil(f);

        figure()
        x = [mi: 0.05: ma];
        y = x;

        plot(y10, obj.y2p, 'ob', x, y, '-k', 'linewidth', 2)
        set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
              'FontName', 'Times New Roman');
        title('FlowLab vs GINA for pressure');
        xlabel('GINA')
        ylabel('FlowLab');
        lgd = legend('FL vs GINA', 'x=y');
        lgd.FontSize = 24;
            lgd.FontName = 'Times New Roman';
        filename3 = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'FLvsGINA_Pres.fig');
        savefig(filename3);

    end

    %-- volume Plotter function plots volume signals against one
    %another
    % Function currently not used
    function volumePlotter(obj)
        figure
        y1 = obj.newFile{1:obj.int, 4};
        y2 = obj.newFile{1:obj.int, 8};
        y10 = interp1(obj.x1,y1,obj.x0);
        obj.y2v = interp1(obj.x2, y2, obj.x0);
        plot(obj.x0, y10)
        title("Volume comparison test")
        xlabel("Time (s)")
        ylabel("Volume (ml)")
        hold on
        plot(obj.x0, obj.y2v)
        legend({"FlowLab", "NALM"});
        hold off
        filename = fullfile(obj.currentFolder,obj.folderName, 'volumeTest.fig');
        savefig(filename);

    end

    %-- peakPlotter function finds the peak pressure. Currently not in
    %use
    function peak = peakPlotter(obj)
        peak = rms(obj.y2p);
    end

    %-- setFile function applies offset and resets time starts to 0 to
    %align signals
    function setFile(obj)

      %set time back to 0 for nalmdata (GINA)
      offset = 0;
      t1 =  obj.nalm_Data{1,1};
      for i = 1:height(obj.nalm_Data)
          x =  obj.nalm_Data{i,1}-t1;
          obj.nalm_Data{i,1} = x;
```

```
385         end
386
387         %set time back to 0 for labdata (flowLab)
388         t2 = obj.lab_Data{1,1};
389         for i = 1:height(obj.lab_Data)
390             x= (obj.lab_Data{i,1}-t2);
391             obj.lab_Data{i,1} = x;
392
393             %offset volume
394             if offset == 0
395                 if obj.lab_Data{i,4} < 0 && abs((obj.lab_Data{i+1,4} - obj.lab_Data{i,4})/.05)<0.2
396                     offset = obj.lab_Data{i,4};
397                 end
398             end
399         end
400         obj.newFile = [obj.lab_Data obj.nalm_Data];
401         obj.size = height(obj.newFile);
402
403     end
404
405     %-- labDataScalar function applies the scaling factor based on peak
406     %difference to the FlowLab time
407     function labDataScaler(obj)
408         for i = 1:obj.size
409             x_old = obj.newFile{i, 1};
410             x_new = x_old*(1 + obj.scale_fac(1));
411             obj.newFile{i, 1} = x_new;
412         end
413     end
414
415     % -- fileTrimmer function trims signals to not start from the same
416     % moment, based on flow gradient and time stamps
417     function fileTrimmer(obj)
418
419         %Trims both signals to same start time
420         time_interval = etime(obj.T_l, obj.T_n);
421         if (time_interval > 15)
422             obj.ex = 1;
423             return
424         end
425
426         rows_to_delete = time_interval/obj.sam_rate -3;
427         obj.nalm_Data(1:rows_to_delete, :) = [];
428
429         %trim nalm to first positive grad where y > 0
430         f_nalm = table2array(obj.nalm_Data(:,2));
431         t_nalm = table2array(obj.nalm_Data(:,1));
432
433         i_n = 1;
434         m = max(f_nalm);
435
436         for i = 4:length(t_nalm)
437             y4 = f_nalm(i);
438             y3 = f_nalm(i-1);
439             y2 = f_nalm(i-2);
440             y1 = f_nalm(i-3);
441             x4 = t_nalm(i);
442             x3 = t_nalm(i-1);
443             x_2 = t_nalm(i-2);
444             x_1 = t_nalm(i-3);
445             if  (y4-y3)/(x4-x3)> 20 && (y3-y2)/(x3-x_2)> 10 && abs((y2-y1)/(x_2-x_1)) <=10 && y1 > -m/2
446                 i_n = i-3;
447                 break
448             end
```

```
449         end
450
451         obj.nalm_Data(1:i_n, :) = [];
452
453         %trim flowLab to first positive grad
454         f_lab = table2array(obj.lab_Data(:,2));
455         t_lab = table2array(obj.lab_Data(:,1));
456
457         il = 1;
458
459         for i = 4:length(t_lab)
460             y4 = f_lab(i);
461             y3 = f_lab(i-1);
462             y2 = f_lab(i-2);
463             y1 = f_lab(i-3);
464             x4 = t_lab(i);
465             x3 = t_lab(i-1);
466             x_2 = t_lab(i-2);
467             x_1 = t_lab(i-3);
468
469             if (y4-y3)/(x4-x3)> 20 && (y3-y2)/(x3-x_2)> 10 && abs((y2-y1)/(x_2-x_1)) <=10  && y1 > -m/2
470                 il = i-3;
471                 break
472             end
473         end
474
475         obj.lab_Data(1:il, :) = [];
476
477         %Trim both to same end time
478         len_nalm = height(obj.nalm_Data);
479         len_lab = height(obj.lab_Data);
480
481         diff_len = len_nalm - len_lab - 1;
482
483         if(diff_len > 0)
484             %delete the extra cells off nalm_Data
485             obj.nalm_Data((len_nalm - diff_len):len_nalm, :) = [];
486         elseif(diff_len < 0)
487             %same for nalm
488             obj.lab_Data((len_lab - diff_len):len_nalm, :) = [];
489         end
490     end
491
492
493     %-- delayCalc function calculates the scaling factor of the delay
494     %between signals based off peak to peak differences of flow
495     function delayCalc(obj)
496         f_lab = table2array(obj.newFile(:,2));
497         t_lab = table2array(obj.newFile(:,1));
498         [~, locs_l] = findpeaks(f_lab, t_lab, 'MinPeakProminence', 3.5);
499         t_nalm = table2array(obj.newFile(:,5));
500         f_nalm = table2array(obj.newFile(:,6)) ;
501         [~, locs_n] = findpeaks(f_nalm, t_nalm, 'MinPeakProminence', 3.5 );
502         diff = length(locs_n)- length(locs_l);
503
504         if diff > 0
505             locs_n((length(locs_n)-diff +1):length(locs_n), :) = [];
506         elseif diff < 0
507             locs_l((length(locs_l) - diff -1):length(locs_l),:) = [];
508         end
509
510         loc_diff = locs_n - locs_l;
511         x = transpose(0:length(loc_diff)-1);
512         p = polyfit(x, loc_diff, 1);
```

```
513             obj.scale_fac = p(1);
514
515             figure
516             plot(x,loc_diff, 'linewidth', 2);
517             set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
518                     'FontName', 'Times New Roman');
519             title("Delay")
520             xlabel("Samples")
521             ylabel("Time diff (s)")
522             hold on
523             x_1 = linspace(0, length(loc_diff));
524             y1 = polyval(p, x_1);
525             h4 = plot(x_1, y1, 'linewidth', 2);
526             str = ['y = ', string(round(p(1), 3)),'x + ', string(round(p(2), 3))];
527             eqstr = join(str);
528             lbl1 = label(h4,eqstr ,'location', 'bottom'); %% equation of line
529             lbl1.FontSize = 20;
530                 lbl1.FontName = 'Times New Roman';
531                 lbl1.Color = 'k';
532             lgd = legend('Delay', 'Linear approximation');
533             lgd.FontSize = 24;
534             lgd.FontName = 'Times New Roman';
535             hold off;
536             %save delay calc
537             filename4 = fullfile(obj.currentFolder,'Tests_Results',obj.folderName, 'DelayCalc.fig');
538             savefig(filename4);
539         end
540
541         %-- infantDataCollector function was to plot infant data. Presently
542         %not in use.
543         function infantDataCollector(obj, a_vent)
544             dt = 0.001;
545             t0 = a_vent{1,1};
546             for i = 1:height(a_vent)
547                 t = a_vent{i,1};
548                 a_vent{i,1} = t - t0;
549
550                 flow = a_vent{i,2};
551                 vol = 1000/60*flow*dt;
552                 a_vent{i,4} = vol;
553             end
554
555             obj.vent = a_vent;
556
557         end
558
559     end
560 end
```

For BlandAltman code, see post: https://au.mathworks.com/matlabcentral/fileexchange/24645-bland-altman-plot

A.2 Manual for GINA VALIDATOR

--- GINA VALIDATOR MANUAL ---

Protocol to ensure proper GINA_Validator functionality:

1. Before recording any data from the GINA, install the following on your computer: TDM Excel Add-In for Microsoft Excel Download . This will allow GINAs output TDMS files to be converted to xlsx files, which is required for this program to function.
2. Follow this checklist before commencing recording from the GINA and FlowLab:
    a. Both machines must calibrated to current atmospheric conditions (see GINA manual and FlowLab manual).
    b. Connect silicon pipes and connectors to approximate Figure 1. Connections must be as hermetic as possible.
    c. Both machines must for 10 minutes to warm up.
    d. Both machines must be set to have a sampling rate of 0.05s (200Hz).
        i. Set this on GINA through Log > Dec.Factor = 10 (Figure 3).
        ii. Set this on FlowLab through Trending > Configuration > Recording interval = 0.05 (fig 4).
    e. In FlowLab, Trending > Configuration must set to record Time, Flow High, Flow Low, Pressure Difference and Tidal Volume Vte, (Figure 4).
    f. Ensure warning lights on GINA interface are not flashing (Figure 2).
    g. Under Log tab, change File Name to desired. Press 'Store Set.' and save this settings file with the same name as the File Name (Figure 3).
3. Commence recording. Interval between GINA and FlowLab start of recording must be less than 15 seconds.
4. After data recording is complete, open the GINA output tdms file using the *TDM Excel converter* and *save as*. Ensure that the GINA xlsx file and the FlowLab log file are both saved into the Tests_Data folder in the GINA Validator Files.
5. Open FileValidator application.
6. Use *Browse* buttons to select files.
7. *Name output file* in appropriate box. Name must contain no spaces and only the characters in these parentheses (_ - .).
8. Click *GO* to run validation. Plots will be displayed and stored in Tests_Results folder.

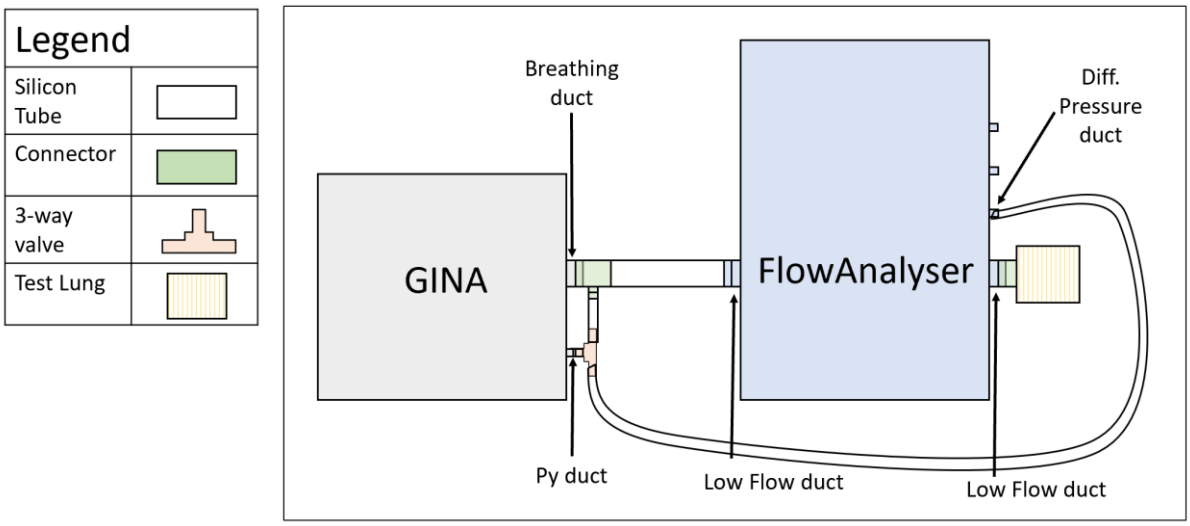


Figure 1: Set-up for GINA to FlowAnalyser connection

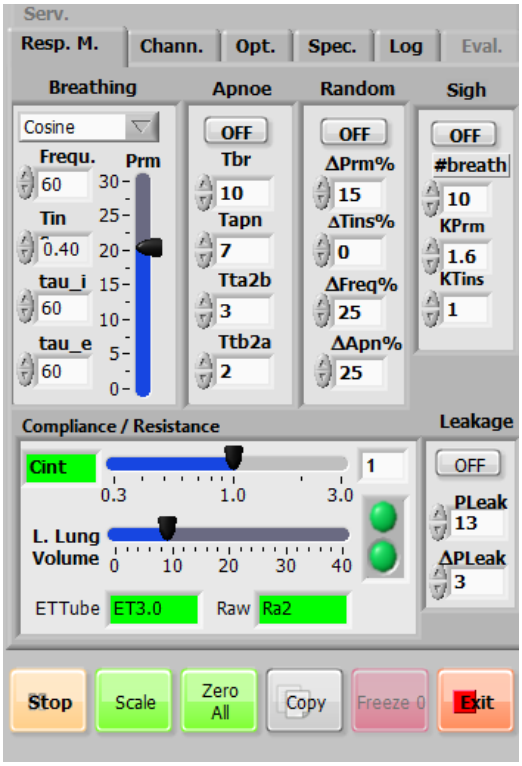


Figure 3: Respiratory Mechanics Setting, GINA v1-2. 'Lights' to the right of 'L.Lung Volume'

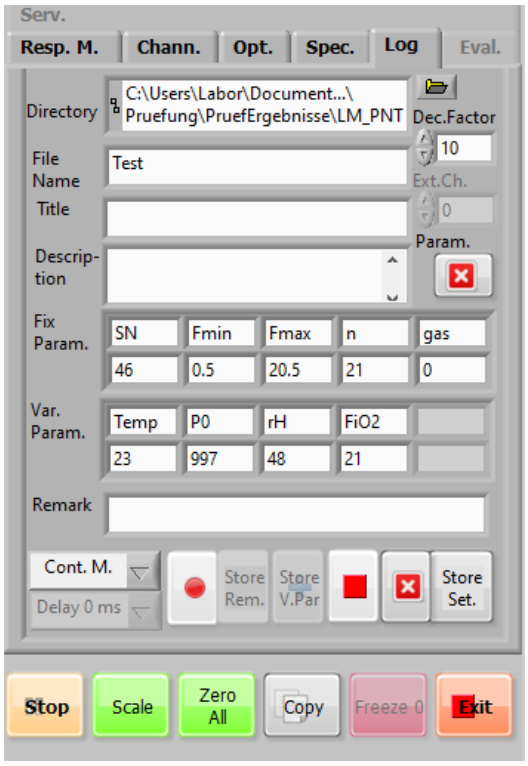


Figure 2: Log settings, GINA v1-2

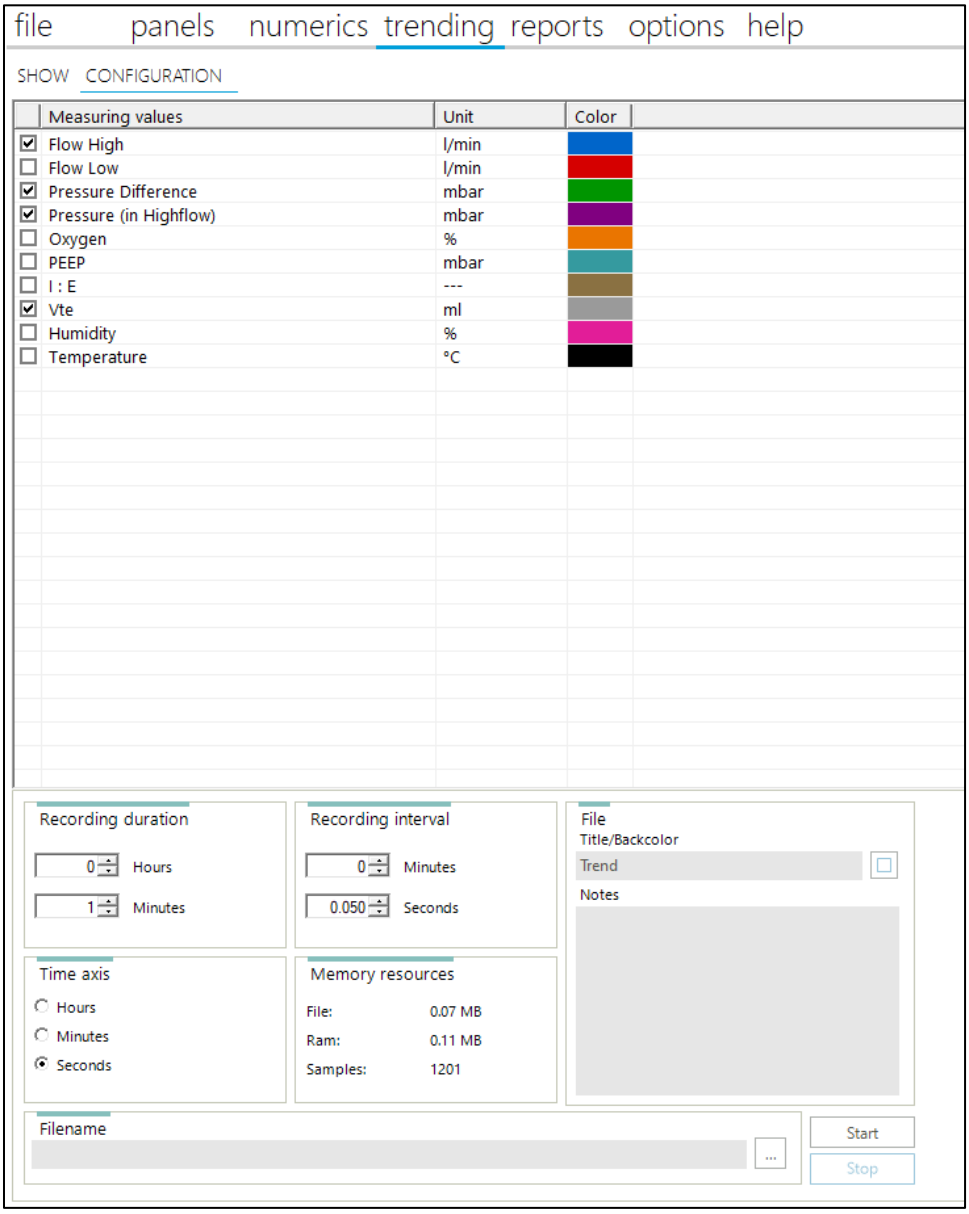


Figure 4: FlowLab settings

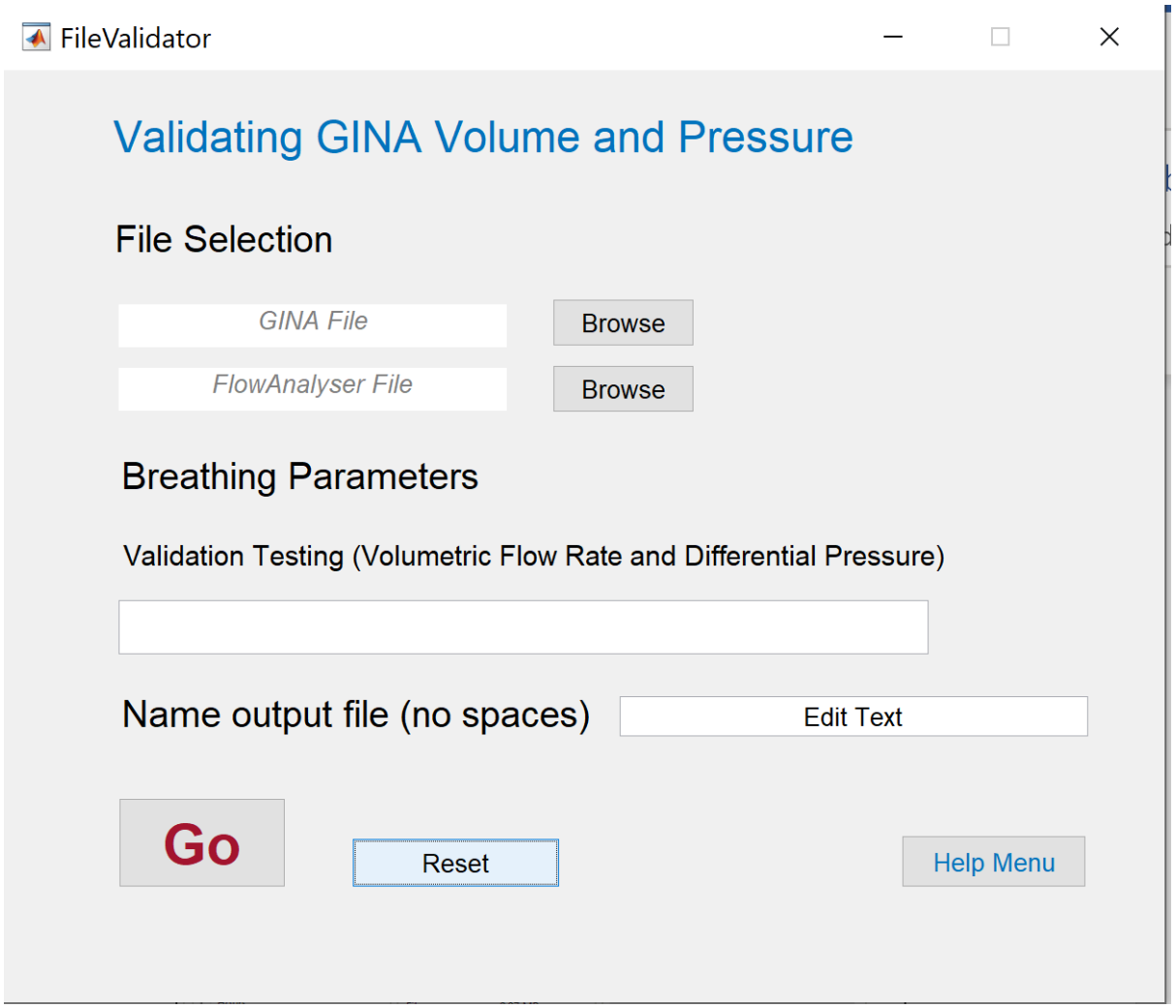


Figure 5: FileValidator GUI

B | Appendix for Core Study Two

B.1 Code for GINA ANALYSER

```
% GINA_ANALYSER MATLAB code for the GINA_ANALYSER GUI.
%       This file GINA_Analyser is responsible for GUI inputs and outputs.
%       It tests that the user has met all requirements for GUI usage and
%       prompts the user to make changes if necessary. Ouputs from this
%       file are sent to GA_MAIN MATLAB file

% --- GUI INITIALISATION CODE - autogenerated (NB: NOT MY OWN WORK)
function varargout = GINA_Analyser(varargin)
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GINA_Analyser_OpeningFcn, ...
                   'gui_OutputFcn',  @GINA_Analyser_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% ---End initialization code - DO NOT EDIT

% --- Executes just before GINA_Analyser is made visible.
function GINA_Analyser_OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for GINA_Analyser
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = GINA_Analyser_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in browse1, which browses for input xlsx
% file from GINA
function browse1_Callback(hObject, eventdata, handles)
currentFolder = pwd;
filename = fullfile(currentFolder, 'GA_TestData', '*.xlsx');
global file_gina
file_gina = (uigetfile({filename}, 'Select one or more files', ...
    'MultiSelect', 'on'));
set(handles.listbox1,'string', file_gina);
out = filegetter(handles, get(handles.listbox1, 'String'), filename, ...
    file_gina, handles.listbox1, hObject, eventdata);
file_gina = cellstr(out);
set(handles.listbox1,'string', file_gina);

% --- Executes on button press go.
function go_Callback(hObject, eventdata, handles)
    % Formats global variables into appropriate data class typess
    global add_table
    add_table = convertCharsToStrings(get(handles.addtable, 'String'));
    global file_gina
    file_gina = cellstr(get(handles.listbox1,'String'));
    global add_check
    global new_check
    global new_table
    % Catch user error of no 'Outout Option' box being checked
```

```
65  if isequal(add_check, 0) && isequal(new_check, 0)
66      fprintf("line 117")
67      warndlg("You must check at least one 'Output Option for Breath by Breath Data'");
68      return
69  end
70  % Catch user error of 'add data...' box being checked but no file
71  % selected
72  if string(add_table)=='0' || isequal(string(add_table),'Select File')
73      add_check = 0;
74      new_check = 1;
75      x = get(handles.newtable, 'String');
76      if isempty(x)
77          warndlg('FileValidator can not run without a valid existing or new table. Select a file and click "Go" again');
78          resetgui(hObject, eventdata, handles, 1)
79          return
80      else
81          a = outfilechecker(x, handles);
82          set(handles.newtable, 'String', a);
83      end
84  end
85  % Catch error of file already existing if 'Create new table' selected
86  if get(handles.newbox, 'Value') == 1
87      check = 0;
88      while check == 0
89          a = get(handles.newtable, 'String');
90          if isempty(a)
91              warndlg('FileValidator can not run without a valid existing or new table. Select a file and click "Go" again');
92              resetgui(hObject, eventdata, handles, 1)
93              return
94          else
95          % First check that user has written in a file name
96          b = outfilechecker(a, handles);
97              if isempty(b)
98                  warndlg('FileValidator can not run without a valid existing or new table. Select a file and click "Go" again');
99                  resetgui(hObject, eventdata, handles, 1)
100                 return
101             else
102                 f = string(outfilechecker(a, handles)) + '.xlsx';
103             end
104         end
105         % Now check if that file name already exists and prompt user
106         %selection to overwrite old file or create new file name
107         currentFolder = pwd;
108         File = fullfile(currentFolder, 'GA_OutputData', f);
109         if isfile(File)
110             str = ['File name " ', File, ' " already exists. Do you want to overwrite this file?'];
111             newStr = join(str);
112             answer = questdlg(newStr, 'Overwrite file? The old version will be deleted', 'Yes', 'No', 'No');
113             switch answer
114                 case 'Yes'
115                     new_table = f;
116                     check = 1;
117                 case 'No'
118                     set(handles.newtable, 'string', 'Enter New File Name');
119                     newFile = inputdlg('Enter a new file name, no spaces: ', 'Output File Name');
120                     set(handles.newtable,'String', newFile);
121             end
122         else
123             new_table = f;
124             check = 1;
125         end
126     end
127 end
128
```

```
129     % Catch error of no input GINA file selection
130     x = get(handles.listbox1, 'String');
131     if isequal(x, '0') || isequal(x,'Select Files')
132         warndlg('FileValidator can not run without a valid file selection. Select a file and click "Go" again');
133         resetgui(hObject, eventdata, handles, 2);
134         clear global
135         clear
136     else
137     % Attempt to run ga_main file. If any additional errors are thrown (eg,
138     % improper file name, excel file open
139         %try
140             ga_main
141         %catch
142             warndlg('Problem using GINA_Analyser. Check that any excel files being written or added to are closed. Ensure file ↙
names do not include illegal symbols. Ensure at least one box is ticked');
143         %end
144         resetgui(hObject, eventdata, handles, 2) ;
145         clear global
146         clear
147      end
148
149
150
151 % --- Checks that a filename has been written by user
152 function a = outfilechecker(f, handles)
153     if strcmp(f, "Enter New File Name") || strcmp(f, "0")
154         uiwait(warndlg('Please enter a new output file name'));
155         newFile = inputdlg('Enter a new file name, no spaces: ', 'Output File Name');
156         set(handles.newtable,'String', newFile);
157         g = get(handles.newtable, 'String');
158         a = outfilechecker(g, handles);
159     else
160         a = f;
161         return
162     end
163
164 % --- Checks that a file has been selected after browse button press
165 function out = filegetter(handles, filestring, filename, setfile, releventhandle, hObject, eventdata)
166 if length(filestring) == 1 && filestring == '0'
167     % set a question
168     answer = questdlg('You must select a file/s. Click "Select File", or "Cancel" to exit and reset.', 'Choose a file', 'Select File', ↙
'Cancel', 'Select File');
169     switch answer
170         case 'Select File'
171             setfile = uigetfile({filename}, 'File Selector');
172             set(releventhandle,'String', setfile);
173             next = get(releventhandle, 'String');
174             out = filegetter(handles, next, filename, setfile, releventhandle, hObject, eventdata);
175         case 'Cancel'
176             out = 'Select File';
177     end
178 else
179     out = setfile;
180     return
181 end
182
183 % --- Executes on button press in browse2, which browses for an existing
184 % table to add data to in 'Ouput Options...' section.
185 function browse2_Callback(hObject, eventdata, handles)
186     %--- Browses for a table, checks that file is selected and catches
187     %error, sets add_table as string of file name and displays file name in
188     %addtable box on the GUI
189     currentFolder = pwd;
190     filename = fullfile(currentFolder, 'GA_OutputData', '*.xlsx');
```

```
191     global add_table
192     add_table = "";
193     add_table = uigetfile({filename}, 'Select one file', 'MultiSelect', 'off');
194     set(handles.addtable,'String', add_table);
195     %tests if 0 is selected
196     out = filegetter(handles, get(handles.addtable, 'String'), filename,...
197         add_table, handles.addtable, hObject, eventdata); %makes sure a file is selected
198     add_table = string(out);
199     set(handles.addtable,'String', add_table);
200
201 % --- Executes on button press in addbox, creates boolean for GA_main to
202 % know if box selected
203 function addbox_Callback(hObject, eventdata, handles)
204 % hObject    handle to addbox (see GCBO) eventdata  reserved - to be
205 % defined in a future version of MATLAB handles    structure with handles
206 % and user data (see GUIDATA)
207     global add_check
208     if get(hObject, 'Value') == get(hObject, 'Max')
209         add_check = 1;
210     else
211         add_check = 0;
212     end
213
214 % --- Executes on button press in newbox, creates boolean for GA_main to
215 % know if box selected
216 function newbox_Callback(hObject, eventdata, handles)
217 % hObject    handle to newbox (see GCBO) eventdata  reserved - to be
218 % defined in a future version of MATLAB handles    structure with handles
219 % and user data (see GUIDATA)
220     global new_check
221     if get(hObject, 'Value') == get(hObject, 'Max')
222         %set(handles.newbox,'String', 'yes');
223         new_check = 1;
224     else
225         new_check = 0;
226     end
227
228 % --- Executes on button press for HELP, brings up the manual.
229 function helpbutton_Callback(hObject, eventdata, handles)
230 % hObject    handle to helpbutton (see GCBO) eventdata  reserved - to be
231 % defined in a future version of MATLAB handles    structure with handles
232 % and user data (see GUIDATA)
233 open GINA_ANALYSER_MANUAL.pdf
234
235 % --- Resets GUI back to its original state
236 function resetgui(hObject, eventdata, handles, erasename)
237 set(handles.addbox,'value',0)
238 set(handles.newbox,'value',0)
239 set(handles.nonspontcheck, 'value', 0)
240 global add_check
241 global new_check
242 global spontcheck
243 add_check = 0;
244 new_check = 0;
245 spontcheck = 0;
246 set(handles.addtable, 'string', 'Select File');
247 if erasename == 1
248     set(handles.newtable, 'string', 'Enter New File Name');
249 elseif erasename ==2
250     set(handles.newtable, 'string', 'Enter New File Name');
251     set(handles.listbox1, 'string', 'Select Files');
252 end
253
254 % --- Executes on button press Reset Fields button
```

```
255 function pushbutton5_Callback(hObject, eventdata, handles)
256 % hObject    handle to pushbutton5 (see GCBO) eventdata  reserved - to be
257 % defined in a future version of MATLAB handles    structure with handles
258 % and user data (see GUIDATA)
259     resetgui(hObject, eventdata, handles, 2);
260
261 % --- Executes on selection change in choicemenu, for selection of Breath
262 % Calculation by WOB or Flow
263 function choicemenu_Callback(hObject, eventdata, handles)
264 % hObject    handle to choicemenu (see GCBO) eventdata  reserved - to be
265 % defined in a future version of MATLAB handles    structure with handles
266 % and user data (see GUIDATA)
267     global flag
268     v = get(handles.choicemenu, 'Value');
269     flag = v;
270
271 % --- Executes during object creation, after setting all properties.
272 function choicemenu_CreateFcn(hObject, eventdata, handles)
273 % hObject    handle to choicemenu (see GCBO) eventdata  reserved - to be
274 % defined in a future version of MATLAB handles    empty - handles not
275 % created until after all CreateFcns called
276
277     if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
278             'defaultUicontrolBackgroundColor'))
279         set(hObject,'BackgroundColor','white');
280     end
281
282 %---Start: Auto-generated code for GUI feature formatting. NB: NOT MY OWN
283 %WORK
284 % --- Executes on selection change in listbox1.
285 function listbox1_Callback(hObject, eventdata, handles)
286
287 % --- Executes during object creation, after setting all properties.
288 function listbox1_CreateFcn(hObject, eventdata, handles)
289 % hObject    handle to listbox1 (see GCBO) eventdata  reserved - to be
290 % defined in a future version of MATLAB handles    empty - handles not
291 % created until after all CreateFcns called
292
293 % Hint: listbox controls usually have a white background on Windows.
294 %       See ISPC and COMPUTER.
295 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
296         'defaultUicontrolBackgroundColor'))
297     set(hObject,'BackgroundColor','white');
298 end
299
300 function newtable_Callback(hObject, eventdata, handles)
301
302 % --- Executes during object creation, after setting all properties.
303 function newtable_CreateFcn(hObject, eventdata, handles)
304 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
305         'defaultUicontrolBackgroundColor'))
306     set(hObject,'BackgroundColor','white');
307 end
308
309 function addtable_Callback(hObject, eventdata, handles)
310
311 % --- Executes during object creation, after setting all properties.
312 function addtable_CreateFcn(hObject, eventdata, handles)
313 if ispc && isequal(get(hObject,'BackgroundColor'), get(0,...
314         'defaultUicontrolBackgroundColor'))
315     set(hObject,'BackgroundColor','white');
316 end
317
318 %---End: Auto-generated code for GUI feature formatting
```

```
319
320 % --- Executes on button press in nonspontcheck.
321 function nonspontcheck_Callback(hObject, eventdata, handles)
322 global spontcheck
323 if get(hObject, 'Value') == get(hObject, 'Max')
324         %set(handles.newbox,'String', 'yes');
325         spontcheck = 1;
326     else
327         spontcheck = 0;
328 end
329
```

```
1
2 % GA_MAIN MATLAB CODE
3 %       This file ga_main is called by GA_Analyser when 'GO' is selected on
4 %       the GINA_ANALYSER GUI. In this file, data from GA_TestData is sent
5 %       to undergo analysis in GA_Tester. Output is arranged into a table,
6 %       saved in GA_Output folder as an xlsx doc. This output file will be
7 %       named according to GINA_ANALYSER GUI selection to 'add data to
8 %       existing table' or 'create new table' and will be named according
9 %       to the users chosen file name. GA_TestData and GA_OutputData are
10 %       saved in the same directory as all MATLAB files.
11
12
13 % ---Imports global values and sets up full file formatting for new_table
14 % and add_table
15 global file_gina
16 global add_check
17 global new_check
18 global flag
19     if isempty(flag) %if no selection made, set automatic to be flag = 2
20         flag = 2;
21     end
22 currentfolder = pwd;
23 global new_table % only executes if new_check ==1
24
25     n = fullfile(currentfolder, 'GA_OutputData', new_table);
26     new_table = n;
27 global add_table % only executes if add_check == 1
28
29     l = fullfile(currentfolder, 'GA_OutputData', add_table);
30     add_table = l;
31 global spontcheck
32
33 % --- Creates bigtable to which all post-analysed data is appended
34 header = {'TestName','Breath', 'Tidal_Volume','Piston_Volume', ...
35     'Insp_Time_s', 'Exp_Time_s', 'Py_max', 'Py_min', 'Py_mean',...
36     'Ptr_max', 'Ptr_min', 'Ptr_mean', 'Palv_max', 'Palv_min', 'Palv_mean'};
37 bigtable = cell2table(cell(0,15), 'VariableNames',header);
38
39 % --- Catches error of the input file already having been saved to the
40 % excel file previously. See fileslistcheck function.
41 fileslist = file_gina;
42 [newfilelist,g,o] = fileslistcheck(add_table,fileslist, add_check, ...
43     new_check, new_table);
44 if isequal(newfilelist,0)
45     return
46 end
47
48 % --- For each input file selection, run GA_Tester Analysis and append to
49 % bigtable
50 for i = 1:length(file_gina)
51 fileslist = string(file_gina(i));
52 test = GA_Tester(fileslist, g, o, flag, spontcheck); %Analysis
53 bigtable = vertcat(bigtable, test.outfile); %Table appending
54 % Data rounding to 2dp (can be commented out if total accuracy required)
55
56     bigtable.Tidal_Volume = round(bigtable.Tidal_Volume, 2);
57     bigtable.Piston_Volume = round(bigtable.Piston_Volume, 2);
58     bigtable.Insp_Time_s = round(bigtable.Insp_Time_s, 2);
59     bigtable.Exp_Time_s = round(bigtable.Exp_Time_s, 2);
60     bigtable.Py_max = round(bigtable.Py_max, 2);
61     bigtable.Py_min = round(bigtable.Py_min, 2);
62     bigtable.Py_mean = round(bigtable.Py_mean, 2);
63     bigtable.Ptr_max = round(bigtable.Ptr_max, 2);
64     bigtable.Ptr_min = round(bigtable.Ptr_min, 2);
```

```
65     bigtable.Ptr_mean = round(bigtable.Ptr_mean, 2);
66     bigtable.Palv_max = round(bigtable.Palv_max, 2);
67     bigtable.Palv_min = round(bigtable.Palv_min, 2);
68     bigtable.Palv_mean = round(bigtable.Palv_mean, 2);
69 end
70 % --- If 'Add data...' box checked in GUI, appends bigtable to existing
71 % table
72 if add_check == 1
73     T1 = readtable(add_table, 'Sheet', 1, 'ReadRowNames', false);
74     newT = vertcat(T1, bigtable);
75     writetable(newT, add_table);
76 end
77
78 % --- If 'Create new table' box checked in GUI, creates table and
79 % overwrites any past data in file
80 if new_check == 1
81    existingmatrix = readtable(new_table, 'Sheet', 1);
82    [m,n] = size(existingmatrix);
83    cleartable = array2table(strings(m,n));
84    writetable(cleartable, new_table, 'Sheet', 1)
85    writetable(bigtable, new_table, 'Sheet', 1);
86 end
87
88 % --- Executes if no errors, informs user that analysis is complete
89 msgbox("Analysis completed");
90
91 % --- This function creates logs added input files to the second sheet of
92 % the output xlsx file. If a the user attempts to add a file that has
93 % already been added (or that has an identical name), user is prompted to
94 % choose to add file again or not.
95 function [f,g,o] = fileslistcheck(addtable, fileslist, addcheck, ...
96     newcheck, newtable)
97     s = length(fileslist);
98     log = strings(s, 1);
99     for i = 1:s
100        log(i) = fileslist(i);
101    end
102
103    log = array2table(log);
104    log.Properties.VariableNames = {'Data_Log'};
105
106    if newcheck == 1 %means we are overriding so need to delete the ...
107        % existing log files
108        g = 0;
109        o = 0;
110        if ~isfile(newtable)
111            T = cell2table(cell(0,7));
112            writetable(T, newtable);
113        end
114        f = fileslist;
115        existingmatrix = readtable(newtable, 'Sheet', 2);
116        n = numel(existingmatrix);
117        cleartable = array2table(strings(n,1));
118        writetable(cleartable, newtable, 'Sheet', 2);
119        writetable(log, newtable, 'Sheet', 2);
120    end
121
122    if addcheck ==1 %check for file previously added
123        o = 0;
124        exOpen(addtable);
125        existingmatrix = readtable(addtable, 'Sheet', 2);
126        n = numel(existingmatrix);
127
128        for i = 1:s
```

```
        for j = 1: n
            if isequal((log{i,1}), string(table2cell...
                    (existingmatrix(j,1))))
                filestring = log{i,1};
                m = ['The data name", filestring,...
                    "has been previously added to this file.",...
                    "Do you wish to add it again or skip?"];
                message = join(m);
                answer = questdlg(message, 'Warning', 'Add again',...
                    'Skip', 'Skip');%send user a warning error
                switch answer %user prompted to answer
                    case 'Add again'
                        number = 1;
                        newf = '';
                        card = 0;

                        while card == 0
                            newf = join([filestring,"(", number, ")"]);
                            if isequal(newf, string(table2cell...
                                    (existingmatrix(j,1))))
                                number = number + 1;
                            else
                                card = 1;
                            end
                        end
                        for k = 1:length(fileslist)
                            if isequal(filestring, fileslist(k,1))
                                fileslist(k,:) = [];
                            end
                        end
                        f = fileslist;
                        l = height(log);
                        log{l, :} = filestring;
                        g = newf;
                        o = filestring;
                    case 'Skip'
                        if isequal(length(fileslist),1) %if there's ...
                            % only one file selected and user ...
                            % chooses to not re-add
                            dlg(Process quit as there are no new files to add');
                            f = 0;
                            g = 0;
                        else %delete the relevant file name from ...
                            %files to be analysed in GA_Tester,...
                            % prevents analysis
                            for k = 1:length(fileslist)
                                if isequal(filestring, fileslist(k,1))
                                    fileslist(k,:) = [];
                                end
                            end
                            f = fileslist;
                            g = 0;
                        end
                end

            else %if file is not already on the log
                f = fileslist;
                g = 0;
            end

        end
    end

    %add new file names to log
```

```
193         t  = vertcat(existingmatrix, log);
194         writetable(t, addtable, 'Sheet', 2);
195     end
196 
197 end
```

```
% GA_TESTER MATLAB CODE
%       This file GA_Tester runs signal analysis on the continuous data
%       provided by the GINA. It determines where breaths start and end and
%       then calculates outputs breath by breath quantities for inspiration
%       and expiration times, volumes and pressures.


classdef GA_Tester <handle
    properties
        filename % string name of the current input file
        filedata % the data from the file
        outfile % the table data specifically
        breaths % number of breaths determined
        time % array of signal sample time
        insp % array of time (s) for inspirations
        exp % array of time (s) for expirations
        wob %Work of Breathing signal
        tv % array of tidal volume per breath
        tvol % array of max piston volume per breath (air moved + dead vol)
        flow % array of signal sample flow
        locs % array of location of peaks (WOB) or negative flow start (flow)
        pres %pressure signal from t piece
        pres_max % array of max pressure per breath at y piece
        presA_max % array of max pressure per breath at alveoli
        presT_max % array of max pressure per breath at trachea
        pres_min % array of min pressure per breath at y piece
        presA_min % array of min pressure per breath at alveoli
        presT_min % array of min pressure per breath at trachea
        p_ave % array of average pressure per breath at y piece
        pA_ave % array of average pressure per breath at alveoli
        pT_ave % array of average pressure per breath at trachea
        g % used if different row names per breath wanted
        o % used if different row names per breath wanted
        insp_func
        breathstart
        breathend
        peaks
        locsdiff %peak-to-peak difference for WOB
        startinsp
        startinsp_y
        freq % a gross approximation of frequency, used for peak finder functions
        spontcheck
    end

    methods
        %--- Initialisation function for the class
        function obj =  GA_Tester(filename, g, o, flag, spontcheck)
            obj.filename = string(filename);
            currentFolder = pwd;
            data = char(fullfile(currentFolder, 'GA_TestData', filename));
            [~,sheet_name]=xlsfinfo(data);
            obj.filedata = readtable(data,'Sheet', sheet_name{4});
            %obj.filedata now contains the continuous data from GINA
            obj.g = g;
            obj.o = o;
            obj.spontcheck = spontcheck;
            bpb_analyser(obj,flag); %call to bpb_analyser
        end

        %--- bpb_analyser function calls all functions to make
        %calculations, creates and populates output table
        function bpb_analyser(obj, flag)  %makes the data table
            try
                alreadySaved = 0;
```

```
65          freqfinder(obj, flag);
66          inspexptime(obj,flag);
67          tidalfinder(obj, 1);
68          tidalfinder(obj, 2);
69          pressure_data(obj, 1);
70          pressure_data(obj, 2);
71          pressure_data(obj, 3);
72          %save figures
73          if flag == 1
74              indicator = '_WOB';
75          else
76              indicator = '_Flow';
77          end
78          fname = join([obj.filename,indicator]);
79          folderName = fullfile(pwd, fname) ;   % Your destination folder
80          mkdir(folderName)
81          FigList = findobj(allchild(0), 'flat', 'Type', 'figure');
82          for iFig = 1:length(FigList)
83              FigHandle = FigList(iFig);
84              FigName   = num2str(get(FigHandle,'Number'));
85              set(0,'CurrentFigure', FigHandle);
86              savefig(fullfile(folderName, [FigName'.fig']));
87          end
88          alreadySaved = 1;
89
90          %make the table
91          header = {TestName','Breath', 'Tidal_Volume',...
92          'Piston_Volume', 'Insp_Time_s', 'Exp_Time_s', 'Py_max',...
93          'Py_min', 'Py_mean', 'Ptr_max', 'Ptr_min', 'Ptr_mean',...
94          'Palv_max', 'Palv_min', 'Palv_mean'};
95          [~, name, ~] = fileparts(obj.filename);
96          %{
97          % Not used in current version, used if seperate row names
98          per breath wanted. Need to use matlab rownames function for
99          this functionality if isequal(name,obj.o) name = obj.g; end
100          %}
101          rownames = cell([obj.breaths,1]);%make a cellular string array
102              % of this size
103          size(rownames);
104          for i = 1:obj.breaths
105              rownames{i} = char(name);
106                  % + i, if separate names per row wanted
107          end
108          out = zeros(obj.breaths, 15);
109          %Populate table with data
110           for i = 1:obj.breaths
111              out(i, 2) = i;
112              out(i, 3) = obj.tv(i);% tidal volume dat
113              out(i,4) = obj.tvol(i);
114              out(i, 5) = obj.insp(i);%inspiratory times
115              out(i,6) = obj.exp(i);
116              out(i,7) = obj.pres_max(i);%max diff pressures
117              out(i,8) = obj.pres_min(i);%min diff pressures
118              out(i,9) = obj.p_ave{i};
119              out(i,10) = obj.presA_max(i);
120              out(i,11) = obj.presA_min(i);
121              out(i,12) = obj.pA_ave{i};
122              out(i,13) = obj.presT_max(i);
123              out(i,14) = obj.presT_min(i);
124              out(i,15) = obj.pT_ave{i};
125           end
126
127           %Delete last row if final breath was not a full breath
128           %(determined based on expiration start locations and
```

```
129             %pressure data
130             while out(length(out), 6) == 0
131                 out(length(out), :) = [];
132                 rownames(length(out)) = [];
133             end
134             while out(length(out), 9) == 200
135                 out(length(out), :) = [];
136                 rownames(length(out)) = [];
137             end
138             while out(length(out), 3) ==0
139                 out(length(out), :) = [];
140                 rownames(length(out)) = [];
141             end
142             k =  array2table(out, 'VariableNames', header);
143             k.TestName = rownames;
144             obj.outfile = k;
145
146         catch e
147             fprintf(e.identifier);
148             fprintf(e.message);
149             fprintf('\n');
150         str = join(['An error occured. Most likely, your data in file ', obj.filename, ' is too random for this version of ↙
GINA_ANALYSER']);
151         warndlg(str);
152         if alreadySaved ~=1 %saving figures
153             if flag == 1
154                 indicator = '_WOB';
155             else
156                 indicator = '_Flow';
157             end
158             fname = join([obj.filename,indicator]);
159             folderName = fullfile(pwd, fname) ; % Your destination folder
160             mkdir(folderName)
161             FigList = findobj(allchild(0), 'flat', 'Type', 'figure');
162             for iFig = 1:length(FigList)
163                 FigHandle = FigList(iFig);
164                 FigName = num2str(get(FigHandle, 'Number'));
165                 set(0, 'CurrentFigure', FigHandle);
166                 savefig(fullfile(folderName, [FigName '.fig']));
167             end
168         end
169         end
170     end
171
172
173       %--- freqfinder function determines number of breaths
174       function freqfinder(obj, flag)
175         %calculation using WOB, channel X
176         if flag ==1
177         % Take file, run peak analysis, count peaks/time,  then get
178         % location of peaks
179             obj.wob  = table2array(obj.filedata(:,9));
180
181             %if wob does not start at a 0 point (ie starts during a
182             %breath), delete til 0 point
183             counter = 0;
184             for i = 1:length(obj.wob)
185                 if obj.wob(i) ~= 0
186                     counter = counter+1;
187                 else
188                     break
189                 end
190             end
191             for i = 1:counter
```

```
            obj.filedata(1,:) = [];
        end

        %reset wob and time properties
        obj.wob  = table2array(obj.filedata(:,9));
        obj.time = table2array(obj.filedata(:,1));
        %find peaks of WOB. Peak determines end of inhalation,
        %start of exhalation. If WOB data > 0, inhalation.
        m = max(obj.wob);
        [obj.peaks, obj.locs] = findpeaks(obj.wob, obj.time,...
            'MinPeakProminence', m/2);
        obj.locsdiff = zeros(1, length(obj.locs)-1);

        for i = 1:(length(obj.locs)-1)
            obj.locsdiff(1, i) = (obj.locs(i+1) - obj.locs(i));
                % gives the peak to peak length
        end
        obj.breaths = length(obj.locsdiff); %number of full breaths
        obj.freq = obj.breaths/obj.time(length(obj.time));
    end

    %calculation using flow
    if flag ==2
        f = table2array(obj.filedata(:,6)); %flow
        t = table2array(obj.filedata(:,1)); %time
        %Butterworth filter design to smooth noise around
        %inflection point in flow signal.
        if obj.spontcheck == 1
            n = 3;
            Wn = 20/(2000/2);
            [b,a] = butter(n,Wn, 'low');
            fil_flow  = filtfilt(b,a,f); % applies filter and fixes phase shift issues.
            str = 'Butterworth';
        else
            fil_flow = lowpass(f, 0.0001);
            str = 'Lowpass';
        end

        % TESTING PURPOSES
        %name = join([obj.filename, '_flowfilt']);
        figure(1)
        plot(t,f, 'linewidth', 2);
        title('Time Domain Filtering')
        set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
            'FontName', 'Times New Roman');
        xlabel = ('Time (s)');
        ylabel = ('Flow (ml/min)');
        hold on
        plot(t, fil_flow, 'linewidth', 2)
        newstr = join(['Filtered :', str]);
        lgd = legend('Unfiltered', newstr);
        lgd.FontSize = 24;
        lgd.FontName = 'Times New Roman';
        hold off

        %now, every time the signal goes from negative to positive,
        %a breath has started. replace flow signal with filtered
        %signal
        for i = 1:length(f)
            obj.filedata{i,6} = fil_flow(i);
        end
        %delete data up to first breath
        counter = 1;
        for i = 1:length(fil_flow)
```

```
256             if fil_flow(i) < 0  || (fil_flow(i+1)-fil_flow(i))/(t(i+1) - t(i)) < 1
257                 counter = counter + 1;
258             else
259                 break
260             end
261         end
262
263         figure(2)
264         plot(t, fil_flow, '-b', t(counter),fil_flow(counter), 'xr');
265         title('Identifying Signal Start');
266         xlabel('Time (s)')
267         ylabel('Filtered Flow (l/min)')
268         legend('Filtered Flow','Start point');
269         set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
270             'FontName', 'Times New Roman');
271
272         i = 1;
273         while i < counter-1
274             obj.filedata(1, :) = [];
275             i = i+1;
276         end
277
278         height(obj.filedata)
279         %reset flow and time properties
280         obj.flow = table2array(obj.filedata(:,6));
281         obj.time = table2array(obj.filedata(:,1));
282         length(obj.flow)
283         length(obj.time)
284         %name = join([obj.filename, '_trim']);
285         figure(3)
286         plot(obj.time, obj.flow, 'linewidth', 2);
287         set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
288             'FontName', 'Times New Roman');
289         title('Did flow trim?')
290         xlabel('Time (s)');
291         ylabel('Flow (l/min)');
292         %breaths will now be equal to number of expirations (flow
293         %from positive to negative)
294         b = 0;
295         for i = i:length(obj.flow)-1
296             if obj.flow(i) >= 0 && obj.flow(i+1) <=0
297                 b = b+1;
298             end
299         end
300
301         obj.breaths = b;
302         %breaths = b
303     end
304 end
305
306 %--- function inspexptime calculates the time in seconds of each
307 %inspiration and expiration per breath
308 function inspexptime(obj, flag)
309     %case of using WOB for breaths
310     if flag  == 1
311         for j = 1:length(obj.locs)  %cycle through the peaks
312             counter = 0; %reset counter. If counter == 1, an
313                 % inspiration start time has been recorded so we
314                 % need to cycle on to the next breath
315             while counter == 0
316                 for i = 1:length(obj.time) %cycle through time
317                     % increments
318                  timecounter = obj.time(i);
319                  % for a time period between timecounter and
```

```
                % locsdiff(i), if WOB > 0, then insp has begun
                if timecounter < obj.locs(j) && (j == 1 || ...
                        (timecounter > obj.locs(j-1)))
                         %ie, if the current time < the
                         %upcoming peak
                    if obj.wob(i) ~= 0 && obj.wob(i) >0
                        counter = 1;
                        obj.startinsp(j) = timecounter;%time
                         %stamp that this inspiration starts
                        break
                    end
                    if counter == 1
                        break
                    end
                end
            end
        end

        obj.insp(j) = obj.locs(j) - obj.startinsp(j);
            %inspiration time equals time stamp of expiration
            % start minus time stamp of inspiration start
       if j>1
        obj.exp(j-1) = obj.startinsp(j) - obj.locs(j-1);
            % expiration time equals time stamp of next insp
            % start minus time stamp of exp start
       end
    end
    %change time beginning back to zero and adjust all time
    %samples
    start_t = obj.filedata{1,1};
    for i = 1:height(obj.filedata)
        new_t = obj.filedata{i,1} - start_t;
        obj.filedata{i, 1} = new_t;
    end
    obj.time = table2array(obj.filedata(:,1));
    %reset startinsp times
    obj.wob = table2array(obj.filedata(:,9));
    m = max(obj.wob);
    [obj.peaks, obj.locs] = findpeaks(obj.wob, obj.time,...
        'MinPeakProminence', m/2); %find location of WOB peaks
    for i = 1:length(obj.startinsp)
        obj.startinsp(i) = obj.startinsp(i) - start_t;
    end
    for i = 1:length(obj.startinsp)
        for j = 1:length(obj.time)
            if obj.time(j) == obj.startinsp(i)
                obj.startinsp_y(i) = obj.wob(j);
            end
        end
    end
    figure()
     x = obj.time;
     y = obj.wob;
     plot(x,y, 'linewidth', 2)
     title('Detection of spontaneous breaths')
     set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
         'FontName', 'Times New Roman');
     hold on
     plot(obj.locs, obj.peaks, 'x', 'color', [.93,.69,.13], 'MarkerSize',10, 'linewidth', 1.5);
     hold on
     plot(obj.startinsp, obj.startinsp_y, 'xr', 'MarkerSize',10, 'linewidth', 1.5);
     lgd = legend('WOB', 'Exp. start', 'Insp. start');
     lgd.FontSize = 24;
     lgd.FontName = 'Times New Roman';
```

```
384             hold off
385         end
386
387         % case of using Flow for breaths
388         if flag ==2
389             %finds timestamps of where inspirations start and whre
390             %expirations start
391             obj.breaths
392             for i = 1:obj.breaths
393                 for j = 1:length(obj.time) - 1
394                     %inspirations
395                     if i== 1&& obj.flow(j) <=0 && obj.flow(j+1) >=0
396                         obj.startinsp(i) = obj.time(j+1);
397                     elseif obj.flow(j) <=0 && obj.flow(j+1) >=0 &&...
398                             obj.time(j) > obj.startinsp(i-1)
399                         obj.startinsp(i) = obj.time(j+1);
400                     end
401                     %expirations
402                     if i==1 && obj.flow(j)>=0 && obj.flow(j+1) <=0
403                         obj.locs(i) = obj.time(j+1);
404                         break
405                     elseif obj.flow(j)>=0 && obj.flow(j+1) <=0 && ...
406                             obj.time(j) > obj.locs(i-1)
407                         obj.locs(i) = obj.time(j+1);
408                         break
409                     end
410                 end
411
412                 %time calculations for inspirations and expirations per
413                 %breath
414                 obj.insp(i) = obj.locs(i) - obj.startinsp(i);
415                 if i>1
416                     obj.exp(i-1) = obj.startinsp(i) - obj.locs(i-1);
417                     if i == obj.breaths
418                         obj.exp(i) = 0;%this means the last breath is
419                           % incomplete, row deleted in bpb_analyser
420                     end
421                 end
422             end
423
424             %change time start back to zero and adjust time stamps
425             start_t = obj.filedata{1,1};
426             for i = 1:height(obj.filedata)
427                 new_t = obj.filedata{i,1} - start_t;
428                 obj.filedata{i, 1} = new_t;
429             end
430
431             obj.time = table2array(obj.filedata(:,1));
432
433             for i = 1:length(obj.startinsp)
434                 obj.startinsp(i) = obj.startinsp(i) - start_t;
435                 if length(obj.locs) == length(obj.startinsp)
436                     obj.locs(i) = obj.locs(i) - start_t;
437                 elseif i < length(obj.startinsp)
438                     obj.locs(i) = obj.locs(i) - start_t;
439                 end
440             end
441
442             locspos = zeros(1, obj.breaths);
443             lengthoflocs = length(obj.locs);
444             for i = 1:length(obj.locs)
445                 for j = 1:length(obj.time)
446                     if i ==1 && obj.locs(i) == obj.time(j)
447                         locspos(i) = obj.time(j);
```

```
                    locsflow(i) = obj.flow(j);
                    break
                end
                if i > 1 && obj.locs(i) == obj.time(j) && obj.time(j) >locspos(i-1)
                    locspos(i) = obj.time(j);
                    locsflow(i) = obj.flow(j);
                end
            end
        end

        for i = 1:length(obj.startinsp)
            for j = 1:length(obj.time)
                if i ==1 && obj.startinsp(i) == obj.time(j)
                    startpos(i) = obj.time(j);
                    startflow(i) = obj.flow(j);
                    break
                end
                if i > 1 && obj.startinsp(i) == obj.time(j) && obj.time(j) >startpos(i-1)
                    startpos(i) = obj.time(j);
                    startflow(i) = obj.flow(j);
                end
            end
        end

        figure(4)
        plot(obj.time, obj.flow, 'linewidth', 2);
        set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
            'FontName', 'Times New Roman');
        hold on
        plot(obj.locs, locsflow, 'xg', 'MarkerSize',10, 'linewidth', 1.5, obj.startinsp, startflow, 'o', 'color', [.93,.69,.13], ↙
'MarkerSize',10, 'linewidth', 1.5 ); %this is where breaths end and start
        lgd = legend('Flow (l/min)','Start Exp','Start Insp');
        lgd.FontSize = 24;
        lgd.FontName = 'Times New Roman';
        title('Breath Detection using Flow Signal')
        xlabel('Time (s)')
        ylabel('Flow (l/min)')
        hold off
    end
    obj.freq = obj.breaths/obj.time(length(obj.time));
end

%--- function tidalfinder calculates peak volumes off volume
%signals
function tidalfinder(obj, tnum)
    if tnum ==1
        vol = lowpass(table2array(obj.filedata(:,7)), 0.0001);
        m = max(vol);
        [pks, lcs] = findpeaks(vol, obj.time,'MinPeakProminence',...
            m/2);
        %name = join([obj.filename, '_tidal_detect']);
        figure(5)
        plot(obj.time, vol, '-b', lcs, pks, 'xr', 'MarkerSize',10)
        set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
            'FontName', 'Times New Roman');
        title('Tidal Volume signal peak recognition')
        xlabel('Time (s)')
        ylabel('Volume (ml)')
        lgd = legend('Volume signal', 'Peak detection');
        lgd.FontSize = 24;
        lgd.FontName = 'Times New Roman';
        while length(pks) >obj.breaths
            pks(length(pks)) = [];
        end
```

```
511             while length(pks) < obj.breaths
512                 pks(length(pks)+1) = 0;
513             end
514             obj.tv = pks;
515             %teevee = length(obj.tv);
516         elseif tnum == 2
517             vol = lowpass(table2array(obj.filedata(:,8)), 0.0001);
518             m = max(vol);
519             [pks, lcs] = findpeaks(vol, obj.time,'MinPeakHeight',...
520                 m/2, 'MinPeakDistance', (1/obj.freq)*2/3);
521             %name = join([obj.filename, '_piston_detect']);
522             figure(6)
523             plot(obj.time, vol, '-b', lcs, pks, 'xr', 'MarkerSize',10)
524             set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
525                 'FontName', 'Times New Roman');
526             title('Piston Volume Signal Peak Recognition')
527             xlabel('Time (s)')
528             ylabel('Volume (ml)')
529             lgd= legend('Volume signal', 'Peak detection');
530             lgd.FontSize = 24;
531             lgd.FontName = 'Times New Roman';
532             while length(pks)>obj.breaths
533                 pks(length(pks)) =[];
534             end
535             while length(pks) < obj.breaths
536                 pks(length(pks)+1) = 0;
537             end
538             obj.tvol = pks;
539             %teevol = length(obj.tvol);
540         end
541     end
542
543     % --- function pressure_data calculates max, min and average
544     % pressures
545     function pressure_data(obj, pnum)
546             % Pressure signal selection
547             if pnum ==1 %py
548                 p = table2array(obj.filedata(:,3));
549                 obj.pres = p;
550             elseif pnum ==2 %Palv
551                 p = table2array(obj.filedata(:,4));
552             elseif pnum ==3 %Ptr
553                 p = table2array(obj.filedata(:,5));
554             end
555             t = obj.time;
556             z = lowpass(p, 0.00001);
557             t_test = t;
558             %trim the first 5 and last 5 data points because of the
559             %filter overshoot
560             counter = 0;
561             while counter < 10
562                     z(1) = [];
563                     t_test(1) = [];
564                     z(length(z)) = [];
565                     t_test(length(t_test)) = [];
566                     counter = counter + 1;
567             end
568             figure()
569             plot(t,p, '-b',  t_test,z, '-r');
570             set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
571                 'FontName', 'Times New Roman');
572             lgd = legend('Pressure signal', 'Lowpass Filtered Wn = 0.0001');
573             lgd.FontSize = 24;
574             lgd.FontName = 'Times New Roman';
```

```
575             xlabel('Time (s)')
576             ylabel('Pressure (atm)')
577             title('Lowpass filtering pressure signal')
578             p = z;
579             p_opp = -1*p; %signal inverted for calculating mins
580
581             %findpeaks only works if signal is above zero. Shifts
582             %signal above zero by amp1, runs peak analysis, shifts
583             %signal and peaks back again
584             max1 = max(p);
585             min1 = min(p);
586             amp1 = abs(max1-min1);
587             %shifts signal up
588             for i = 1: length(p)
589                 p(i) = p(i) + amp1;
590             end
591             max1 = max(p);
592             [pks, w] = findpeaks(p,t_test, 'MinPeakHeight',...
593                 (max1-amp1/4), 'MinPeakDistance', (1/obj.freq)*2/3);
594             pk2pk1 = w(2) - w(1);
595
596             %shift pks and p values back again
597             for i = 1:length(pks)
598                 pks(i) = pks(i) - amp1;
599             end
600             for i = 1:length(p)
601                 p(i) = p(i)-amp1;
602             end
603             max2 = max(p_opp);
604             min2 = min(p_opp);
605             amp2 = abs(max2-min2);
606
607             %Similar routine as previous, using inverted pressure
608             %signal to findpeaks for minimums shifts up
609             for i = 1:length(p_opp)
610                 p_opp(i) = p_opp(i) + amp2;
611             end
612             max2 = max(p_opp);
613             [pks1, locs1] = findpeaks(p_opp, t_test, 'MinPeakDistance', pk2pk1*2/3, 'MinPeakHeight', (max2-amp2/4));
614
615             %shift pks values back again
616             for i = 1:length(pks1)
617                 pks1(i) = -1*(pks1(i) - amp1);
618             end
619
620             %trim if there is a min before the first max
621             if locs1(1)<w(1)
622                 locs1(1) = [];
623                 pks1(1) = [];
624             end
625
626             %if there are more maxs than breaths, clip the last breath
627             while length(pks)>obj.breaths
628                 pks(length(pks)) = [];
629                 w(length(w)) = [];
630                 pks1(length(pks1)) = [];
631                 locs1(length(locs1)) = [];
632             end
633             %plot to see where peaks are identified name =
634             %join([obj.filename, 'pres_detect']);
635             figure()
636             plot(t_test, p, '-b',locs1, pks1, 'xr', 'MarkerSize',10);
637             set(gca,'XGrid','off', 'YGrid', 'off', 'Fontsize', 24,...
638                 'FontName', 'Times New Roman');
```

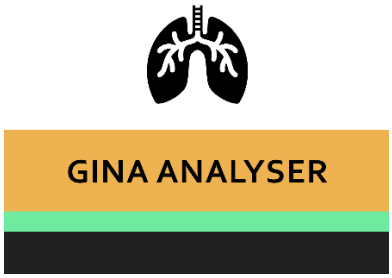
```
639             hold on
640             plot(w, pks, 'x', 'color', [.93,.69,.13], 'MarkerSize',10)
641             title('Pressure max and min identification')
642             xlabel('Time (s)');
643             ylabel('Pressure (atm)');
644             lgd = legend('Pressure (atm)', 'Min', 'Max');
645             lgd.FontSize = 24;
646             lgd.FontName = 'Times New Roman';
647             hold off
648             %add a zero to the end to ensure when data organised in
649             %table in bpb_analyser, arrays are the same length
650             %(otherwise error)
651             while length(pks) < obj.breaths
652                 pks(length(pks) +1) = 0;
653                 w(length(w) + 1) = 0;
654             end
655             while length(pks1) < obj.breaths
656                 pks1(length(pks1)+1) = 0;
657                 locs1(length(locs1)+1) = 0;
658             end
659             %max and min pressures
660             if pnum ==1
661                 obj.pres_max = pks;
662                 obj.pres_min = pks1;
663             elseif pnum ==2
664                 obj.presA_max = pks;
665                 obj.presA_min = pks1;
666             elseif pnum ==3
667                 obj.presT_max = pks;
668                 obj.presT_min = pks1;
669             end
670
671             %calculation of average pressure across each breath
672             sample_mins = cell(obj.breaths,1);
673             for i = 1:length(locs1)
674                 for j = 1:length(t)
675                     if t(j) == locs1(i)
676                         sample_mins{i} = j;%time index location of min
677                     end
678                 end
679             end
680             for i = 1:length(sample_mins)-1
681                 if sample_mins{i}>sample_mins{i+1}
682                     sample_mins(i+1) = [];
683                 end
684             end
685             while sample_mins{length(sample_mins)}>length(p)
686                 sample_mins{length(sample_mins)} = [];
687             end
688
689             %smins = length(sample_mins) s = sample_mins
690             for i = 1:(length(sample_mins)-2)
691                 min_ = sample_mins{i};
692                 max_ = sample_mins{i+1};
693                 sum_p = 0;
694                 for j = min_:max_
695                     sum_p = sum_p + p(j);
696                 end
697                 if pnum ==1
698                     %pave1 = length(obj.p_ave)
699                     obj.p_ave{i} = sum_p/(max_ - min_);
700                 elseif pnum ==2
701                     obj.pA_ave{i} = sum_p/(max_ - min_);
702                     %o = obj.pA_ave
```

```
703                 elseif pnum ==3
704                     obj.pT_ave{i} = sum_p/(max_ - min_);
705                 end
706             end
707 
708             if pnum ==1
709                 while length(obj.p_ave) < obj.breaths
710                     obj.p_ave{length(obj.p_ave)+1} = 200;
711 
712                 end
713             elseif pnum ==2
714                 while length(obj.pA_ave) < obj.breaths
715                     obj.pA_ave{length(obj.pA_ave)+1} = 200;
716 
717                 end
718             elseif pnum ==3
719                 while length(obj.pT_ave) < obj.breaths
720                     obj.pT_ave{length(obj.pT_ave)+1} = 200;
721 
722                 end
723             end
724         end
725     end
726 end
```

For exOpen code, see post: https://au.mathworks.com/matlabcentral/answers/93885-how-can-i-determine-if-an-xls-file-is-open-in-microsoft-excel-without-using-answer_103236

B.2 Manual for GINA ANALYSER

--- GINA ANALYSER MANUAL ---



Author and designer: Corinne Gard, Westmead Hospital NICU & University of Sydney

Contents

Running GINA ANALYSER

Protocol to ensure proper GINA_ANALYSER functionality:

1. Install GINA_ANALYSER software (or transfer files from USB). Ensure that all of these files exist in the same folder. Warning: **Do not change file names**
    a. GINA_Analyser.m
    b. GA_main.m
    c. GA_Tester.m
    d. exOpen.m
    e. GA_OutputData (this is a folder)
    f. GA_TestData (this is a folder)
2. Before recording any data from the GINA, ensure the following is installed on your computer: TDM Excel Add-In for Microsoft Excel Download . This will allow GINAs output TDMS files to be converted to xlsx files, which is required for this program to function.
3. Follow this checklist before commencing recording from the GINA:
    a. GINA must calibrated to current atmospheric conditions (see GINA manual).
    b. GINA should run for 10 minutes to warm up.
    c. Ensure warning lights on GINA interface are not flashing.
    d. Under Log tab, change File Name to desired. Press 'Store Set.' and save this settings file with the same name as the File Name. Select Continuous Mode (Cont. M).
    e. In the case that you intend to calculate breaths using Work of Breathing ensure that channel X is set as Work Of Breathing.
4. Commence recording. When recording is stopped, press 'Beenden' on the window that pops up.
5. Open the GINA output tdms file using the *TDM Excel converter* and *save as*. Ensure that the GINA xlsx file is saved into the GA_TestData folder.
6. Open GINA_ANALYSER application (see figure 1). If the GUI app doesn't immediately open, go to the file GINA_Analyser.m and on the editor tab, click *Run*.

7. On the GINA_Analyser GUI, use *Browse* buttons to select files. There is no file limit.
8. Under the 'Output options for Breath by Breath data' menu, you can choose to create a new breath by breath (bbb) log file, or add to an existing one. Ensure that at least one box is checked.
9. Select the method of breath calculation. Note: in version 1, only 'Work of Breathing (chX)' can be selected.
10. Click *GO.* Data will be stored in GA_OutputData folder.

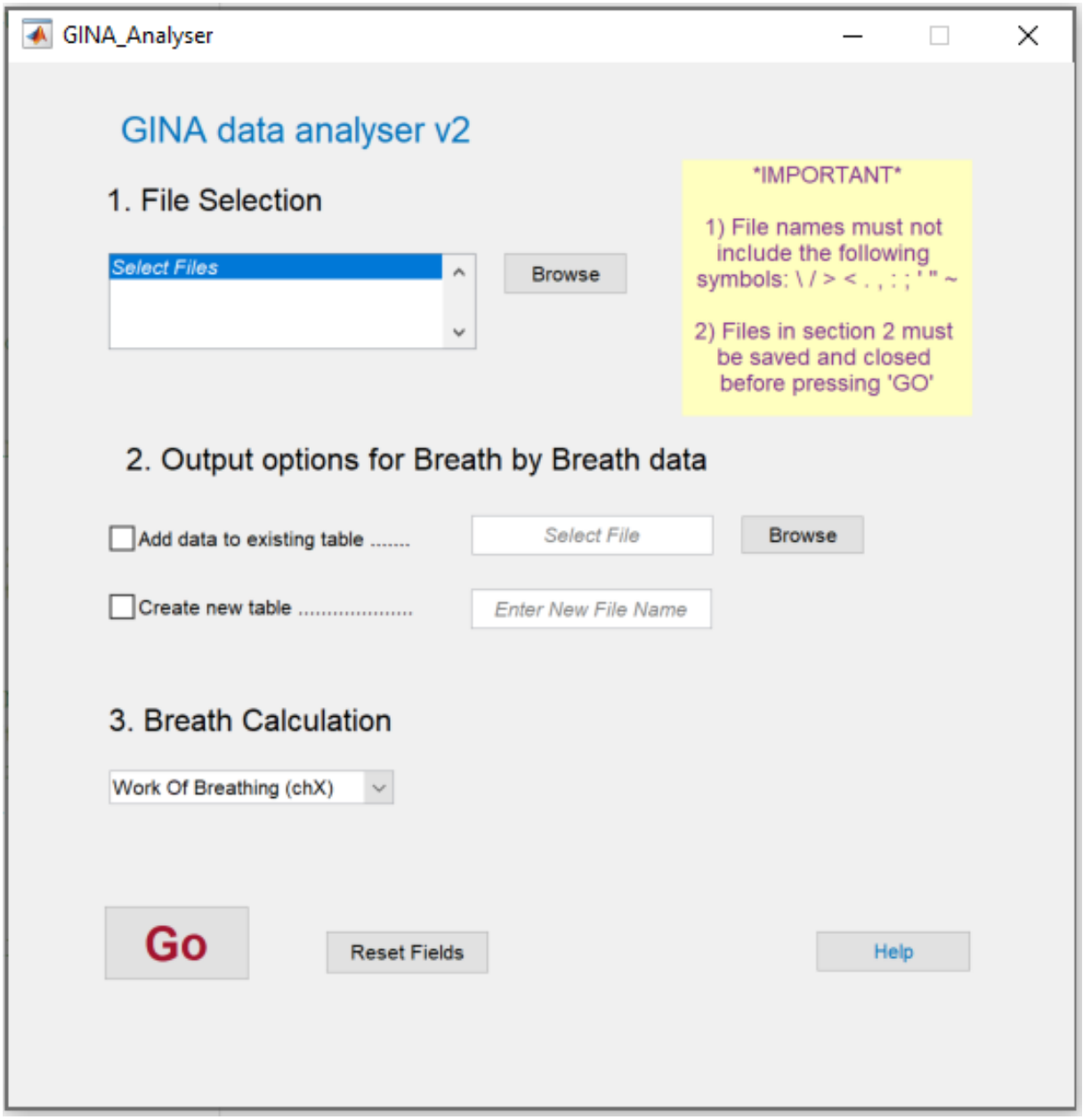


Figure 1: GINA_Analyser GUI

Understanding GINA ANALYSER output

Output from the GINA ANALYSER will be stored in the GA_OutputData folder. If in '2. Output options for Breath by Breath data', 'Create new table' was selected, the file name will be the one you

entered. If 'Add data to existing table' was selected, the file name will be the name of the selected xlsx file.

Your output file, sheet 1 will contain the breath by breath analysis table. Sheet 2 will contain a list of the GINA files that have been analysed in the given xlsx document.

The following table defines output parameter abbreviations and definitions.

Abbreviation	Definition	Units
TestName	Name of input GINA xlsx file	-
Breath	Breath number	-
Tidal_Volume	Volume of air moved in one breath, calculated from flow	mls
Piston_Volume	Volume of air moved in one breath plus piston dead space	mls
Insp_Time_s	Time that inspiration takes	s
Exp_Time_s	Time that expiration takes	s
Py_max	Maximum pressure at y piece (GINA outlet)	atm/hPa
Py_min	Minimum pressure at y piece (GINA outlet)	atm/hPa
Py_mean	Mean pressure at y piece (GINA outlet) over whole breath	atm/hPa
Ptr_max	Maximum pressure at endotrachea	atm/hPa
Ptr_min	Minimum pressure at endotrachea	atm/hPa
Ptr_mean	Mean pressure at endotracheal over whole breath	atm/hPa
Palv_max	Maximum pressure in alveoli	atm/hPa
Palv_min	Minimum pressure in alveoli	atm/hPa
Palv_mean	Mean pressure at alveoli over whole breath	atm/hPa

Figure 2: Output parameters

C | Appendix for additional work

C.1 My design recommendations for GINAs update

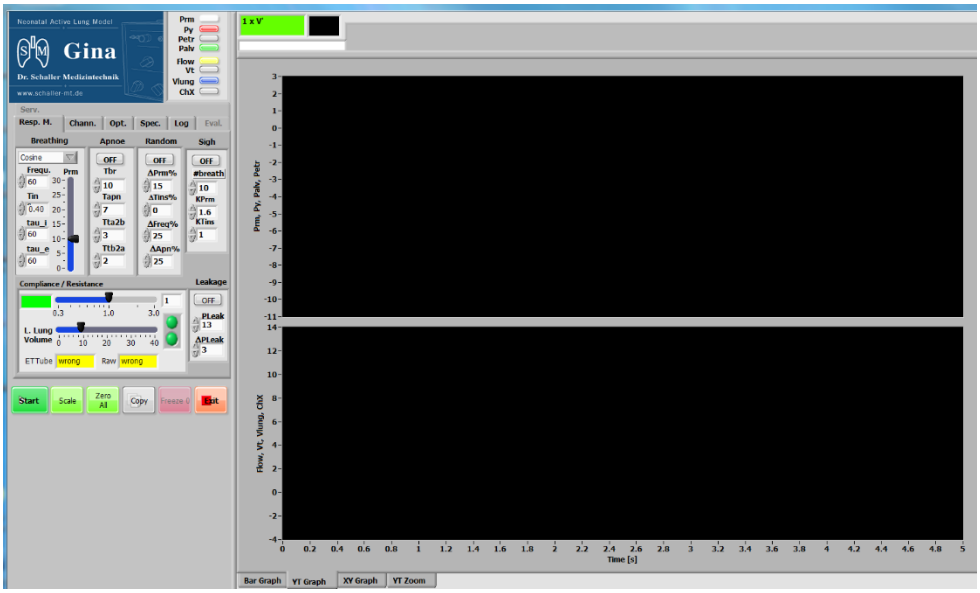
Dear Dr Schaller,

I believe what we have in mind is similar to you.

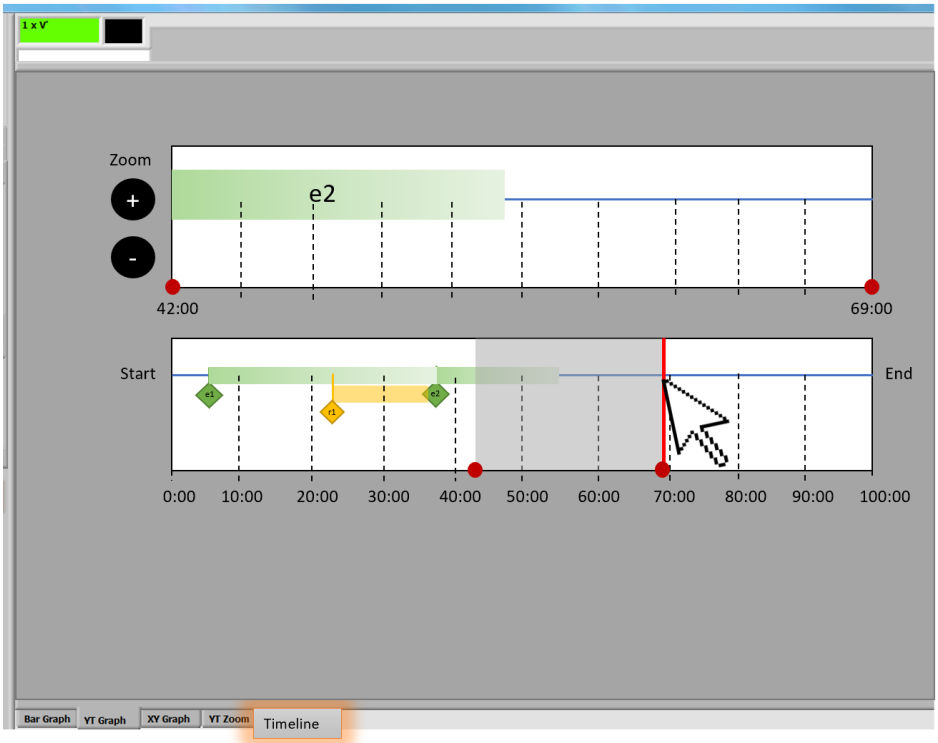
What we would like is to have a GUI interface whereby we set a time duration and can insert events during that timeline.

For example, we might want to select that the duration T = 240s. Then, just like in your example from your email, we could choose points within this 240s to insert an event. An event would be an incidence where the settings (C, Prm, Freq, Tin, Leak, Apnea parameters and so on) could be pre-set to change automatically on the event time line.

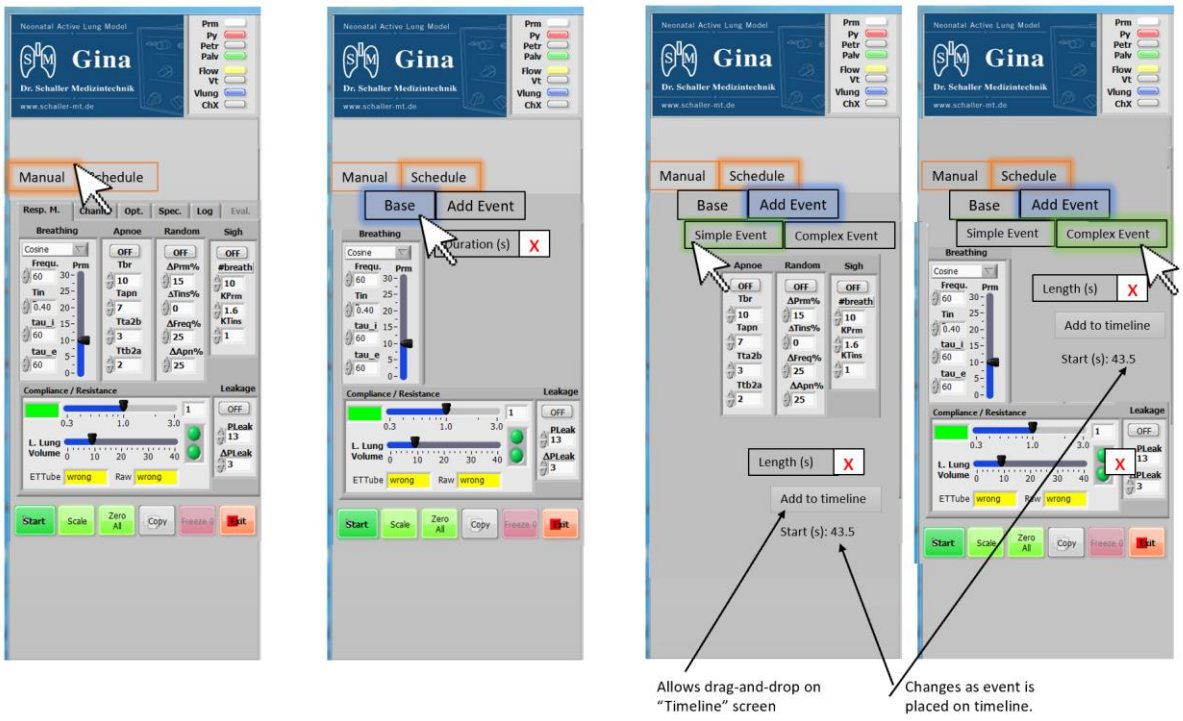
Here is a draft schematic of how I imagine these changes might look. Hopefully it will help to make more sense of exactly what it is that we want GINA to do.

1. There would be two modes to select, “Manual” and “Schedule”
2. “Manual” mode would be the same as the GINA is right now, like this.
    a. 

3. “Schedule” mode would be similar but would have added features including:
    a. A visual timeline, with click and drag features



b. An “add event” option
c. For each event, when it is selected, the user can change the settings
d. “Simple” events can’t overlap other “simple” events. “Complex” events can not overlap “complex” events. But “simple” and “complex” events can overlap. (See image)



e. A log of events

D | Work Log

D.1 Code Repository

To access the GitHub version control repository for the GINA ANALYSER code, please use this link: `https://github.com/Corinne12345/File-Analyser.git`
To access the GitHub version control repository for the GINA VALIDATOR code, please use this link: `https://github.com/Corinne12345/Ventilator-Validator.git`