Dynamic Control of a 2-D Waveguide Model of the Vocal Tract

DOMINIC FREESTON

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2009
Dynamic Control of a 2-D Waveguide Model of the Vocal Tract

DOMINIC FREESTON

Master’s Thesis in Music Acoustics (30 ECTS credits)
School of Computer Science and Engineering
Royal Institute of Technology year 2009
Supervisor at CSC was Svante Granqvist
Examiner was Sten Ternström

TRITA-CSC-E 2009:150
ISRN-KTH/CSC/E--09/150--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication
KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.csc.kth.se
Abstract

Recent work in the field of voice synthesis has suggested a novel approach, using variable impedance maps, for controlling the frequency response of a 2D digital waveguide mesh in order to overcome issues associated with real-time dynamic modification of the shape the vocal tract model. This approach is further explored and a new software framework is developed to facilitate testing and further work. Various impedance mapping schemes are established and evaluated and although a fully successful scheme is not presented, a promising approach, involving a distinction between an ‘air’ section and ‘wall’ section, is suggested. Recommendations for further work include in-depth testing, improved boundaries, new impedance map schemes and software extensions.

# Contents

1 Introduction and Motivation

1.1 The Project .......................... 11
1.2 Why Voice Synthesis? .................. 12
1.3 Why this Approach? ................. 13
1.4 Objectives .......................... 13

2 Overview of Speech Production and Acoustics

2.1 Basic Acoustics ....................... 17
2.2 Speech and the Human Voice ........ 22

3 Speech Synthesis and Physical Modelling

3.1 Speech Synthesis Methods ............ 25
3.2 Physical Modelling Methods .......... 27
3.3 The 1-D Digital Waveguide .......... 27

4 The 2-D Digital Waveguide Mesh and the Impedance Map

4.1 The 2-D Waveguide Mesh .............. 33
4.2 The Impedance Mapping Method ........ 38
4.3 Implementing a Waveguide Mesh ...... 40
5 Analysis of Previous Work 41

5.1 Modelling Issues ............................................. 41
5.2 Software Issues .............................................. 42
5.3 Issues to be tackled .......................................... 45

6 Development of an Impedance Mapping Method 47

6.1 Representing 3D Space in 2D ............................... 48
6.2 Air to Wall Impedance Boundaries .......................... 52
6.3 Mapping Schemes .............................................. 55

7 Software Development 59

7.1 Software Design and Structure ............................... 59
7.2 Working with the Software ................................... 63
7.3 Known Issues and Bugs ....................................... 70

8 Establishing Target Formants 71

8.1 Sources of Data .............................................. 71
8.2 Method ....................................................... 73
8.3 Comments ..................................................... 74

9 Evaluation of the Model 77

9.1 Straight Tube ................................................ 77
9.2 Boundary Issues .............................................. 81
9.3 Two Concatenated Tubes ..................................... 82
9.4 Measured Area Functions ..................................... 86
## Contents

10 Conclusions and Recommendations

10.1 Project Objectives ........................................ 91
10.2 Future Work ............................................... 94

A Frequency Response Plots .................................. 97

B Cross-sectional Area Calculations ......................... 111

C Impedance Map Implementation ......................... 115

  C.1 Flat 1D .................................................. 115
  C.2 Raised Cosine ......................................... 116
  C.3 Flat 2D .................................................. 117
  C.4 Shared Area .......................................... 119

D Verifying Boundaries Implementation ................... 125

References ..................................................... 131
List of Figures

2.1 The human vocal tract .................................. 23

3.1 A 1D waveguide chain with travelling pressure components. . . 28

3.2 Alternative depiction of a single waveguide. .................... 29

3.3 Equivalence of notation and diagrams. .......................... 30

3.4 A one-port boundary junction. ............................... 32

4.1 A four-port scattering junction. ............................... 35

4.2 Topology of an eight by four mesh. ........................... 37

4.3 The raised cosine profile for two concatenated tubes. ...... 39

5.1 Expected topology for an eight by four mesh (with array indexes). ........................................ 44

5.2 Eight by four mesh as implemented by Mullen (with array indexes). ........................................ 45

6.1 Area used to calculated waveguide impedance. ............ 49

6.2 Cross-sectional area of length-wise waveguides. ........... 50
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3</td>
<td>Different central areas for different tube widths.</td>
<td>51</td>
</tr>
<tr>
<td>6.4</td>
<td>Cross-sectional area of width-wise waveguides.</td>
<td>52</td>
</tr>
<tr>
<td>6.5</td>
<td>The Shared Area Impedance Map (Darker = Higher Admittance)</td>
<td>56</td>
</tr>
<tr>
<td>6.6</td>
<td>The Flat 1D Impedance Map (Darker = Higher Impedance)</td>
<td>57</td>
</tr>
<tr>
<td>6.7</td>
<td>The Flat 2D Impedance Map (Darker = Higher Admittance)</td>
<td>58</td>
</tr>
<tr>
<td>6.8</td>
<td>The Shared Area/Flat 2D Impedance Map (Darker = Higher Impedance)</td>
<td>58</td>
</tr>
<tr>
<td>7.1</td>
<td>Overview of code structure.</td>
<td>61</td>
</tr>
<tr>
<td>7.2</td>
<td>VocalTract2D Main Window.</td>
<td>64</td>
</tr>
<tr>
<td>7.3</td>
<td>VocalTract2D Audio Settings.</td>
<td>65</td>
</tr>
<tr>
<td>7.4</td>
<td>VocalTract2D Mesh Settings.</td>
<td>65</td>
</tr>
<tr>
<td>7.5</td>
<td>VocalTract2D Impedance Map Settings.</td>
<td>66</td>
</tr>
<tr>
<td>7.6</td>
<td>VocalTract2D I/O Settings.</td>
<td>66</td>
</tr>
<tr>
<td>8.1</td>
<td>A plexiglass tube</td>
<td>72</td>
</tr>
<tr>
<td>8.2</td>
<td>Loudspeaker used as source</td>
<td>73</td>
</tr>
<tr>
<td>8.3</td>
<td>Microphone used to measure impulse response</td>
<td>74</td>
</tr>
<tr>
<td>9.1</td>
<td>RichAB area function.</td>
<td>87</td>
</tr>
<tr>
<td>9.2</td>
<td>JackAB area function.</td>
<td>87</td>
</tr>
<tr>
<td>9.3</td>
<td>FredAB area function.</td>
<td>88</td>
</tr>
<tr>
<td>A.1</td>
<td>Frequency response for tube ‘richAB’ (1)</td>
<td>98</td>
</tr>
</tbody>
</table>
List of Figures

A.2 Frequency response for tube ‘richAB’ (1) .......................... 99
A.3 Frequency response for tube ‘richAB’ (3) .......................... 100
A.4 Frequency response for tube ‘richAB’ (4) .......................... 101
A.5 Frequency response for tube ‘jackAB’ (1) .......................... 102
A.6 Frequency response for tube ‘jackAB’ (1) .......................... 103
A.7 Frequency response for tube ‘jackAB’ (3) .......................... 104
A.8 Frequency response for tube ‘jackAB’ (4) .......................... 105
A.9 Frequency response for tube ‘fredAB’ (1) .......................... 106
A.10 Frequency response for tube ‘fredAB’ (1) .......................... 107
A.11 Frequency response for tube ‘fredAB’ (3) .......................... 108
A.12 Frequency response for tube ‘fredAB’ (4) .......................... 109
B.1 Slice near the circle’s diameter. ..................................... 112
B.2 A slice away from circle’s diameter. ................................. 113
B.3 A slice across the circle’s diameter. ................................. 114
Acknowledgements

I would like to thank Damian Murphy, both my project and personal supervisor throughout my degree, for suggesting that I consider undertaking my project in Stockholm and for his continued support while I was away. My thanks to Sten Ternstrom and Svante Granqvist for their time, dedication and insight whilst taking care of my supervisory needs in Stockholm. Thanks also to David Howard for his regular encouraging emails.

I would like to thank everyone in the Department of Speech, Music and Hearing at KTH for making me feel welcomed and including me in a number of social, musical and sporting activities right from the start. Special thanks go to everyone involved playing innebandy, which was one of the highlights week after week. Thanks to everyone who supplied me with sources of information and data, it was a privilege to work in an environment with so much expertise in the field.

I would like to thank my father, Mark Freeston, who encouraged me to take the opportunity of going abroad and supported me both financially and emotionally. Finally, I would like to dedicate this work to my late mother, Josee Charbonneau, whom I know would have been proud to see me graduate.

This project was supported by the ERASMUS student exchange scheme.
Chapter 1

Introduction and Motivation

1.1 The Project

This report provides a record, explanation and analysis of the work carried out as part of the project entitled *Dynamic control of a 2-D waveguide model of the vocal tract*.

This first chapter introduces the goals of the project and the motivation for the work carried out. Chapters 2 to 4 cover the necessary theory required for the understanding of the rest of the project, with appropriate references to more detailed explanations. Chapter 5 provides an analysis of the work by Mullen that this project is based on [1], which forms the basis for the rest of the work that was carried out. Chapters 6 to 9 describe the work itself and the results obtained. Finally, chapter 10 provides conclusions and recommendations for further work.

The project is a continuation of work undertaken by Mullen [1] regarding the application of variable impedance to a static 2D digital waveguide mesh in order to control its frequency response for the purposes of vocal synthesis.
1. Introduction and Motivation

It is believed that the 2D mesh has the potential to produce more natural vocal synthesis than is achievable with 1D models, whilst maintaining real-time potential not achievable with current 3D methods.

1.2 Why Voice Synthesis?

Speech synthesis has multiple applications, one of the most important is as a feedback mechanism for visually impaired users of various computer systems. The most obvious interface to drive such a synthesiser is text as it is one of the most common feedback mechanisms used in all forms of computer systems.

A full text-to-speech synthesiser is a complex system. The text needs to be converted to appropriate phonetic representations which vary depending on factors such as language and accent. This representation then needs to be converted to parameters suitable for driving a vocal synthesiser with extra parameters for other factors such as pitch, stress, rhythm and intonation.

This project focuses on the vocal synthesis part of such a system. Although highly intelligible speech has already been achieved, most successfully using concatenative synthesis as used in most commercial systems, natural sounding speech is still not achievable in rule-based systems, let alone real time systems.

This projects attempts to move toward such a system through the development of dynamic vocal tract modelling in 2-D using the digital waveguide mesh based on Mullen’s impedance map method [1].
1.3 Why this Approach?

With many approaches to voice synthesis being explored across the academic world, it is of interest to put this work into proper context. One of the main motivation is that this project has the potential to contribute to ongoing work happening both at the University of York, where the writer carried out his studies, and at Kungliga Tekniska Högskolan, where the project was carried out.

Supervisors at University of York carry out work with waveguides, and its alternative but compatible approach the finite difference mesh, for various purposes including speech synthesis and room modelling. The work by Mullen [1] was carried out within York, and there is a desire for this work to be continued. Furthermore, there is interest in having a real-time 'playable' model of the vocal tract, which requires accurate dynamic control of a real-time simulation.

Supervisors at Kungliga Tekniska Högskolan have particular interest in the high-frequency content of the voice spectrum, which is hoped may be more accurately modelled by a higher dimension model than a simple 1D implementation. Furthermore, speech and voice synthesis is one of the main areas of research within the department and any progress made is therefore of interest.

1.4 Objectives

1.4.1 Initial Objectives

At the start of the project, the following potential objectives were stated:
1. Introduction and Motivation

Objectives

1. Survey the literature to establish existing and alternative approaches.
2. Identify any issues to be fixed and potential improvements to be made with Mullen’s [1] model and establish which ones should be implemented.
3. Adapt/modify the existing software to meet requirements.
4. Establish the relationship between area function and impedance map to get predictable formants in the output.
5. Establish and implement a mapping function taking into account some asymmetries of a real vocal tract.
6. Investigate possible parametric control schemes and how these could be applied to the 2D digital waveguide mesh.
7. Design and code a user-interface implementing the most appropriate parametric control.

The intention was that to make the project as open-ended as possible and so it was never intended for all objectives to be completed. Rather, they ensured that should progress be made faster than expected, it would still be possible to make full use of the time available.

1.4.2 Revised Objectives

Whilst studying the source code for Mullen’s Vocal Tract software, it was decided that the structure of the code made extending and reusing it difficult, and that re-implementing the model in a flexible and reusable way would be a worthwhile undertaking (see section 5.3.1). Due to the increased time requirement for such an undertaking, the last three initial objectives were
unfortunately never reached and remain as potential areas of research for the future. Below are the revised objectives of the project.

Objectives

1. Survey literature to establish existing and alternative approaches.
2. Identify any issues to be fixed and potential improvements to be made with Mullen’s [1] model and establish which ones should be implemented.
3. Re-implement the model in software to meet requirements.
4. Document the functionality of the source code with sufficient detail to allow re-use of all or parts of the software.
5. Establish the relationship between area function and impedance map to get predictable formants in the output.

The final objectives were partially achieved. A previous impedance map scheme was re-evaluated and a new scheme developed and tested. However, there is still much scope for more accurately establishing the relationship between area function and impedance map. It is hoped that the completion of the previous objectives, involving the development of a flexible software framework, will make this task easier in future work.
Chapter 2

Overview of Speech Production and Acoustics

This chapter provides a brief overview of the human speech production system and the associated acoustic mechanisms.

2.1 Basic Acoustics

2.1.1 Definition of Sound

There are two main definitions of sound. In perceptual terms, “sound is the auditory sensation produced through the ear by the alteration in pressure, particle displacement, or particle velocity propagated in an elastic material” [2]. In physical terms, sound is the propagation of mechanical disturbances of particles within a medium, i.e. the physical process which causes the perception described above. It is this latter definition of sound that is simulated in acoustical physical modelling synthesis.
2. Overview of Speech Production and Acoustics

2.1.2 The Speed of Sound

The speed of sound $c$ in air can be approximated by [3]:

$$c = 20.03 \sqrt{273 + T_C}$$  \hspace{1cm} (2.1)

where $T_C$ is the temperature in Celsius.

At the often-assumed room temperature of $20^\circ$C this gives the speed of sound as 343m/s, whilst the temperature inside the human body is closer to $37^\circ$C, which gives the speed of sound inside the vocal tract as approximately 354m/s. However, to maintain consistency with previous work [1], 343m/s is assumed in this work whenever the speed of sound $c$ appears.

2.1.3 Impedance and Admittance

The literature does not agree on standard terminology regarding the different acoustic properties collectively known as impedance. As such, this section defines the terms as they are used in this work.

For every quantity, an equivalent admittance can be obtained by taking the inverse:

$$Z = \frac{1}{Y}$$  \hspace{1cm} (2.2)

Specific Acoustic Impedance

Sometimes simply referred to simply as acoustic impedance (such as by Mullen [1]), specific acoustic impedance is defined as the ratio of scalar pressure, $p$, to particle velocity, $u$. It is a property of the medium and can also be
2. Overview of Speech Production and Acoustics

defined as the product of density, \( \rho_0 \), and speed of sound, \( c \), of the medium. It is sometimes referred to as characteristic impedance [3], which is in contrast to the way it is used here (see below).

\[
Z_s = \frac{p}{u} = \rho_0c
\]  

(2.3)

**Characteristic Acoustic Impedance**

Also known simply as acoustic impedance or characteristic impedance, it is the ratio of pressure, \( p \), to volume flow, \( U = uA \). It is a property of an air column and can be defined as the ratio of specific acoustic impedance to the cross-sectional area of the acoustic tube, \( A \).

\[
Z_c = \frac{p}{uA} = \frac{\rho_0c}{A}
\]  

(2.4)

This is the property that is varied in an impedance map scheme.

**Clarification of Terminology**

Analogies which may help justify the choice of terminology used here are as follows. Specific heat capacity is the property of a medium, the same way that specific acoustic impedance is here. Characteristic impedance of an electrical cable is the property of a component, irrespectively of its length, the same that characteristic acoustic impedance is used here as a property of an acoustic tube. Another term that is sometimes used to describe either of the above properties is wave impedance [4].
2. Overview of Speech Production and Acoustics

2.1.4 The Wave Equation

The second order partial differential equation that describes the propagation of a variety of waves is known as the wave equation.

For the sound pressure of a wave travelling in a one dimensional plane (i.e. in a tube) the equation is [3]:

$$\frac{1}{c^2} \frac{\delta^2 p(x, t)}{\delta t^2} = \frac{\delta^2 p(x, t)}{\delta x^2}$$ (2.5)

where \(c\) is the speed of sound in air, \(p\) is the wave pressure, \(t\) is the elapsed time and \(x\) is the displacement along the tube.

D’Alembert Solution

D’Alembert showed that a numerical solution to the 1D wave equation can be obtained through the separation of wave variables. In other words, the wave motion can be broken down into a ‘left-going’ and a ‘right-going’ component. If we consider equation 2.5 as a difference of two squares, it may be factorised as such:

$$\left(\frac{\delta}{\delta x} + \frac{1}{c} \frac{\delta}{\delta t}\right)\left(\frac{\delta}{\delta x} - \frac{1}{c} \frac{\delta}{\delta t}\right)p(x, t) = 0$$ (2.6)

from which we can see that solutions of the form \(p(x \pm ct)\) will satisfy the equation. Using ‘right-going’ \(p_r\) and ‘left-going’ \(p_l\) components the general form is defined as the sum of travelling components.

$$p(x, t) = p_r(x - ct) + p_l(x + ct)$$ (2.7)

This form is used in the development of the digital waveguide, discussed in section 3.3.
2.1.5 Predicting Resonances

For simple acoustic systems, such as straight tubes and simple arrangements of two concatenated tubes, the resonant modes can easily be predicted mathematically.

**Straight Tube Closed at One End**

The expected resonant frequencies $f_x$ of a straight tube closed at both ends can be obtained from:

$$f_x = \frac{kc}{2L} \quad k = 1, 2, 3... \quad (2.8)$$

where $c$ is the speed of sound and $L$ is the length of the tube [1].

**Straight Tube Opened at One End**

In the case of a tube opened at one end, the expected resonant frequencies $f_x$ can be obtained from:

$$f_x = \frac{(2k + 1)c}{4L} \quad k = 0, 1, 2... \quad (2.9)$$

where $c$ is the speed of sound and $L$ is the length of the tube [1].

**Two Concatenated Tubes Open at One End**

When two tubes of different diameters are concatenated, signal scattering which occurs because of the change in impedance causes the resonant frequencies to move away from their predicted values for single tubes. The
greatest frequency shift occurs when the natural frequency (and therefore the length) of both tubes is equal [5]. In this case, the location of the first two formants can be obtained from:

\[ f'_0 = f_0 \left(1 \pm \frac{2}{\pi} \sqrt{\frac{A_1}{A_2}} \right) \]  \hspace{1cm} (2.10)

where \( A_1 \) and \( A_2 \) are the areas of the respective tubes and \( f_0 \) is the predicted frequency for a straight tube opened at one end of length equal to each of the two concatenated tubes. This equation assumes \( A_1 \ll A_2 \).

**Rectangular Plane**

In the case of two a dimensional rectangular plane bound at all four surfaces, the modes can be predicted using:

\[ f_{xy} = \frac{c}{2} \sqrt{\left(\frac{k_x}{L}\right)^2 + \left(\frac{k_y}{W}\right)^2} \quad k = 0, 1, 2... \]  \hspace{1cm} (2.11)

Note that the equation for the length-wise modes (i.e. by setting \( k_y = 0 \)) is the same as the resonant modes for a tube of the same length.

### 2.2 Speech and the Human Voice

#### 2.2.1 Voice Production Mechanisms

There are four main components of the human speech mechanism: *airstream, phonation, oro-nasal* and *articulatory* processes [6]. The airstream process is the main energy source, providing air from the lungs. The phonation process consists of turning that energy into an audible waveform, a job primarily
2. Overview of Speech Production and Acoustics

Figure 2.1: The human vocal tract

carried out by the glottis (the vocal folds and the area between them). The oro-nasal process can be seen as an ‘on-off’ coupling between the oral cavity and the nasal cavity and is controlled by the velum. The articulatory process modifies the shape of the oral cavity by controlling the articulators, the physical components inside the human mouth used for articulation.

Figure 2.1 shows the structure of the upper part of the human vocal system (excluding the airstream process). The main mobile articulators are the tongue (with the tip, blade, front and back often considered separate articulators), the lips and the jaw. These combine with the static articulators such as the teeth, the hard palate and the soft palate to form various shapes, which affects the sound output of the overall system. For a large number of sounds, the nasal cavity can be ignored as the velum is in the ‘closed’ position. The nasal cavity comes into play for appropriately-named nasal sound such as /m/ and /n/ which are otherwise produced the same way as /b/ and /d/.

To a good approximation, the speech-making mechanism can be thought of as a source-filter model [5], with the glottis being the primary source of
excitation and the vocal tract acting as a filter. This work focuses on the filtering aspect caused by the vocal tract by modelling the change in area, with a focus on producing natural sounding vowels.

2.2.2 Vowel Formants

The filtering effect of the vocal tract gives rise to what is known as 'formants': peaks in the frequency response. The combination of 3-4 formants produces what we recognise as a particular vowel. The formants for each vowel sound have a fixed frequency and are irrespective of the frequency of the glottis excitation, i.e. they are not related to the pitch of the voice [6].

There has been extensive work establishing formants for various vowels. There is some variation in formants from speaker to speaker and so ranges or averages of suitable formants can be obtained. As well as the location of the formants, their bandwidth has a significant effect on the clarity and type of vowel that we perceive [6].
Chapter 3

Speech Synthesis and Physical Modelling

This chapter provides a brief overview of various speech synthesis methods and provides some background on the physical modelling technique known as digital waveguide synthesis.

3.1 Speech Synthesis Methods

The idea of artificial speech reproduction has a history dating back to the middle of the 18th century. Long before computer simulations were possible, attempts at mechanical speech machines were made, most notably by Wolfgang von Kempelen. An overview of the history of voice synthesis through audio samples can be found in [7].

Various approaches to speech synthesis have been taken throughout the years. The majority of approaches can be broken down into the following
three categories: Formant Synthesis, Concatenative Synthesis and Articulatory Synthesis (which falls under the more general technique known as Physical Modelling) [8].

Formant Synthesis attempts to reproduce the output of the speech production mechanism using acoustic models without taking into consideration any similarities between the model and the speech production mechanisms. Although high intelligibility has been achieved using such systems, they tend to sound very unnatural and the control is rather unintuitive due to the lack of clear relationship between the model and its real-life counterpart.

Concatenative Synthesis consists of recording large amounts of speech data then breaking it down into essential components, concatenating these to form new sentences. Certain systems use advanced algorithm to use the longest possible segments in an attempt to improve naturalness. To date, this is the most natural sounding method, but requires storage of large amounts of data for natural sounding speech and offers little flexibility because creating a new voice requires a whole new set of recordings to be made.

Articulatory Synthesis attempts to create synthesised speech with models of the vocal tract and the possible articulations. Physical Modelling Synthesis is seen as the most promising approach to producing natural sounding speech as some of the more complex non-linear aspects are inherently dealt with by having a system that models the real-life phenomena.

As such, the possibilities offered by physical modelling for the purpose of real-time synthesis were further investigated in this project.
3.2 Physical Modelling Methods

The term physical modelling encompasses multiple methods applied to a range of problems, all with the common goal of simulating real physical phenomena to various degrees of accuracy.

These techniques include spring-mass modelling, finite difference models, modal decomposition methods, wave digital filters, source-filter models and digital waveguides.

As the goal of the project is to expand on previous work regarding 2D digital waveguides with varying impedance profiles, some background regarding waveguides will be given in the following section. Other methods of physical modelling synthesis will not be considered in this work. A good overview of various techniques is given in [9].

3.3 The 1-D Digital Waveguide

This section attempts to provide just enough background to understand the function of a digital waveguide model. A detailed description of the theory can be found in [10] and [1] amongst others. Throughout this text, the term digital waveguide will refer to a bidirectional delay of one sample with associated impedance. A 1D combination of many waveguides can be described as a digital waveguide model.

3.3.1 Bi-Directional Wave Decomposition

A one dimensional digital waveguide model implements a discretized solution of the D’Alembert solution to the wave equation (see section 2.1.4). The
right-going’ and ‘left-going’ travelling wave components are implemented using a bidirectional delay line. Whilst useful for simulation, travelling wave components have no real physical equivalent. The wave quantity being simulated, whether it be displacement on a string or pressure in a tube (as in this work), can be obtained by summing the two components at a particular point in space and time.

Figure 3.1 shows a chain of waveguides and how to obtain the output pressure at point $x$ after $n$ samples. The subscripts $r$ and $l$ refer to the ‘right-going’ and ‘left-going’ travelling wave components respectively.

![Figure 3.1: A 1D waveguide chain with travelling pressure components.](image)

An alternative way to illustrate a waveguide used extensively in this work is shown on the left of figure 3.2, with the diagram on the right included for clarification of the equivalence with the previous representation. The minus ($\ -$) symbol represents the outgoing wave components, whilst the plus ($\ +$) symbols represent incoming wave components. The circle is a junction for which a real physical value can be computed (as shown in figure 3.1).

### 3.3.2 Notation

The notation used in this work is similar to that used by Mullen [1].
The physical variable that is modelled in the air column is particle pressure, denoted by the variable $p$.

A single subscript refers to the specified junction, i.e. $p_J$ represents the pressure at junction $J$.

A superscript represents the direction of travel of a travelling wave component. As with the diagram in figure 3.2, the minus (−) symbol represents the outgoing wave components, whilst the plus (+) symbols represent incoming wave components.

Along with these superscripts, a pair of subscripts refer to the travelling wave components on the waveguide connecting two junctions specified by the subscripts.

Therefore, $p_{J,J+1}^-$ represents the wave components outgoing from junctions $J$ towards $J+1$. Also $p_{J,J+1}^+$ represents the wave component incoming into junction $J$ from junction $J+1$.

It is worth noting that $p_{J+1,J}^+ = z^{-1}p_{J,J+1}^-$, that is the pressure outgoing junction $J$ reaches junction $J+1$ one sample later, in accordance with figure 3.2.

Figure 3.3 explicitly illustrates the relationship between the notation described here and the graphical representation introduced in figure 3.2.

Impedances and admittances can be specified using two subscripts, where the order is irrelevant as the impedance seen by both ends of the waveguide...
should be equal \((Z_{a,b} = Z_{b,a})\). However as scattering equations refer to a single junction, the subscript of the junction being updated is commonly omitted.

### 3.3.3 The Scattering Junctions

Signal scattering is caused by a change in characteristic acoustic impedance \(Z\) along the waveguide model, which causes part of the signal to be reflected back along each waveguide.

As described in section 2.1.3, characteristic acoustic impedance is a function of cross-sectional area, therefore in a concatenated tube model the impedance change occurs at the boundaries between tubes.

Signal scattering is simulated using scattering junctions. For a junction \(J\), the outgoing travelling wave components can be obtained using [1]:

\[
p_{J,J-1}^- = p_J - p_{J,J-1}^+	ag{3.1}
\]
\[
p_{J,J+1}^- = p_J - p_{J,J+1}^+	ag{3.2}
\]

where the junction pressure \(p_J\) can be obtained from:

\[
p_J = \frac{2(Y_{J-1}p_{J,J-1}^+ + Y_{J+1}p_{J,J+1}^+)}{Y_{J-1} + Y_{J+1}}	ag{3.2}
\]
3. Speech Synthesis and Physical Modelling

Substituting $Y_{J-1} = Y_{J+1}$ into equation 3.2 allows it to be simplified to $p_J = p_{j,J-1}^+ + p_{j,J+1}^+$, which demonstrates that a standard junction between two waveguides can be regarded as a scattering junction with equal impedance at both ports. As such the circles in figure 3.2 essentially represent scattering junctions as well as potential input/output points to the model.

### 3.3.4 Boundary junctions

Special care must be taken at the physical limits of the model in order to accurately simulate the intended physical object. How to deal with boundaries is a complex problem in itself, but for the purposes of this project the approach used by Mullen [1] was used.

Figure 3.4 shows a simple reflective boundary. As illustrated, such a boundary is equivalent to having an extra waveguide with associated impedance connected to a dummy junction which absorbs any of the energy not reflected. In practice, this can be implemented by specifying a reflection coefficient and calculating the reflected component by multiplying the incoming pressure component by that coefficient, i.e.:

$$p_{B,J}^- = r p_{B,J}^+$$ (3.3)

where r is the reflection coefficient in the range $-1 \leq r \leq 1$. A positive reflection coefficient produces a phase preserving reflection, which corresponds to a closed boundary. A negative reflection coefficient produces a phase inverting reflection, which corresponds to an open end [1].

The junction pressure at the dummy junction and the junction itself need not be simulated.
3.3.5 Physical Dimensions

The physical size of, or length represented by, a single waveguide is a function of the speed of sound and the sampling rate. Each waveguide represents a unit time step $T$, and as signal travels at speed $c$, the distance $d$ travelled over that time must be:

$$d = Tc = \frac{c}{f_s} \quad (3.4)$$
Chapter 4

The 2-D Digital Waveguide Mesh and the Impedance Map

4.1 The 2-D Waveguide Mesh

Multiple waveguides can be connected in arbitrary ways to form Digital Waveguide Networks, a technique used, for example, in the development of reverberation units [10]. An equally-spaced Digital Waveguide Network is known as a Digital Waveguide Mesh. This technique can be used so that the shape of the waveguide is a direct analogue to the shape of the object being modelled, a technique used, for example, in room-modelling applications [11]. In recent years, the possibilities offered by such an approach for the purposes of voice modelling have been investigated [1] [12] and shown promises that warrant further investigation.
4. The 2-D Digital Waveguide Mesh and the Impedance Map

4.1.1 The Multiple-Port Scattering Junction

In order to form digital waveguide networks and meshes, it must be possible to connect multiple waveguides to a single junction. For this purpose, we need to expand the scattering junction (see section 3.3.3) to handle multiple ports.

It can be shown [1] that the junction pressure for an N-port junction can be obtained from:

\[ p_J = 2 \sum_{i=1}^{N} Y_i p_{j,i}^+ + \sum_{i=1}^{N} Y_i (4.1) \]

where all the terms have their usual meaning as described in section 3.3.2. The output travelling wave component at each port can be obtained from:

\[ p_{j,i} = p_J - p_{j,i}^+ (4.2) \]

Figure 4.1 shows a four-port scattering junction. The North (N), East (E), South (S) and West (W) tag represent how directions will be referred to throughout the text. In the software associated with this work (see chapter 7), the North-South direction represents the vocal tract width and the West-East direction represents the vocal tract length, with the glottis at the West end and the lips at the East end.

The pressure equation for such a junction is therefore:

\[ p_J = 2 \frac{Y_N p_{j,N}^+ + Y_E p_{j,E}^+ + Y_S p_{j,S}^+ + Y_W p_{j,W}^+}{Y_N + Y_E + Y_S + Y_W} (4.3) \]
4. The 2-D Digital Waveguide Mesh and the Impedance Map

4.1.2 Physical Dimensions and Dispersion Effects

In a 2D model, the length of a waveguide $d$ is given by [13]:

$$d = \frac{c\sqrt{N}}{f_s}$$  \hspace{1cm} (4.4)

where $c$ is the speed of sound in the medium, $N$ is the model dimensionality (in this case 2), and $f_s$ is the sampling rate of the system.

For a sampling rate of 48 kHz and assuming the speed of sound is 343m/s, the length of a single waveguide in a 2D mesh is approximately 10mm.

It is important to consider the fact that although a 1D waveguide model is a complete solution to the 1D wave equation (2.5), a 2D model is an approximation of the 2D plane. True 2D simulation would require an infinite number of plane waves radiating out from the source, combining to form a circular wavefront. The error caused by the approximation takes the form of frequency-dependent dispersion [1].

In the rectilinear mesh, maximum dispersion occurs in the directions par-
allel to the waveguides and zero dispersion occurs at a 45° angle (i.e. diagonally).

4.1.3 Choice of Mesh Topology

Although various topologies can be implemented, in this study, the rectilinear mesh is used due to its simplicity of implementation, thus allowing for easier experimentation with various mapping methods (see section 4.2). This approach has already showed potential [1] and keeping the same mesh shape allows for existing results to be used as a frame of reference. Furthermore, it has been suggested (by project supervisor Sten Ternström) that the frequency-dependent dispersion effect, although undesirable for exact simulation, could in fact be a beneficial side-effect in the case of voice modelling as the vocal tract has a similar filtering effect which could be difficult to model accurately, especially in a 2D model.

Sampling Rate

One side-effect of the rectilinear arrangement is that the output of the mesh is only valid up to half the Nyquist frequency (i.e. \( f_s/4 \)), due to the fact that the available paths between any two junction can either be odd or even, but never a combination [1].

4.1.4 Boundaries in a Waveguide Mesh

In the case of a rectangular mesh, single-port boundary junctions are used, as shown in figure 4.2. In practice, this means that there is \( N - 1 \) direct paths across the mesh where \( N \) is the number of waveguides in the perpendicular
4. The 2-D Digital Waveguide Mesh and the Impedance Map

direction of travel being considered. This means there are \( N + 1 \) nodes: \( N - 1 \) four-port junctions and 2 boundary nodes. Figure 4.2 shows the exact topology of a 8 by 4 mesh implemented using this approach. As illustrated, there is 9 by 5 nodes or 7 by 3 four-port junctions.

![Figure 4.2: Topology of an eight by four mesh.](image)

From this description it is obvious that an even number of waveguides is required in order to have a single central path, which may be required in some instances.

**Reflection Coefficients**

The choice of reflection coefficients can have a significant impact on the response of the mesh. For the purposes of this work, the values derived by Mullen [1] have been used (and set as default in the software) unless otherwise specified. These are:

- 0.97 at the glottal end.
- -0.90 at the lips.
- 0.94 at both the North and South walls.
4. The 2-D Digital Waveguide Mesh and the Impedance Map

4.2 The Impedance Mapping Method

The impedance map approach attempts to overcome the difficulties in dynamically altering the vocal tract shape by modifying the mesh topology, the main issue being audible non-linear discontinuities in the output [1].

Initially developed by Mullen [1], this approach uses a fixed-sized mesh and applies variable impedances to the mesh’s waveguides. By varying the impedance over time, the frequency response of the mesh can be varied dynamically without changing the actual shape of the mesh.

Another advantage of this approach is the possibility of reducing the required number of nodes as the smallest possible constriction is no longer defined by the waveguide length, which itself depends on the the sampling frequency (see equation 3.4). This allows for a lower sampling frequency, thus opening real-time modelling possibilities.

4.2.1 The Raised Cosine Map

One approach suggested by Mullen [1] is the idea of a ‘raised cosine’ profile. A free path (i.e. equal impedance) of impedance equal to the minimum acoustic impedance in an equivalent 1-D model is kept present through the middle of the mesh. The impedance at the edges is based on the radius of the particular section. The impedance across the mesh varies between $Z_x$ (see equation 2.4) at the edges and $Z_{min}$ in the centre according to a ‘raised cosine’ function. The impedance at a waveguide of coordinates $(x,y)$ would therefore be given by:

$$Z(x, y) = Z_{min} + \frac{Z_x - Z_{min}}{2} [1 + \cos(2\pi \frac{y}{w})]$$

(4.5)
where $w$ is the width of the mesh and $y$ is the distance across the mesh.

It was also suggested that using higher powers of $r$ rather than the true area given by $\pi r^2$ could incorporate some aspects of the missing third dimension (although no mathematical or theoretical justification was offered) and potentially offer a more natural or accurate response so the expression for $Z_x$ becomes:

$$Z_x = \frac{\rho c}{\pi r_x^4}$$  \hspace{1cm} (4.6)$$

Figure 4.3 shows the raised cosine profile applied to a model consisting of two concatenated tubes, a smaller tube followed by a larger tube. The ‘free path’ down the centre can be seen, as well as the fact that the central impedance is equal to the lower impedance of the two tubes. It is worth noting that the width of the mesh is in no way related to the width of the tubes, unless it has been explicitly setup to do so (by having prior knowledge of the area function to be modelled).

Figure 4.3: The raised cosine profile for two concatenated tubes.
4. The 2-D Digital Waveguide Mesh and the Impedance Map

4.2.2 The Linear Map

This approach is the same as the raised cosine map, but uses linear interpolation between \( Z_x \) and \( Z_{\text{min}} \). This approach will not be further investigated due to its similarity to the raised cosine map.

4.3 Implementing a Waveguide Mesh

In a digital waveguide mesh, every junction should be updated every sample, ideally simultaneously. In practice, this is usually implemented using a two-step approach.

The *scattering step* computes the junction pressure based on the incoming travelling wave components. Then, the outgoing travelling wave components are computed. This is repeated for every junction. Boundary nodes are treated differently, with their output computable directly from their input using equation 3.3.

The *timestep* or *delay step* carries every outgoing travelling wave component to the input of the appropriate adjacent junctions.
Chapter 5

Analysis of Previous Work

This chapter looks at the work done by Mullen [1] as part of his PhD thesis, including some of the software he designed. The purpose of this was to identify any issues with the model or software and establish the intended course of action.

5.1 Modelling Issues

Mullen [1] suggested that “a more physically meaningful version of the technique might involve use of moving impedance boundaries. The impedance map could be defined with two distinct regions; a lower $Z_{\text{air}}$ through the centre of the tract and a higher $Z_{\text{flesh}}$ towards the edges of the rectangular mesh.”

This approach is of interest for at least two reasons. Firstly, this would provide a perhaps more meaningful and accurate representation of the space being modelled, which is both more intuitive to work with and more likely to aid understanding of speech mechanisms. Secondly, if a suitable way of
modelling boundaries using impedances can be established, the same method could potentially be used in a 3D dynamic model.

Another issue Mullen mentions regards the need for “clarification of the space represented by the 2D mesh”. This is of particular interest if we are to produce a model that is a meaningful analogue of a real physical tube and is further explored in section 6.1.

Other issues which could be of interest regard glottal excitation, tract energy losses, and additional vocal tract features (e.g. nasal tract).

5.2 Software Issues

This section looks at the source code available or Mullen’s VocalTract software [14]. The source code available is for a different, MIDI-enabled version of the software than the one available, and so unknown differences may be present. It has been assumed that the implementation of the digital waveguide mesh is the same in both versions.

5.2.1 Software Design

Detailed observation of the source code showed that its structure made extensions or code re-use difficult due to a tight binding between user interface and physical model and also due to the scattering of closely related properties across multiple classes. A primary example of this regards the implementation of the mesh and impedance map.

The code also made use of the Microsoft Core Foundation (MFC) library, which has been superseded and is proprietary to Microsoft™. It is thought
that open-source libraries would be more appropriate, partly to limit issues regarding code distribution and full access to library source code. Furthermore, using cross-platform libraries rather than a system-specific framework would improve accessibility to the software.

### 5.2.2 Implementation Error

It was noticed that an implementation error raises questions regarding the accuracy of some of the results obtained for the ‘raised cosine’ mapping methodology (see section 4.2.1). The issue regards the use of array indexes. The program specifies the size of the mesh in waveguides and ensures that the mesh is an even number of waveguides wide, as is required in order to have an odd-number of paths down the mesh and therefore a single central path (see section 4.1.4).

The scattering equations however, are with reference to junctions (or nodes). As mentioned in section 4.1.4, there is one less junction in each direction as there are waveguides, plus two boundary nodes. Therefore a C++ loop iterating over all junctions (excluding boundaries) should take the form:

```c++
for (int x=0; x < waveguideslong - 1; ++x) {
  for (int y=0; y < waveguideswide - 1; ++y) {
    //Loop Body
  }
}
```

which in the case of an 8x4 mesh takes the form seen in figure 5.1. The source code available from Mullen’s work implements these loop as follows:
for (int x=0; x < waveguideslong; ++x) {
    for (int y=0; y < waveguideswide; ++y) {
        //Loop Body
    }
}

which suggests that he implemented an 8x4 mesh as seen in figure 5.2, which is actually a 9x5 mesh.

Although such an observation does not take away from the potential of the method suggested by Mullen, it clearly raises questions regarding the validity of the ‘raised cosine’ method. The main point of interest is that there is in fact no ‘free path’ down the centre of the mesh, as the raised cosine function causes the two central paths to vary in impedance along the mesh.
5.3 Issues to be tackled

5.3.1 Software

Due to the unsuitable nature of the existing implementation, it was decided that re-writing the software would be more beneficial in the long term. Furthermore, another implementation could serve as further validation of existing results. A modular, strict object-oriented approach should be taken to facilitate code re-use for extensions to the work carried out here. This requires appropriate compartmentalisation of the various parts of the program including a clear MVC (model, view, controller) structure. Most crucial is the separation of the physical model from the other parts of the program to allow easy re-use with different interfaces.

The error identified regarding the model dimensions suggested that a visual representation of the model dimensions might be useful as a verifica-
tion step. Furthermore, it was thought that a visual representation of the impedance map based on the data being used by the model (rather than produced separately from the theory) would be a useful verification of the accuracy of its implementation. In addition, such a graphical representation could become a useful teaching tool as visual representations are especially useful to aid understanding of physical modelling problems.

5.3.2 Modelling

It was decided that it would be of interest to implement the ‘raised cosine’ method using the new software in order to observe its effect on a mesh with a single ‘free path’ down the centre as was initially intended by Mullen. Results without the ‘free path’ could also be obtained and compared to Mullen’s.

The alternative approach discussed in section 5.1 with two distinct areas was also chosen to be worthy of further investigation.
Chapter 6

Development of an Impedance Mapping Method

Functional models using 1D waveguides already exist, but it is generally thought that increasing the dimensionality of the model will lead to a more accurate, natural response. However, it has been shown that 3D digital waveguide mesh models require both a high sampling rate and a large volume of nodes, making anything approaching real-time computation unfeasable [12].

A 2D model is the natural compromise, but a high sampling rate is still required if mesh-width variation is the only method employed to alter the frequency response of the model [1]. A coarser (i.e. lower sampling rate and less nodes) mesh with an impedance map, as discussed in section 4.2, has been shown to be a potentially suitable solution [1].

However, further work is required to establish the best impedance mapping scheme. As mentioned in section 5.1, it would be of particular interest to develop a method that provides an intuitive and relatively accurate physical representation of a three dimensional space using a two dimensional model.
This chapter details the logic used to develop an impedance mapping method that could go some way towards achieving this goal. No rigorous mathematical proof is offered, with empirical results intended to be used to indicate the validity of the method.

There are two main issues to be considered: (1) clarifying the space represented by the 2D mesh and (2) defining two distinct regions and the transition between them.

The issue of the space represented by the 2D mesh could apply equally to a static varying-width mesh as it could apply to the ‘air’ region in a two-region impedance mapping scheme and so is considered first in the following section. The issue of two distinctive regions could also be used as part of alternative mapping schemes.

6.1 Representing 3D Space in 2D

6.1.1 Sharing the Tube Cross-Sectional Area Between Waveguides

In a 1D model, a vocal tract shape is approximated to concatenated tubes of different areas. The characteristic acoustic impedances for each of these tubes is calculated based on the cross-sectional area as in equation 2.4 and scattering junctions (see section 3.3.3) are used to model the reflections at such junctions. A simple 3D model may use the same concatenated tube approximation, but model the physical shape of the tubes, thus modelling cross modes not present in a 1D model.

A 2D model could combine both approaches, i.e. represent one physical dimension using waveguides and the other physical dimension using the wave-
guide’s characteristic acoustic impedance. A potentially suitable approach can be deduced by comparing the 1D and 3D models.

Considering the cross-sectional area of a tube, a 1D model uses the entire area to calculate the impedance of a single waveguide. The 3D model usually has equal impedances across all nodes, but this can be thought of as the area used for the impedance calculation being limited by the surrounding waveguides; as the distance between waveguides is equivalent so too is the area represented by each waveguide and thus the impedance is equal. This is not true near the boundaries, but can be ignored if the mesh is fine enough. Figure 6.1 shows the area represented by the ‘central’ waveguide for both the 1D and 3D models.

![Figure 6.1: Area used to calculate waveguide impedance.](image)

The 2D case can be thought of as a single plane down the diameter of the circular tube. Figure 6.2 shows the area used to calculate the acoustic impedance of the central waveguide in the 2D case. Appendix B provides the necessary mathematical background to calculate the area.

There are a number of potentially interesting features of such an approach. Firstly, the impedance of the waveguides through the middle of the mesh is lower than the impedance of the waveguides nearer the walls. This may go
some way towards including some of the effects of the third dimension not present in the waveguide mesh. Secondly, in the case of concatenated tubes of different diameters (i.e. any vocal tract area function) the impedance through the middle of the mesh will not be constant due to varying tube widths, as illustrated in figure 6.3, and so no ‘free path’ down the mesh will be present. This is in contrast to a 3D model, but in accordance with a 1D model. Interestingly, the varying in impedance is likely to be less than that observed in a 1D mesh, suggesting that perhaps only the effect of the dimension being modelled by the impedance is indeed being modelled.

This approach could potentially be used to include some features of interest and asymmetries found in real vocal tract shapes whilst maintaining the computational speed offered by a 2D mesh over a 3D mesh. In such a case, care would be needed in deciding the orientation of the mesh down the vocal tract. In the simple case of concatenated tube considered in this work, the orientation of the mesh is of no consequence.
6. Development of an Impedance Mapping Method

6.1.2 Cross-Sectional Area For Perpendicular Waves

The next issue to be considered is the impedance of the width-wise waveguides, i.e. the ones carrying wave components perpendicular to the tube length. Here, there are no obvious 1D and 3D analogies, but a potential solution can be found by extending the logic used above, that is, consider the cross-sectional area being represented by each waveguide.

As shown in figure 6.4, the effective area that the waveguide represents can be approximated by the average waveguide width of the tube at that point multiplied by the width of a waveguide. This is in fact roughly equivalent to the shaded area on the figure due to the regular rectilinear configuration. This area is in between the area for the central waveguide and the next waveguide along, which provides a gradual impedance change along the mesh’s width in a similar way to the raised cosine scheme.

This case is only valid in the event that the tube sections before and after the waveguides being considered are the same. If the two tubes are of varying length, then the average of the two expected areas could be taken, that is:
6. Development of an Impedance Mapping Method

In the event where the waveguide is nearer the boundary for the larger tube and not present or required for the smaller tube, this must be dealt with in a different way. In the case of a static varying-width mesh then the waveguide will simply not be present, with the length-wise waveguide in the larger tube terminated by a boundary node as described in section 3.3.4. How this could be dealt with in a fixed-width impedance-mapped mesh is discussed in the next section.

6.2 Air to Wall Impedance Boundaries

The derivation of the boundaries in section 3.3.4 uses a dummy unilateral waveguide with an impedance equivalent to that of the wall medium. This allows the derivation of a reflection coefficient that can be applied directly rather than implementing the dummy waveguide. An obvious way to simu-

\[
A_x = \frac{w_{\text{tube}1} \cdot w_{\text{waveguide}1}}{2} + \frac{w_{\text{tube}2} \cdot w_{\text{waveguide}2}}{2} \quad (6.1)
\]

Figure 6.4: Cross-sectional area of width-wise waveguides.
late a boundary in a fixed-size mesh could be to apply the impedance of the wall medium to the waveguides considered to be part of the wall.

The next issue to consider regards what to do with the pressure components once they have reached the wall. One option would be to simply set a high wall impedance and accept that some pressure would eventually return in the model. The alternative would be to remove the energy that reaches the ‘inside’ of the wall.

Therefore, there are three main possibilities as to how to deal with boundaries:

1. Static-style boundaries with reflection coefficient.
2. Set the wall impedance and accept that energy will return.
3. Set the wall impedance with absorption inside the wall (i.e. stop the travelling wave components from propagating within the wall).

The first and third option can be implemented by setting a flag for nodes considered to be inside the wall and treating such nodes in the appropriate way when computing a sample step. The second option simply requires an appropriate impedance to be set for the wall (the third option also requires an appropriate wall impedance value).

The first option is likely to cause the audible non-linearities that Mullen observed when dynamically modifying the mesh dimensions. The second one seems less appropriate, especially considering that the speed of sound inside the wall should change. The third one is equivalent to implicitly implementing boundaries, which may potentially also cause audible non-linearities.
6. Development of an Impedance Mapping Method

6.2.1 Defining the Wall Impedance

It can be shown that the output at a junction port is equivalent the input at that port multiplied by a reflection \( r \) coefficient plus a proportion from the other inputs [1].

\[
p_k^- = r_k p_k^+ + \sum_{i,i\neq k}^N (1 + r_i) p_i^+ \tag{6.2}
\]

where \( r_k \) is given by:

\[
r_k = \frac{Y_k - \sum_{i,i\neq k}^N Y_i}{\sum_{i}^N Y_i} \tag{6.3}
\]

Assuming that the incoming pressure at all other ports is zero, the node essentially acts as a simple reflective boundary for a particular port. This ‘absorption’ inside the wall can be achieved one of two equivalent ways: treating nodes considered to be part of the wall differently (implementable as part of the scattering stage) or intercepting travelling wave components before they reach the input ports, thus ensuring that waves do not propagate through the wall (implementable as part of the timestep stage).

If we consider a boundary node in a straight piece of tube, three of the waveguides can be considered to be inside the wall. Assuming we know the impedance of the air waveguide connected to the junction in question, we can re-arrange equation 6.3 to give an expression for the wall waveguides impedance \( Z_{\text{wall}} \) based on \( Z_{\text{air}} \) and the desired reflection coefficient:

\[
Z_{\text{wall}} = 3Z_{\text{air}} \frac{1 + r_k}{1 - r_k} \tag{6.4}
\]

As an example, for a reflection coefficient of 0.94 a value of \( Z_{\text{wall}} \) equal to \( 97Z_{\text{air}} \) would be required. In the event where two of the waveguides are
6. Development of an Impedance Mapping Method

part of the ‘air’ section and have the same impedance, the situation becomes very different. Substituting this value of $Z_{\text{wall}}$ into equation 6.3 (taking care of converting the impedance values into admittance) results in a reflection coefficient of -0.01.

Re-arranging equation 6.3 for this latter scenario the equation becomes:

$$Z_{\text{wall}} = -Z_{\text{air}} \frac{1 + r_k}{r_k}$$

which suggests that only negatives reflection coefficients are obtainable for such an arrangement, which in a 2D model will occur at transitions between tubes. Furthermore, as illustrated with the previous example, the amount of reflection is completely different and could result in excessive absorption at these particular nodes.

Further difficulties include the fact that the impedance of the ‘air’ section is likely to vary along the mesh and so a fixed reflection coefficient can only be achieved by varying the wall impedance accordingly.

No solution to these problems is offered here, but the effects of various decisions is documented in chapter 9 and suggestions for further exploration can be found in chapter 10.

6.3 Mapping Schemes

As well as the ‘raised cosine’ scheme introduced by Mullen [1] (see section 4.2.1), the intention was to explore the potential of other mapping methods. Below is a brief description of each scheme, along with images of the schemes applied to an arrangement of two concatenate tubes.
of the schemes both an ‘impedance-weighted’ view and an ‘admittance-weighted’ view are included as they highlight different properties of the mapping schemes. Appendix C contains the source code for each mapping scheme.

6.3.1 Shared Area

The ‘shared area’ map, figures 6.5 and 6.8, is an approach combining the concepts discussed in this chapter. The impedance for each waveguide is calculated based on the area the waveguide is considered to be representing and the width of the air section is set to be as near as possible to the tube diameter (limited by the mesh spatial sampling rate). Possible ways to implement the air-to-wall boundaries are discussed in the previous section.

![Figure 6.5: The Shared Area Impedance Map (Darker = Higher Admittance)](image)

The impedance map can be implemented using the following steps:

- Calculate the radius to use as reference.
- Establish if a waveguide is within the tube or part of the wall.
- If within the wall, set an appropriate flag and set the wall impedance.
- If part of the air, calculate the relevant area and from this the impedance.
6. Development of an Impedance Mapping Method

6.3.2 Flat 1D

The ‘flat 1D’ map, figure 6.6, simply consists of applying the 1D impedance, obtainable using equation 2.4, across the whole width of the mesh. This is unlikely to be a suitable solution as there will most likely be little benefit over a 1D model due to there being no variation across the width of the mesh. However, this could be used as a useful point of reference as the location of the main formants are likely to be as accurate as they are in the 1D model.

Figure 6.6: The Flat 1D Impedance Map (Darker = Higher Impedance)

6.3.3 Flat 2D

The ‘flat 2D’ map, figures 6.7 and 6.8, consists of applying the 1D impedance to the whole ‘air’ section of the mesh but also include the ‘wall’ section. The idea is that the change in impedance down the centre of the mesh could have the effect of the 1D model on the location of the main formants, whilst having the width equal to the diameter for each tube could result in more accurate higher-frequency cross modes.

The impedance map can be implemented using the following steps (although alternative methods could achieve the same results):
Figure 6.7: The Flat 2D Impedance Map (Darker = Higher Admittance)

- Apply the 1D impedance to the whole mesh.
- Calculate the radius to use as reference.
- Establish if a waveguide is within the tube or part of the wall.
- If within the wall, set an appropriate flag and reset the wall impedance.

Figure 6.8: The Shared Area/Flat 2D Impedance Map (Darker = Higher Impedance)
Chapter 7

Software Development

One aspect of the project was to produce software that is flexible enough to allow further work to be carried out with minimum effort. This chapter explains some of the design choices, as well as describe the functionality of the various classes in order to facilitate later use. A more detailed description of the classes can be found in the associated documentation [15], produced from the source code using Doxygen [16]. The entire source code, including the libraries used, is included on the accompanying CD.

7.1 Software Design and Structure

7.1.1 Language and Libraries

It was decided that for maximum flexibility, cross-platform and open software should be produced. The software is written in the C++ programming language due to its efficiency and maturity.

Two software libraries were used to facilitate development.
The Synthesis Toolkit (STK) [17] is “set of open source audio processing algorithmic synthesis classes written in the C++ programming language”, developed by Cook and Scavone to “facilitate rapid development of music synthesis and audio processing software”. It makes use of RtAudio [18] for real-time audio input and output and offers in-built ‘wave’ file read and write capabilities. It has a number of pre-existing signal generators which could be used as excitation functions as well as customisable filter units which could be used if required in further model extensions.

The Qt library [19] is used mostly for GUI purposes. Qt produces native style GUI elements, known in Qt terminology as widgets, on multiple platforms, including Mac OS, Windows and the most popular Unix-like variants. Other advantages of Qt which influenced its choice as a library include Qt’s Signal and Slot and meta object properties. The former allows for different objects to communicate with no knowledge of each other, as long as they have matching signals and slots (i.e. with the same parameters). The latter allows objects to be queried as to their type, inheritance and properties. Combined with the efficiency and flexibility of C++, it was thought that this library would provide a suitable development environment.

### 7.1.2 Programme Structure

In order to allow for maximum flexibility, various parts of the software were designed to be as self-contained as possible. Although this requires a relatively large number of classes, the intention is to allow ease of use of certain parts without requirement for the others. Furthermore, it should allow for easier extension, modification or bug-fixing of a particular part of the program. Figure 7.1 provides an overview of the relationship between the various
classes, with a clear separation between the physical model and the user interface.

The Physical Model

The core of the program consists of the DynamicMesh and ImpedanceMap (and its inherited) classes. Both inherit from the ‘Stk’ class so that they work seamlessly with the STK framework. However, they do not inherit from or use any of the Qt library classes in an attempt to allow maximum flexibility.
by allowing the same code to be used with any other framework (alongside STK).

All ImpedanceMap’s data and methods are protected (i.e. only accessible by itself, inherited classes and friend classes), but DynamicMesh is a friend of the class and therefore has access to them. This separation between the two classes is to allow new impedance map schemes to be easily implemented by inheriting ImpedanceMap. Meanwhile, the close relationship ensures that all communications go through the DynamicMesh class, thus reducing the risk of other parts of the code interfering with its proper operation.

**The Interface Layer**

The MeshController class provides a Qt-based wrapper around the physical model, allowing for communication using Qt’s Signal and Slots mechanism. It also allows the mesh to be seemingly resized with the same impedance map and area function applied to it, although it in facts creates a new mesh and reappplies the same parameters.

The AudioController class offers similar functionality for the RtAudio class (although not all of RtAudio’s functionality has so far been implemented), allowing easy query of available audio interfaces and sample rates. In the current state of the program, this is not actually used for playback but would be necessary for real-time synthesis, which is one of the main interesting possibilities offered by the 2D waveguide mesh.

**The User Interface**

The SettingsControl class is a dialogue box with four tabs, each holding its own widget, the Signals and Slots of which can be accessed via the relevant functions. These widget classes are:
7. Software Development

- AudioSettings
- MeshSettings
- ImpedanceMapSettings
- IOSettings

The intention here is that any or all of these could be used in a different setting, or a single one could be replaced or modified without affecting the functionality of the others.

The Graphical Display

The MeshView class provides a graphical representation of the mesh. It holds separate pointers to each node and associated edges, allowing display of waveguide impedance and/or node pressure. Combined with MeshController’s processAnimation() function which sends out periodic updates of all the node pressures, the latter can be used to display an animation of the wave propagation through the mesh.

7.2 Working with the Software

7.2.1 Interacting with VocalTract2D

VocalTract2D, shown in figure 7.2, was designed to accomplish the following purposes:

- Provide visual verification of new impedance map implementations.
- Compare various mesh sizes and impedance map types.
7. Software Development

Figure 7.2: VocalTract2D Main Window.

- Get the model response for any area function with any excitation source.

The ‘Preferences’ menu allows audio (figure 7.3), mesh (figure 7.4), impedance map (figure 7.5) and input/output settings (figure 7.6) to be modified easily. An area function can be applied to the mesh using ‘Open Area Function’. The audio output can be obtained by clicking process and once this is done, the output can be played back by clicking ‘Play’.

If ‘Animation’ is selected in ‘IO Settings’, the ‘Process’ button is disabled and ‘Play’ processes and plays back an animation of the wave propagation through the mesh.

7.2.2 Working with the Dynamic Mesh

The DynamicMesh class provides control over mesh size, area function, impedance map type, impedance map parameters and reflection coefficients via the relevant functions [15]. It also has a sampling rate factor (SRF) which can
Figure 7.3: VocalTract2D Audio Settings.

Figure 7.4: VocalTract2D Mesh Settings.
7. Software Development

Figure 7.5: VocalTract2D Impedance Map Settings.

Figure 7.6: VocalTract2D I/O Settings.
be used to easily run the mesh at a higher sampling rate than that of the audio playback (i.e. the mesh computes the number of time steps specified each time it is asked to compute one sample). The mesh allows access to impedance, admittance and pressure values as well as the parameters for the current impedance map, since impedance maps cannot be queried directly by other classes (to avoid other classes interfering with the proper function of the mesh).

It also offers control over how to deal with ‘flesh’ or ‘wall’ areas of the model. These functions only apply to impedance maps which make use of the ‘inside wall’ flag for nodes that are considered to be part of the wall. There are two two flags which can be set to control this: absorption and static. The absorption mechanism stops the signal from propagating inside the wall by setting the relevant travelling wave components to zero. The static mode ignores the wall impedance and uses standard 1D boundaries with reflection coefficients. Setting neither to true cause the scattering equations to ignores the ‘inside wall’ flag.

One feature of the DynamicMesh class not currently accessible in VocalTract2D is the possibility of doing a linear transition between two impedance maps, providing a basic diphthong facility. The transition time is specified and the target area is specified as a new impedance map and area function (allowing possible transition between different mapping schemes if required) and the mesh provides a linearly-interpolated transition between the two impedance maps. At the end of the transition, the target impedance map is set to be the current impedance map and the previous one is deleted.
7.2.3 Creating New Impedance Maps

Creating a new impedance map implementation is relatively straight-forward. A class that inherits ImpedanceMap must be created. If the map has user-controllable parameters, the constructor must be re-implemented (with the original constructor called) and the parameters specified there.

As mentioned, setAreaFunction() method re-sizes the area function then calls the calculateImpedance() method, which must be re-implemented. The setAreaFunction() is virtual and can be re-implemented if the default area function interpolating scheme is considered unsuitable.

The calculateImpedance() method must be implemented. The requirements are:

- both the impedance and admittance must be calculated.
- the values must be stored in the from the point of view of each node.
- the impedance seen by two connected nodes must be equal.
- the maximum and minimum values must be stored.

To facilitate this, utility functions have been written in ImpedanceMap. There are two functions that obtain maximum and minimum values for either impedance and admittance, and two functions to obtain all impedance and admittance values from their reciprocal.

As mentioned in the previous section, an ‘inside wall’ flag can be set which allows for special treatment of the ‘fleshy’ part compared to the normal ‘air’ sections to be implemented in the DynamicMesh.
7.2.4 Command Line Programs

As well as VocalTract2D, two small command-line programs have been written. As well as provide specific functionality, they provide a good introduction to using the DynamicMesh class as they are short and have a simple procedural structure.

The first one, ‘impulseresponse’ allows quick computation of the impulse response of a mesh with particular settings and parameters. The response is save to a ‘out.wav’ file and can then be used to calculate the frequency response of the mesh.

The second, ‘speakmachine’ demonstrates the transition feature of the DynamicMesh. It requires a number of text files (a script file and at least two area function files) and a ‘.wav’ file which contains the source signal. The script file must take the following structure:

- Each line represents a single vowel or a transition.
- A line with ‘vowelFile time’ represents a single vowel.
- A line with ‘startVowel time targetVowel’ represents a transition.
- The time must be specified in samples.

As an example, a file running at 48000kHz containing:

```
vowel1.txt 48000
vowel1.txt 24000 vowel2.txt
vowel2.txt 48000
```

would have the area function of vowel1.txt for 1s, followed by a transition from vowel1.txt to vowel2.txt that takes 0.5s and finally hold vowel2.txt for 1s.
7.3 Known Issues and Bugs

7.3.1 Impedance Map Parameters

In VocalTract2D, the slider and number box control for the Impedance Map parameters only work with the impedance map selected first. This is likely to be due to the fact that the sliders are destroyed and recreated when the map is changed, but the exact source of the error has not been identified. The problem also occurs when the mesh factor or size is modified after the map type has been selected due to the fact that changing the size of the mesh actually creates a new instance and requires the map to be re-applied.

7.3.2 Animation Mode

The ‘Process’ button is enabled in ‘Animation’ mode if the area function is loaded in after ‘Animation’ is selected. The button has no effect even when enabled, and so this bug is relatively minor. There is currently no proper way of terminating the animation before it ends. However, reducing the number of required samples to below the current number of samples elapsed (e.g. 1) will end the animation immediately. The animation functionality is not especially robust and should be used with care.

There is currently no way to control the speed of the animation, but this could be implemented relatively easily. It can be currently modified in the source code simply by changing the frame length, specified in ms.
Chapter 8

Establishing Target Formants

In order to first verify the model is running as intended and then compare the success of various impedance map schemes, there must be appropriate benchmarks against which the response of the model can be compared.

For relatively simple arrangements, such as a single straight tube or two concatenated tubes, resonances can be predicted using equations derived from the theory (see section 2.1.5). For more complex arrangements, such as those resembling a vocal tract, experimental data is necessary.

8.1 Sources of Data

There are many sources of data, including vowel tract shape data and area functions gathered from imaging techniques such as x-ray (2D data) and MRI (3D data). Both have their disadvantages, most notably the 2D and therefore incomplete nature of x-rays and the slow acquisition speed of MRI machines. The length of acquisition in MRIs prevents the test subject to vocalise for the entire length of the scan, requiring non-ideal solutions such
8. Establishing Target Formants

There has also been extensive research regarding the formants expected for various vowels. However, the purpose of this work is not purely to recreate the target formants using any arbitrary rules. One of the main purposes of this work is to establish a mapping method that would produce predictable and accurate formants based on a given area function (i.e., concatenated tubes). As such, it would be useful to know the exact formants expected from such a tube, rather than from the vocal area shape approximated by the concatenated tubes. For this purpose, it was decided to take impulse response measurements of plexiglass tubes (figure 8.1) made from 5mm slices stacked on two metal rods. This would allow the output of the model to be compared directly to the shape that is being modelled.

Figure 8.1: A plexiglass tube as the subject attempting to maintain the vocal tract shape while breathing after vocalisation.
8. Establishing Target Formants

8.2 Method

The frequency response measurements were taken using Sirp [20], a program designed for measuring the frequency response of loudspeakers. The software uses a sine sweep rather than an impulse to overcome signal to noise ratio issues associated with impulse measurements and allows use of loop back to eliminate the effect of the audio interface from the results.

A small loudspeaker was used as the source (figure 8.2) with plasticine used to make a seal between the loudspeaker and the plexiglass tube. At the other end, the Behringer ECM 8000 microphone (figure 8.3) was place just below the tube opening to minimise its effect as an obstacle. Furthermore, the tubes were placed on the desk with the end protruding slightly off the desk to avoid reflections from the table influencing the measurements.

The frequency response of six existing plexiglass tubes were measured, from each end, providing 12 area functions. This was done once before the tubes were taken apart for diameter measurement and once after, on different days, in order to provide a confirmation of measurements and minimise the
risk of a systematic error. The area of each slice was measured to the nearest 0.1mm. As well as the frequency response, an audio recording using a glottal-like excitation was made for non-rigorous perceptive comparisons.

As the intended vowel of the tubes was not known, each was given an arbitrary name and one end labeled A and the other B, so that a particular area function can be referred to as ‘nameAB’ and the area function of the same tube in the opposite direction referred to as ‘nameBA’.

Files containing the area functions as well as the impulse and frequency responses of all the tubes are included on the accompanying CD. As well as the original Sirp files, the impulse response is available as an audio file and the frequency response as a text file.

8.3 Comments

There are a few issues that need to be taken into account. Firstly, the size of the loudspeaker used as a source suggests that it will have a poor low-frequency
response and as such the response of the tube at lower frequencies may not be fully captured.

Secondly, the frequency response of the microphone is also unknown and so is likely to have some effect on the response. However, the microphone is unlikely to have significant sharp peaks, and so should have a minimal effect on the appearance of formants.

Lastly, the reflection of the tube material has not been established, but it is most likely that it will be very different to that of a vocal tract, and so the reflections chosen for voice synthesis may not be the most suitable for comparing the output of the model to that of the plexiglass tubes.
Chapter 9

Evaluation of the Model

This chapter covers the testing of the physical model and observations on the frequency response resulting from a range of situations. For testing purposes, the reflection coefficients were set to the maximum value (+1) in order to maximise reflections.

9.1 Straight Tube

To verify that the model is operating as intended, the response of the straight tube was observed, as this can be easily predicted mathematically (see section 2.1.5).

Substituting $L = dx$, where $d$ is the length of a single waveguide and $x$ is the length of the mesh in waveguides, and equation 4.4 into equation 2.8 leads to the following expression for the resonances of a tube closed at both ends (reflection coefficients of 1):

$$f_z = \frac{k f_s}{2x\sqrt{N}}$$  \hspace{1cm} (9.1)

72
9. Evaluation of the Model

<table>
<thead>
<tr>
<th>Size $x$ (waveguides)</th>
<th>Measured $f_0$ (Hz)</th>
<th>Predicted $f_0$ (Hz)</th>
<th>$x - 1 f_0$ (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>565.4</td>
<td>530.3</td>
<td>565.7</td>
</tr>
<tr>
<td>21</td>
<td>424.8</td>
<td>404.1</td>
<td>424.3</td>
</tr>
<tr>
<td>36</td>
<td>243.2</td>
<td>235.7</td>
<td>242.4</td>
</tr>
</tbody>
</table>

Table 9.1: First resonance for a straight tube open at one end (original boundaries).

whilst doing the same with equation 2.9 leads to the following expression for a tube opened at one end (reflection coefficient of 1 at one end and -1 at the other):

$$f_z = \frac{(2k + 1)f_s}{4x\sqrt{N}} \quad (9.2)$$

where $f_s$ is the sampling frequency, $N$ is the model dimension (2 in this case). Assuming $x \gg y$ (where $y$ is the width of the model) so that the first few resonances are entirely dictated by $x$, equations 9.1 and 9.2 allow prediction of of the resonant frequencies based on the sampling frequency and the mesh length in waveguides.

Experimental data, outlined by the location of $f_0$ in table 9.1, for a range of tubes showed that the resonances were equivalent to a tube one waveguide shorter.

Further analysis of the source code suggested that the error was caused by the implementation of the boundaries. The reflection from the boundary was calculated straight from the output of the preceding junction such that it arrived back at the junction at the next sample. This effectively causes the waveguides at the edges to be half the length than intended, making the
overall length tube one waveguide shorter. Comparison to Mullen’s implementa-
tion [1] showed that this issue was also present. However, the error regard-
ing the number of waveguides highlighted in section 5.2.2 caused the model to be the intended size (however there is still no ‘free path’ down the centre of the mesh), which would explain why it was never identified. There is no known evidence to suggest that this was a purposeful decision rather than the coincidental result of two programming errors.

9.1.1 Influence of Mapping Schemes

Some impedance map schemes cause a straight tube to act as an equal-
impedance mesh, such as the ‘flat 1D’ and ‘raised cosine’ schemes. This is appropriate considering that the response of such a mesh is accurate (with a few issues highlighted in the following section).

However, such an approach does not take into account the width of the tube. As mentioned previously, as long as $x \gg y$ then this is not an issue for the lower resonances. However, the higher frequency cross-modes will clearly be affected by this dimension and so schemes that implement a distinction between air and wall and ensure the air section is of an appropriate width may be at an advantage.

The ‘shared area’ and ‘flat 2D’ schemes are examples of such schemes. However, it is important to first ensure that they behave as expected in the straight tube case. As discussed in section 6.2, there are three main ways to implement boundaries. Observing their impact on the straight tube configuration could go some way towards establishing the most appropriate solution.
9. Evaluation of the Model

Flat 2D

Implementing the air-to-wall boundary nodes as normal reflective boundaries, in the manner of a static width-wise mesh causes the mesh to produce the predicted resonances, which was to be expected considering it is essentially acting as an equal impedance mesh. The added advantage here is that the mesh can be of any width without affecting the effective width of the tube, which is dictated by the area specified in the area function.

Implementing the ‘absorption’ method causes the resonances to move away from their predicted value. However, it was thought that this could be caused by a similar effect to that observed with the ‘corrected’ boundaries, discussed in the next section. To verify this, the North and South boundaries were set to be ‘corrected’ whilst leaving the East and West boundaries as they were. A tube of the correct width was then tested using an equal impedance mesh. The resulting resonances were indeed equal to those observed using the ‘absorption’ approach. This confirms that in a straight tube section with appropriately set impedance values and ‘absorption’ in the walls, the ‘virtual wall’ node acts as a ‘correctly’ implemented wall node.

Shared Area

The shared area impedance map with reflective boundaries has a first resonant peak at 574.2Hz. This is of interest as it is of higher frequency than the predicted 565.7Hz, which indicates the model acts as a shorter tube. This is counter intuitive as it implies the signal propagates through the model slightly faster than expected, unless the tube as somehow been indeed shortened by the implementation. No answer to this observation can be currently provided and this issue may merit further investigation. Implementation
using the absorption boundaries produces a resonance at 544.9Hz, which is slightly lower than the predicted value. The increased impedance around the central path may be the cause. These results clearly raises some questions regarding the suitability of the shared area mapping scheme.

9.2 Boundary Issues

Attempting to fix the boundary error mentioned previously raised new apparent problems. The location of the first resonance was not as predicted, and the width of the tube began to have an effect on its location. As it was first suggested that this could be the result of a coding error, the main scatter/delay loop was re-implemented two other, more explicit ways, which confirmed that this was indeed the behaviour of the simple reflective boundaries. Appendix D includes the source code and describes the differences between the various implementations.

Table 9.2 shows the results for a 16-long tube running at 48kHz and a 32-long tube running at 96kHz (which should both have the same resonances) with varying widths. As can be seen from the data, thinner tubes have resonances that are further away from their predicted value. Also, the tube with a higher node density has a more accurate response for a tube of the same dimensions (for example the 32x16 tube has a resonant mode of 527.3Hz, whilst the equivalent 16x8 tube has a resonant mode of 521.5Hz).

It is thought that this is caused by two phenomena. Firstly, the dispersion error mentioned in section 4.1.2 is worse for waves travelling parallel to the waveguides, which is the direction of travel of the main resonant mode. The dispersion error reduces with higher sampling rates, which would explain the above observation. Secondly, simple reflective boundaries only have the
9. Evaluation of the Model

<table>
<thead>
<tr>
<th>Width</th>
<th>$f_0$ (Hz)</th>
<th>Width</th>
<th>$f_0$ (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>530.3</td>
<td>Predicted</td>
<td>530.3</td>
</tr>
<tr>
<td>4</td>
<td>501</td>
<td>4</td>
<td>495.1</td>
</tr>
<tr>
<td>5</td>
<td>509.8</td>
<td>5</td>
<td>503.9</td>
</tr>
<tr>
<td>6</td>
<td>515.6</td>
<td>6</td>
<td>509.8</td>
</tr>
<tr>
<td>8</td>
<td>521.5</td>
<td>8</td>
<td>515.6</td>
</tr>
<tr>
<td>10</td>
<td>524.4</td>
<td>10</td>
<td>521.5</td>
</tr>
<tr>
<td>16</td>
<td>530.3</td>
<td>16</td>
<td>527.3</td>
</tr>
<tr>
<td>32</td>
<td>530.3</td>
<td>32</td>
<td>530.3</td>
</tr>
</tbody>
</table>

Table 9.2: First resonance for a straight tube open at one end (corrected boundaries).

correct response for waves approaching them at $90^\circ$, which could explain the influence of the tube width.

It is interesting to note that the ‘half-length’ boundaries provide more predictable resonant modes are not susceptible to the effects of the tube width. However, it is unknown what other problems they may suffer from.

### 9.3 Two Concatenated Tubes

The effect of a transition from a small tube to a large tube, such as in a very simple approximation of the vocal tract, on the first few formants is well understood and can be predicted mathematically (see section 2.1.5. Comparison of the output produced by various impedance map schemes to the predicted response should provide an initial indication of their suitability
as a mapping scheme.

It was decided to test the response of the model with both the original ‘half-length’ boundaries and the corrected ones as the original appeared to be less susceptible to dispersion error but could suffer from other unknown issues.

For the purposes of the test, the size was chosen to be 32 waveguides long (so that it would be effectively equivalent to two equal 15.5-long tubes with the original boundaries and two 16-long tubes with the corrected boundaries).

Tables 9.3 and 9.4 show the resulting formants for a range of scenarios, for the original and corrected boundaries respectively. The terms ‘static’ and ‘absorb’ refer to the implementation of the air/wall boundaries, whilst 2xSRF signifies a sample rate factor of 2 (see section 7.2.2) and a mesh size that is doubled as to have the same physical dimensions. For the half-length boundaries, using a sampling rate factor of 2 (thus causing the mesh to run at an effective rate of 96kHz) and doubling the mesh size produces slightly different results. This however, is to be expected due to the error introduced by the mesh boundaries implementation. A tube 64-long is actually only 63-long which is equivalent to two 31.5-long tubes - double the length of the 15.5-long tubes should obviously be 31. Table 9.3 shows there is also a good match for the predicted values for this case, but this is a fundamental problem with this implementation.

### Flat 1D

For the half-length boundaries, the location of the first two formants are in the expected locations. Although not exact, the accuracy of the response with the corrected boundaries is still relatively good.
## Evaluation of the Model

### Cases

<table>
<thead>
<tr>
<th>Cases</th>
<th>$f_0$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted</td>
<td>477.7</td>
<td>617.1</td>
</tr>
<tr>
<td>Predicted (2xSRF)</td>
<td>470.2</td>
<td>607.3</td>
</tr>
<tr>
<td><strong>Flat 1D</strong></td>
<td><strong>477.5</strong></td>
<td><strong>615.2</strong></td>
</tr>
<tr>
<td>Flat 1D (2xSRF)</td>
<td>471.7</td>
<td>606.5</td>
</tr>
<tr>
<td>Raised Cosine ($r^2$)</td>
<td>407.2</td>
<td>808.6</td>
</tr>
<tr>
<td>Raised Cosine ($r^3$)</td>
<td>436.5</td>
<td>832.0</td>
</tr>
<tr>
<td>Raised Cosine ($r^4$)</td>
<td>445.0</td>
<td>837.9</td>
</tr>
<tr>
<td>Flat 2D (Static)</td>
<td>521.5</td>
<td>585.9</td>
</tr>
<tr>
<td>Flat 2D (Static, 2xSRF)</td>
<td>506.8</td>
<td>577.2</td>
</tr>
<tr>
<td>Flat 2D (Absorb)</td>
<td>436.5</td>
<td>539.1</td>
</tr>
<tr>
<td>Shared Area (Static)</td>
<td>521.5</td>
<td>673.8</td>
</tr>
<tr>
<td><strong>Shared Area (Absorb)</strong></td>
<td><strong>477.5</strong></td>
<td><strong>615.2</strong></td>
</tr>
<tr>
<td>Shared Area (Abs, 2xSRF)</td>
<td>471.7</td>
<td>606.5</td>
</tr>
</tbody>
</table>

Table 9.3: First two resonances for two concatenated tubes (original boundaries).
9. Evaluation of the Model

<table>
<thead>
<tr>
<th>Cases</th>
<th>$f_0$</th>
<th>$f_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Hz)</td>
<td>(Hz)</td>
</tr>
<tr>
<td>Predicted</td>
<td>462.5</td>
<td>597.5</td>
</tr>
<tr>
<td>Flat 1D</td>
<td>454.1</td>
<td>585.94</td>
</tr>
<tr>
<td>Raised Cosine ($r^2$)</td>
<td>392.6</td>
<td>685.6</td>
</tr>
<tr>
<td>Flat 2D (Absorb)</td>
<td>430.7</td>
<td>524.4</td>
</tr>
<tr>
<td>Flat 2D (Static)</td>
<td>509.8</td>
<td>571.3</td>
</tr>
<tr>
<td>Shared Area</td>
<td>465.8</td>
<td>600.6</td>
</tr>
<tr>
<td>Shared Area (Abs, 2xSRF)</td>
<td>465.8</td>
<td>600.1</td>
</tr>
<tr>
<td>Shared Area (Static)</td>
<td>506.9</td>
<td>656.3</td>
</tr>
</tbody>
</table>

Table 9.4: First two resonances for two concatenated tubes (corrected boundaries).

**Raised Cosine**

The location of the two resonances for the raised cosine impedance map are inaccurate and increasing the power of ‘$r$’ used to calculate the impedance appears to increase the value of both formant by a similar amount.

**Flat 2D**

The distance between the first two formants is consistently smaller than predicted. This may be caused by the change in area being represented by both the impedance change and the actual width change, thus causing a greater perceived change in areas. Using the ‘absorption’ implementation of the air/wall boundaries produce slightly more accurate resonances than the static method.
Shared Area

The shared area impedance map using static-style reflective boundaries produces resonances that are higher than required. Interestingly, the use of the ‘absorption’ implementation produces very accurate formant locations, including with the corrected boundaries, despite the straight tube case not being accurate. Running the mesh at twice the sampling rate provides a similarly good match. This is promising for the usefulness of the scheme.

9.4 Measured Area Functions

The next step was to observe the response of the various mapping schemes compared to that measured in the plexiglass tubes. Three tubes of suitably varied length and topologies were selected. A sample rate of 48kHz and a mesh sample rate factor of 2 was chosen which results in waveguides approximately 5mm long, which is the same width as a single plexiglass section. Thus by setting the model to be of the right length, it was possible to apply the area function to the mesh without any re-sizing required and have a model of the same length as the tubes.

The frequency response plots for the various impedance maps applied to the above area functions are included in appendix A. The data used to produce the plots can be found on the accompanying CD alongside MATLAB functions to easily load the data into arrays ready for plotting.

The tube labelled ‘RichAB’, figure 9.1, is 35 sections long. The frequency response produced by the mapping schemes are compared to the measured response in figures A.1 to A.4.
9. Evaluation of the Model

The tube labelled ‘JackAB’, figure 9.2, is 39 sections long. The frequency response produced by the mapping schemes are compared to the measured response in figures A.5 to A.8.

The tube labelled ‘FredAB’, figure 9.3, is 33 sections long. The frequency response produced by the mapping schemes are compared to the measured response in figures A.9 to A.12.
Figure 9.3: FredAB area function.

**Flat 1D**

This scheme continues to accurately reproduce the first formant, but the location of the following formants is not especially accurate. This confirms the scheme’s usefulness as a benchmark other schemes can be compared against without offering a viable solution in itself.

**Flat 2D**

This scheme seems to produce consistently lower formants than expected. Alongside the previous unsatisfactory results, it does not appear that the ‘flat 2D’ map is worthy of further investigation.

**Raised Cosine**

The ‘raised cosine’ scheme continues to provide a poor match with the expected response. This seems to suggest that the ‘free path’ down the centre of the mesh is not the most appropriate way of designing an impedance map scheme.
Shared Area

This produces almost exactly the same formants as the ‘flat 1D’ scheme, but with some extra resonances. These are probably due to the varying width of the mesh. This suggests that there is some potential in this mapping scheme, as it appears to profit from the extra dimension whilst providing a similarly accurate low frequency response as the 1D impedance applied to the 2D mesh.

Air-to-Wall Boundaries

Potential problems with boundaries highlighted in section 6.2.1 appear to have been founded. In one of the cases (richAB), the frequency response for both the ‘flat 2D’ and ‘shared area’ schemes with absorption inside the wall varies wildly from the two alternative boundary implementations. It is thought that this may occur due to large amounts of energy being lost at ‘corner’ nodes. The reason only one of the three cases was affected could be due to the ‘richAB’ tube being the model with the most variation in width (it is also the only one with part of the air section only one waveguide wide).
Chapter 10

Conclusions and Recommendations

This chapter offers conclusions regarding the completion of objectives of the project and suggestions on possible extensions.

10.1 Project Objectives

Survey literature to establish existing and alternative approaches.

The literature survey was detailed as part of the initial report before the approach explored was settled on. The 2D digital waveguide mesh with impedance map approach was selected for its potential in real-time applications whilst offering a more natural output than simpler 1D models. Furthermore, the available expertise within both departments involved ensured that useful progress could be made and that any outcome could contribute to a greater body of work.
Identify any issues to be fixed and potential improvements to be made with Mullen’s [1] model and establish which ones should be implemented.

It was decided that re-implementing the software in a more flexible and reusable manner was required due to the rigid nature of the existing available software. Software issues regarding array indexes, and later regarding boundaries, were identified in the original implementation. The potential impact of these errors was identified as a point of interest and so it was decided that an existing impedance map scheme should be re-implemented and tested. Suggestions regarding new impedance map schemes involving air-to-wall boundaries and clearer definition of the space represented by the 2D mesh were selected for investigation. The ‘shared area’ mapping scheme was developed as a way of incorporating these ideas and potentially provide a more accurate and natural response.

Re-implment the model in software to meet requirements.

The model was successfully implemented, as demonstrated by the appropriate response to simple tube arrangements. A piece of software that allows easy testing of impedance map schemes with a range of area functions and excitation sources was designed and described. Although still containing a few known bugs, the software was successfully used as a testing mechanism when implementing impedance map schemes. It was found, from personal experience, that the graphical display was especially useful as a verification tool. The model itself was also used in alternate programs which demonstrates its potential for re-use.

Document the functionality of the source code with sufficient detail to allow re-use of all or parts of the software.
The functionality and structure of the software has been described to facilitate use and extension of the software in further work. Detailed documentation of the software API was produced by including appropriate comments in the source code that could then be used to generate the HTML documentation using Doxygen [16], which can be accessed on the accompanying CD. The software is already being used by a PhD student at the University of York as part of his research in the field of voice synthesis, and has received positive initial feedback regarding its potential usefulness.

Establish the relationship between area function and impedance map to get predictable formants in the output.

The intended impedance map schemes were implemented and compared against mathematically predicted formants for simple cases and against the measured frequency response of the more complex area functions of the plexiglass tubes. Although, no scheme provided a convincing solution, the ‘shared area’ scheme showed the most promise as it offers similar early formants to the ‘flat 1D’ scheme but is believed to have more potential regarding its higher frequency response, which was not examined as part of this work. The air-to-wall boundaries using absorption appear to have a drastic negative impact on the resulting frequency response for more complex tubes, which is probably due to the increased number of ‘corner’ nodes which were predicted to be problematic. Aforementioned issues with mesh boundaries reduced the time available for testing of the impedance map scheme and so some of the intended testing was not carried out and is left as recommendations for further work.
10.2 Future Work

This final section contains some suggestions regarding further testing and developments, potential new directions and related areas of research which this work would benefit from.

10.2.1 Further Testing

As mentioned above, unforeseen issues limited the available time for model testing and further evaluation of the model could be carried out. A further nine plexiglass tube area functions with associated frequency response measurements are available on the accompanying CD alongside the three used in the evaluation. Further testing could be carried out using real vocal tract data or against predicted formants for simulated area functions [21]

The linear impedance transition feature, although successfully implemented, was not formally tested and so the accuracy of the frequency changes remain unknown.

10.2.2 Model Development

Other Impedance Map Schemes

Although the experimental data shown here shows that the ‘raised cosine’ scheme is not especially successful, there may still be scope for successful schemes that do not require an air-to-wall impedance boundary. One approach could be to apply the 1D impedance to the centre of the mesh and a high impedance near the wall and apply a raised cosine profile transition between the two.
10. Conclusions and Recommendations

**Improved Mesh Boundaries**

One issue highlighted by this work is the need for improved boundary conditions over the simple reflective boundary. The effect of the boundaries has a significant enough impact on the resulting formant location to warrant further development.

**Improved Air-to-Wall Boundaries**

Although the air-to-wall boundaries are shown to work in principle, a systematic way of setting the wall impedance to produce the required reflections should be established.

### 10.2.3 Software Extensions

**Phrases**

One possibility offered by the linear transition between area functions is to string together many such transitions to form complete phrases. For a more accurate and natural transition, it is likely that it would be necessary to provide a number of transitory states and use the linear transitions between those.

**Complex Shapes**

Rather than simply using one dimensional area functions, it may be possible to apply more complex shapes and calculate a relevant area function. This would require slight modifications to the DynamicMesh and ImpedanceMap classes and entirely new implementations of impedance mapping schemes as
inherited classes of ImpedanceMap. However, the main processing section of DynamicMesh could remain the same.

**Splitting the Area**

After establishing an appropriate way to treat areas of ‘wall’ within the model, it should be possible to insert such areas in the middle of the mesh to explore the possibility of forming additional sounds such as /l/ and /t/.
Appendix A

Frequency Response Plots
Figure A.1: Frequency response for tube ‘richAB’ (1)
Figure A.2: Frequency response for tube ‘richAB’ (1)
Figure A.3: Frequency response for tube ‘richAB’ (3)
Figure A.4: Frequency response for tube ‘richAB’ (4)
Figure A.5: Frequency response for tube ‘jackAB’ (1)
Figure A.6: Frequency response for tube ‘jackAB’ (1)
Figure A.7: Frequency response for tube ‘jackAB’ (3)
Figure A.8: Frequency response for tube ‘jackAB’ (4)
Figure A.9: Frequency response for tube ‘fredAB’ (1)
Figure A.10: Frequency response for tube ‘fredAB’ (1)
Figure A.11: Frequency response for tube ‘fredAB’ (3)
Figure A.12: Frequency response for tube ‘fredAB’ (4)
Appendix B

Cross-sectional Area Calculations

This appendix details the necessary mathematical background required to calculate the cross-sectional area represented by a particular waveguide in a 2-D model of concatenated tubes, as described in section 6.

First we discuss how to calculate the shaded area in figure B.1, which also shows the relevant areas, distances and angles required. \( A_{\text{half}} \) is identified on the opposite side of the circle for clarity.

From the area of a circle, given by:

\[
A_{\text{circle}} = \pi r^2 \tag{B.1}
\]

it is trivial to calculate the area covered by half the circle:

\[
A_{\text{half}} = \frac{\pi r^2}{2} \tag{B.2}
\]
B. Cross-sectional Area Calculations

![Figure B.1: Slice near the circle’s diameter.]

Using geometry/vudu it can be shown that the area of the top segment can be calculated using:

\[ A_{\text{segment}} = \frac{r^2}{2} (\theta - \sin \theta) \]  

(B.3)

where the angle \( \theta \) can be obtained using:

\[ \theta = 2 \sin^{-1} \frac{\sqrt{r^2 - h^2}}{r} \]  

(B.4)

where \( h \) is the perpendicular distance of the chord away from the center, as shown in figure B.1.

Finally, the shaded area is the difference between the larger area (half the circle) and the smaller area (the segment).

\[ A_{\text{shaded}} = A_{\text{half}} - A_{\text{segment}} \]  

(B.5)

Using a similar ideas, we can calculate the area of a slice away from the center of the circle, as illustrated in figure B.2. Once again, \( A_{\text{large}} \) is identified on the opposite side of the circle for clarity.
A\textsubscript{large} and A\textsubscript{small} are two segments, and therefore their areas can be calculated using equation B.3. The shaded area is then given by:

\[ A_{\text{shaded}} = A_{\text{large}} - A_{\text{small}} \quad (B.6) \]

There is a special case where the required shaded area is across the diameter, as illustrated in figure B.3. In this case, the same technique as above can be used to calculated half the area and then it is simply a question of doubling the result, i.e.:

\[ A_{\text{shaded}} = 2(A_{\text{half}} - A_{\text{segment}}) \quad (B.7) \]
Figure B.3: A slice across the circle’s diameter.
Appendix C

Impedance Map

Implementation

C.1 Flat 1D

```cpp
void FlatOneDimension::calculateImpedance() {
  // Fill array for length-wise waveguides with 1D impedance
  for (int x=0; x<NX_; ++x) {
    // params_[0] contains the power of r and equals 2 by default
    StkFloat yLoc = pow(sqrt(area_[x]), params_[0]);
    for (int y=0; y<NY_; ++y)
      z_[x][y] = yLoc;
  }
  // Convert to from the point of view of each junction
  // Calculate width-wise waveguide impedance as average of surrounding ones
  for (int x=0; x<NX_-1; ++x) {
    for (int y=0; y<NY_-1; ++y) {
      yNorth_[x][y] = (z_[x][y] + z_[x+1][y]) / 2;
      yEast_[x][y] = z_[x+1][y];
      ySouth_[x][y] = (z_[x][y] + z_[x+1][y]) / 2;
      yWest_[x][y] = z_[x][y];
    }
  }
  // Find maximum and minimum value and generate impedance arrays
```
C. Impedance Map Implementation

```cpp
findYMaxMin();
makeZFromY();
```

C.2 Raised Cosine

```cpp
void RaisedCosine::calculateImpedance() {
    StkFloat zMin = 0;

    // Find largest area and calculate zMin;
    for (int x=0; x<NX_; ++x) {
        if (area_[x] > zMin)
            zMin = area_[x];
    }
    zMin = pow(sqrt(1/zMin), params_[0]);

    // Calculate raised cosine function
    for (int y=0; y<NY_+1; ++y) {
        cosFunction_[y] = (1+cos(2*PI*y/NY_))/2;
    }

    // Fill impedance array
    for (int x=0; x<NX_; ++x) {
        StkFloat zLoc = pow(sqrt(1/area_[x]), params_[0]);
        for (int y=0; y<NY_-1; ++y) {
            z_[x][y] = zMin + (zLoc-zMin)*cosFunction_[y];
        }
    }

    // Convert to from the point of view of each junction
    // Calculate width-wise waveguide impedance as average of surrounding ones
    for (int x=0; x<NX_-1; ++x) {
        for (int y=0; y<NY_-1; ++y) {
            zNorth_[x][y] = (z_[x][y+1] + z_[x][y+2] + z_[x+1][y+1] + z_[x+1][y+2]) / 4;
            zEast_[x][y] = z_[x+1][y+1];
            zSouth_[x][y] = (z_[x][y] + z_[x][y+1] + z_[x+1][y] + z_[x+1][y+1]) / 4;
            zWest_[x][y] = z_[x][y+1];
        }
    }

    // Find maximum and minimum value and generate admittance arrays
```
C. Impedance Map Implementation

```cpp
findZMaxMin();
makeYFromZ();
}

C.3 Flat 2D

```cpp
void FlatTwoDimension::calculateImpedance() {
    // d = (c * sqrt(2)) / fs
    StkFloat edgeLength = (SOS_ROOT2) / (Stk::sampleRate() * factor_);
    int center = (NY_−2) / 2;

    // Reset flags in case new area function is applied
    for (int x=0; x≤NXMAX−1; ++x)
        for (int y=0; y≤NYMAX−1; ++y)
            insideWall_.[x][y] = false;

    // Generate 1D impedance array
    for (int x=0; x<NX_; ++x) {
        StkFloat zLoc = pow(sqrt(1/area_[x]), params_[0]);
        for (int y=0; y<NY_; ++y)
            z_[x][y] = zLoc;
    }

    // Convert to from the point of view of each junction
    // Calculate width-wise waveguide impedance as average of surrounding ones
    for (int x=0; x<NX_; ++x) {
        for (int y=0; y<NY_; ++y) {
            zNorth_[x][y] = (z_[x][y] + z_[x+1][y]) / 2;
            zEast_[x][y] = z_[x+1][y];
            zSouth_[x][y] = (z_[x][y] + z_[x+1][y]) / 2;
            zWest_[x][y] = z_[x][y];
        }
    }

    findZMaxMin();

    // Lengthwise wall waveguides (x index with reference to area function)
    for (int x = 0; x < NX_; ++x) {
        StkFloat rSquared = area_[x] / Pi;
        StkFloat radius = sqrt(rSquared);
```

110
// Do bottom half only, then use symmetry
for (int y = center - 1; y >= 0; --y) {
    StkFloat nodeDist = (center - y) * edgeLength;
    if (nodeDist >= radius) {
        if (x < NX - 1) {
            insideWall[x][y] = true;
            // if (y == 0 || insideWall[x][y+1])
            zWest[x][y] = zMax * 10000;
        }
        if (x != 0) {
            insideWall[x-1][y] = true;
            // if (y == 0 || insideWall[x-1][y+1])
            zEast[x-1][y] = zMax * 10000;
        }
    }
}

// Use symmetry...
if (x < NX - 1) {
    zWest[(NY-2) - y] = zWest[x][y];
    insideWall[x][(NY-2) - y] = insideWall[x][y];
} else if (x != 0) {
    zEast[x-1][(NY-2) - y] = zEast[x-1][y];
    insideWall[x-1][(NY-2) - y] = insideWall[x-1][y];
}

// Widthwise wall waveguides (x index refers to node)
for (int x = 0; x < NX - 1; ++x) {
    // Do bottom half only, then use symmetry
    for (int y = 0; y <= center; ++y) {
        // Case where node is part of the wall
        if (insideWall[x][y]) {
            zSouth[x][y] = zMax * 10000;
            if (y!=0) {
                zNorth[x][y-1] = zMax * 10000;
            }
        }
    }
}

// Use symmetry...
C. Impedance Map Implementation

```cpp
zNorth.\[ \text{x}|(\text{NY}−2) − \text{y}] = zSouth.\[ \text{x}|\text{y}];
\text{if } (y! = 0)
\quad zSouth.\[ \text{x}|(\text{NY}−2) − (y−1)] = zNorth.\[ \text{x}|y−1];
\}
// Find maximum and minimum value and generate admittance arrays
findZMaxMin();
makeYFromZ();
```

C.4 Shared Area

```cpp
void SharedArea::calculateImpedance() {
  // d = (c * sqrt(2)) / fs
  StkFloat edgeLength = (SOS_ROOT2) / (Stk::sampleRate() * factor_);
  int center = (NY−2) / 2;

  // Reset flags in case new area function is applied
  for (int x=0; x < NX_MAX−1; ++x)
    for (int y=0; y < NY_MAX−1; ++y)
      insideWall.\[ \text{x}|\text{y}] = false;

  // Lengthwise waveguides (x index with reference to area function).
  for (int x = 0; x < NX_; ++x) {
    StkFloat rSquared = area.\[ \text{x}] / Pi;
    StkFloat radius = sqrt(rSquared);

    // Center waveguide is a special case
    // If area is represented only by central waveguide
    if (radius < edgeLength) {
      if (x < NX_−1)
        zWest.\[ \text{x}|center] = AIRCONST /
        (area.\[ \text{x}] * pow(sqrt(area.\[ \text{x}] / Pi), params_[0] − 2));
      if (x != 0)
        zEast.\[ \text{x−1}|center] = AIRCONST /
        (area.\[ \text{x}] * pow(sqrt(area.\[ \text{x}] / Pi), params_[0] − 2));
    }
    // If area is represented by multiple waveguides
    else {
      StkFloat angle = 2 * asin(sqrt(rSquared − pow(edgeLength/2,2)) / radius);
```

112
### C. Impedance Map Implementation

```cpp
StkFloat area = area[x] - (rSquared) * (angle - sin(angle));
if (x < NX-1)
    zWest[x][center] = AIRCONST / (area * pow(sqrt(area/Pi), params[0] - 2));
if (x != 0)
    zEast[x-1][center] = AIRCONST / (area * pow(sqrt(area/Pi), params[0] - 2));
}
// Rest of mesh: Do bottom half only, then use symmetry
for (int y = 0; y < center; ++y) {
    StkFloat nodeDist = (center - y) * edgeLength;
    // If node is inside the wall
    if (nodeDist >= radius) {
        if (x < NX-1) {
            zWest[x][y] = WALLCONST * params[1];
            insideWall[x][y] = true;
        }
        if (x != 0) {
            zEast[x-1][y] = WALLCONST * params[1];
            insideWall[x-1][y] = true;
        }
    }
    // If node is within the tube
    else {
        StkFloat largeAngle = 2 * asin(sqrt(rSquared -
            pow(nodeDist - edgeLength/2, 2)) / radius);
        StkFloat largeArea = (rSquared/2) * (largeAngle - sin(largeAngle));
        StkFloat smallAngle = 0;
        if (nodeDist + edgeLength/2 < radius)
            smallAngle = 2 * asin(sqrt(rSquared -
                pow(nodeDist + edgeLength/2, 2)) / radius);
        StkFloat smallArea = (rSquared/2) * (smallAngle - sin(smallAngle));
        StkFloat effectiveArea = largeArea - smallArea;
        if (x < NX-1)
            zWest[x][y] = AIRCONST /
```
C. Impedance Map Implementation

    (effectiveArea * pow(sqrt(effectiveArea/Pi),
                  params_[0] - 2));

    if (x != 0)
        zEast_[x-1][y] = AIRCONST /
        (effectiveArea * pow(sqrt(effectiveArea/Pi),
                  params_[0] - 2));
    }

    // Use symmetry...

    if (x < NX-1) {
        zWest_[x][(NY-2) - y] = zWest_[x][y];
        insideWall_[x][(NY-2) - y] = insideWall_[x][y];
    }

    if (x != 0) {
        zEast_[x-1][(NY-2) - y] = zEast_[x-1][y];
        insideWall_[x-1][(NY-2) - y] = insideWall_[x-1][y];
    }

    }

    }

    // Use symmetry...

    if (x < NX-1) {
        zWest_[x][(NY-2) - y] = zWest_[x][y];
        insideWall_[x][(NY-2) - y] = insideWall_[x][y];
    }

    if (x != 0) {
        zEast_[x-1][(NY-2) - y] = zEast_[x-1][y];
        insideWall_[x-1][(NY-2) - y] = insideWall_[x-1][y];
    }

    }

    }

    // Widthwise waveguides (x index refers to node)

    for (int x = 0; x < NX-1; ++x) {
        // Do bottom half only, then use symmetry

        for (int y = 0; y <= center; ++y) {
            // Case where node is part of the wall

            if (insideWall_[x][y]) {
                zSouth_[x][y] = WALLCONST * params_[1];
                if (y!=0) {
                    zNorth_[x][y-1] = WALLCONST * params_[1];
                }
            }
            else {
                StkFloat nodeDist = (center - y) * edgeLength;
                // Area on West side
                StkFloat rSquared = area_[x] / Pi;
                StkFloat radius = sqrt(rSquared);

                StkFloat largeAngle = 2 * asin(sqrt(rSquared - pow(nodeDist, 2)) / radius);
                StkFloat largeArea = (rSquared/2) * (largeAngle - sin(largeAngle));

                StkFloat smallAngle = 0;
            }
        }
    }

    }
if (nodeDist + edgeLength < radius)
    smallAngle = 2 * asin(sqrt(rSquared - 
                       pow(nodeDist + edgeLength, 2)) / radius);

StkFloat smallArea = (rSquared/2) * (smallAngle - sin(smallAngle));

StkFloat effectiveArea = largeArea - smallArea;

zSouth[x][y] = AIRCONST /
               (effectiveArea * pow(sqrt(effectiveArea/Pi),
                           params[0] - 2));

// Area on East side
rSquared = area_[x+1] / Pi;
radius = sqrt(rSquared);

largeAngle = 2 * asin(sqrt(rSquared - pow(nodeDist, 2)) / radius);
largeArea = (rSquared/2) * (largeAngle - sin(largeAngle));

smallAngle = 0;
if (nodeDist + edgeLength < radius)
    smallAngle = 2 * asin(sqrt(rSquared - pow(nodeDist + edgeLength, 2)) / radius);
smallArea = (rSquared/2) * (smallAngle - sin(smallAngle));
effectiveArea = largeArea - smallArea;

// Take average of two impedances
zSouth[x][y] = (zSouth[x][y] + AIRCONST/
               (effectiveArea * pow(sqrt(effectiveArea/Pi),
                           params[0] - 2))) / 2;

if (y!=0)
    zNorth[x][y-1] = zSouth[x][y];
}

// Use symmetry...
zNorth[x][(NY-2) - y] = zSouth[x][y];
if (y!=0)
    zSouth[x][(NY-2) - (y-1)] = zNorth[x][y-1];
}
C. Impedance Map Implementation

152 }  
153 // Find maximum and minimum value and generate admittance arrays
154 findZMaxMin();
155 makeYFromZ();
156 }
Appendix D

Verifying Boundaries
Implementation

This section contains the three different versions of the main sample scattering/delay passes that were implemented in an attempt to confirm that the behaviour observed using full-length boundaries was a property of the boundaries rather than a coding error. This is included as a record of the work carried out, and all three implementations are in fact equivalent and produced the same frequency response in experimental tests.

For the purposes of this appendix, all code pertaining to input/output, absorption within the wall or transition from one impedance map to another has been removed. Only the main scattering and delay loops have been kept in order to illustrate the differences in implementation.

The first example includes the boundary reflections as part of the delay loop. Initially, the incoming wave component for junctions next to boundaries was calculated directly from the outgoing component. However, this in fact cause the reflection to arrive at the junction one sample earlier. To rectify
this, extra arrays were created that would hold the reflection for one sample and thus remove the error.

```cpp
D. Verifying Boundaries Implementation

StkFloat DynamicMesh :: computeSample()
{
    // Scattering pass
    for(int x=0; x<NX-1; ++x) {
        for(int y=0; y<NY-1; ++y) {
            // Calculate junction Pressure
            pressure_[x][y] = 2*( (pNorthPlus_[x][y] * YNorth(x,y)
                + pEastPlus_[x][y] * YEast(x,y)
                + pSouthPlus_[x][y] * YSouth(x,y)
                + pWestPlus_[x][y] * YWest(x,y)
            ) /
            ( YNorth(x,y)
            + YEast(x,y)
            + YSouth(x,y)
            + YWest(x,y)
            );

            // Calculate outgoing waves
            pNorthMinus_[x][y] = pressure_[x][y] - pNorthPlus_[x][y];
            pEastMinus_[x][y] = pressure_[x][y] - pEastPlus_[x][y];
            pSouthMinus_[x][y] = pressure_[x][y] - pSouthPlus_[x][y];
            pWestMinus_[x][y] = pressure_[x][y] - pWestPlus_[x][y];
        }
    }

    //Delay pass (update incoming waves for next sample)
    //Includes reflections at boundaries
    for(int x=0; x<NX-1; ++x) {
        for(int y=0; y<NY-1; ++y) {
            if(x==0) {
                pWestPlus_[x][y] = pWestBoundary_[y];
                pWestBoundary_[y] = glottalRef_ * pWestMinus_[x][y];
            }
            else
                pWestPlus_[x][y] = pEastMinus_[x-1][y];

            if(x==NX-2) {
                pEastPlus_[x][y] = pEastBoundary_[y];
                pEastBoundary_[y] = lipRef_ * pEastMinus_[x][y];
            }
        }
    }
}
```
D. Verifying Boundaries Implementation

```c
StkFloat DynamicMesh :: computeSample(
{

    // Scattering pass
    for (int x=0; x<NX-1; ++x) {
        for (int y=0; y<NY-1; ++y) {
            // Calculate junction Pressure
            pressure_[x][y] = 2*( ( pNorthPlus_[x][y] * YNorth(x,y) + pEastPlus_[x][y] * YEast(x,y) + pSouthPlus_[x][y] * YSouth(x,y) + pWestPlus_[x][y] * YWest(x,y) ) / ( YNorth(x,y) + YEast(x,y) ));

    }
}
```

After it was suggested that this approach could cause confusion due to the inclusion of some signal scattering within the delay pass, and thus was more likely to contain errors, it was decided to re-implement the function. This next version puts the reflection equations within the scattering pass and explicitly holds both an incoming and outgoing component for the boundary nodes.
D. Verifying Boundaries Implementation

```plaintext
// Calculate outgoing waves
pNorthMinus[x][y] = pressure[x][y] - pNorthPlus[x][y];
pEastMinus[x][y] = pressure[x][y] - pEastPlus[x][y];
pSouthMinus[x][y] = pressure[x][y] - pSouthPlus[x][y];
pWestMinus[x][y] = pressure[x][y] - pWestPlus[x][y];

// Reflection at Boundaries (previously done in delay pass)
if (x==0)
    pWestBoundaryMinus[y] = glottalRef * pWestBoundaryPlus[y];
if (x==NX-2)
    pEastBoundaryMinus[y] = lipRef * pEastBoundaryPlus[y];
if (y==0)
    pSouthBoundaryMinus[x] = wallRef * pSouthBoundaryPlus[x];
if (y==NY-2)
    pNorthBoundaryMinus[x] = wallRef * pNorthBoundaryPlus[x];

// Delay pass (update incoming waves for next sample)
for (int x=0; x<NX-1; ++x) {
    for (int y=0; y<NY-1; ++y) {
        if (x==0) {
            pWestPlus[x][y] = pWestBoundaryMinus[y];
            pWestBoundaryPlus[y] = pWestMinus[x][y];
        } else
            pWestPlus[x][y] = pEastMinus[x-1][y];

        if (x==NX-2) {
            pEastPlus[x][y] = pEastBoundaryMinus[y];
            pEastBoundaryPlus[y] = pEastMinus[x][y];
        }
        else
            pEastPlus[x][y] = pWestMinus[x+1][y];

        if (y==0) {
            pSouthPlus[x][y] = pSouthBoundaryMinus[x];
            pSouthBoundaryPlus[x] = pSouthMinus[x][y];
        } else
            pSouthPlus[x][y] = pSouthBoundaryMinus[x];
```
D. Verifying Boundaries Implementation

```cpp
else
    pSouthPlus_[x][y] = pNorthMinus_[x][y-1];

if(y==NY-2) {
    pNorthPlus_[x][y] = pNorthBoundaryMinus_[x];
    pNorthBoundaryPlus_[x] = pNorthMinus_[x][y];
} else
    pNorthPlus_[x][y] = pSouthMinus_[x][y+1];
} })

As final confirmation, the model indexing system was modified so that the main array includes the boundary nodes. The pressure at boundaries is computed using an alternative junction pressure equation before computing the output components normally. It is worth noting that although travelling wave components are computed and carried over between boundaries, the junction pressure equation only uses the relevant component and so the extra components have no influence. This was done to simplify the source code. Note that this version is not compatible with the current implementation of the impedance mapping schemes.

StkFloat DynamicMesh :: computeSample()
{
    // Scattering pass (indexing includes boundary nodes)
    for(int x=0; x<NX+1; ++x) {
        for(int y=0; y<NY+1; ++y) {
            if ((x==0 && y==0) || (x==NX && y==0) ||
                (x==0 && y==NY) || (x==NX && y==NY))
                continue;
            // Boundary junction pressures computed explicitly
            if (x==0) //WEST
                pressure_[x][y] = (1 + glottalRef_ ) * pEastPlus_[x][y];
            else if (x==NX) //EAST
                pressure_[x][y] = (1 + lipRef_ ) * pWestPlus_[x][y];
            else if (y==0) //SOUTH
                pressure_[x][y] = (1 + wallRef_ ) * pNorthPlus_[x][y];
            else if (y==NY) //NORTH
                pressure_[x][y] = (1 + wallRef_ ) * pSouthPlus_[x][y];
```
D. Verifying Boundaries Implementation

else {
    // Calculate junction Pressure
    pressure[x][y] = 2 * ( pNorthPlus[x][y] * YNorth(x,y)
    + pEastPlus[x][y] * YEast(x,y)
    + pSouthPlus[x][y] * YSouth(x,y)
    + pWestPlus[x][y] * YWest(x,y)
    ) /
    ( YNorth(x,y)
    + YEast(x,y)
    + YSouth(x,y)
    + YWest(x,y)
    )
    }

    // Calculate outgoing wave components
    pNorthMinus[x][y] = pressure[x][y] - pNorthPlus[x][y];
    pEastMinus[x][y] = pressure[x][y] - pEastPlus[x][y];
    pSouthMinus[x][y] = pressure[x][y] - pSouthPlus[x][y];
    pWestMinus[x][y] = pressure[x][y] - pWestPlus[x][y];

    // Delay pass (update incoming waves for next sample)
    for (int x=0; x<NX+1; ++x) {
        for (int y=0; y<NY+1; ++y) {
            if (x!=0)
                pWestPlus[x][y] = pEastMinus[x-1][y];
            if (y!=0)
                pEastPlus[x][y] = pWestMinus[x+1][y];
            if (y!=NY)
                pSouthPlus[x][y] = pNorthMinus[x][y-1];
            if (y!=0)
                pNorthPlus[x][y] = pSouthMinus[x][y+1];
        }
    }
}
References


