Acoustic pulse reflectometry for the measurement of musical wind instruments

Thesis

Acoustic pulse reflectometry for the measurement of musical wind instruments

David Brian Sharp

PhD

University of Edinburgh

1996

Abstract

The bore profile and input impedance of a musical wind instrument provide valuable information about its acoustical properties. The time domain technique of acoustic pulse reflectometry can be used to measure the input impulse response of a tubular object, such as a wind instrument, from which both its bore profile and input impedance can be calculated.

In this thesis, after a discussion of the theory of acoustic pulse reflectometry, the operation of a practical reflectometer is described and measurements of input impulse response, bore profile and input impedance are investigated.

In general, the experimentally measured input impulse response of a tubular object contains a DC offset which must be removed for accurate bore reconstruction. A new, faster method of determining the DC offset is introduced which doesn't require prior knowledge of the object's dimensions.

The bore profile of a test object, calculated by applying a lossy reconstruction algorithm to its input impulse response (after removal of the DC offset), is found to agree with directly measured radii to within 0.05mm. Various brass instrument reconstructions of similar accuracy are presented.

An input impedance curve, calculated from the input impulse response of the test object, is found to have peak frequencies which agree with those of a theoretical curve to within 0.7% (a considerably better agreement than when a standard frequency domain measurement technique is used). Impedance curves of various brass instruments are presented.

Bore reconstructions are used to confirm the presence, and in certain cases, the positions of leaks in instruments. For the special case of a leaking cylinder, the impedance curve is successfully used to calculate the size of the leak.

Finally, a method is investigated which allows the practical reflectometer to measure longer objects than previously possible.

Declaration

I declare that this thesis has been composed by me and that the work is my own.

.....

Acknowledgements

I would like to thank Dr.D.Murray Campbell and Professor Clive Greated, my supervisors, for their enthusiastic support and guidance during the course of this research programme.

Thanks are also due to Stephen Malloch, Arnold Myers and Dr.Raymond Parks, the other members of the University of Edinburgh Musical Acoustics Group, for their helpful comments.

Special thanks are due to Dr.Noam Amir not only for the provision of his work on reconstruction algorithms prior to publication but also for his invaluable advice and useful suggestions. Comments made by Dr.Ian Marshall and Dr.Kees Nederveen were also appreciated.

Financial support was provided by the Engineering and Physical Sciences Research Council. The University of Edinburgh Physics Department also assisted by funding international conference visits and providing access to electronic and mechanical workshop facilities.

The instruments measured during the course of the study were made available by the Edinburgh University Collection of Historic Musical Instruments.

I would like to thank my flatmates, both past and present, for providing a pleasant and sociable home environment. Peter Boyle and Iain Rankin also provided a great deal of assistance on both computational and physics-related matters.

Finally, I would like to express my gratitude to my girlfriend Carol, to my parents, to my brother Philip and to the rest of my family for their constant moral support.

Contents

Chapter 1

Introduction

In the acoustical study of musical wind instruments, two types of measurement have proved particularly valuable. One is the measurement of the bore profile; the other is the evaluation of the input impedance, defined as the ratio of the acoustic pressure to the air volume flow rate at the entrance to the instrument. These two quantities are closely related and, in principle, one can be calculated from the other. In practice, however, complications from features such as sideholes limit the accuracy with which the impedance can be calculated from the bore profile.

For many years, standard methods existed for measuring the two quantities. The bore profile was measured directly with accurate tools such as calipers. However, in many cases it was not possible to access the whole length of the instrument, leaving the measured profile incomplete. The input impedance of the instrument was measured in the frequency domain. To make the measurement, it was first necessary to measure the volume flow rate. The pressure was then measured at the entrance to the instrument and divided by the volume flow rate to give the input impedance [Backus 1974, Backus 1976, Pratt et al 1977, Causse et al 1984]. The method yielded excellent results but was very time-consuming, requiring the initial volume flow rate measurement and then a pressure measurement at each frequency of interest.

Recently, the time domain technique of acoustic pulse reflectometry has begun

to be applied to wind instruments. A pulse of sound is injected into the instrument and the resulting reflections are analysed to give the input impulse response, from which the cross-sectional area as a function of axial distance and the input impedance can be calculated. The advantages of acoustic pulse reflectometry are that it only requires a pressure measurement (removing the need for a volume flow rate calibration measurement) and that it is non-invasive (allowing inaccessible bore sections to be measured). As a pulse technique, all frequencies are dealt with simultaneously making acoustic pulse reflectometry less time-consuming than the frequency domain method.

1.1 History of acoustic pulse reflectometry

Acoustic pulse reflectometry was originally developed as a seismological technique for the observation of stratifications in the earth's crust. The earth's crust is made up of layers of different types of rock. When an approximately impulsive pressure wave is produced by a source such as dynamite and used to probe the crust, reflections are generated due to impedance differences between the layers. These reflections return to the surface where they are recorded and, because of the impulsive nature of the excitation, termed the input impulse response. Ware and Aki [1969] developed a solution to the inverse problem of calculating the reflection coefficients of the layer boundaries from the input impulse response. The solution assumed lossless propagation through the layers. From the boundary reflection coefficients and the impedance of the surface layer of rock, the impedances of deeper layers could be calculated.

In the early seventies, the medical research team led by Sondhi noted the potential of acoustic pulse reflectometry as a method for measuring airway dimensions. Previous attempts to measure the cross-sectional area of an airway as a function of axial distance had been carried out in the frequency domain, usually by measuring the input impedance at the mouth and then (having assumed the vocal tract length) using resonances to estimate the area profile [Schroeder 1967, Mermelstein 1967]. In

a mainly theoretical paper, Sondhi and Gopinath [1971] described how, by applying a sound pulse to the airway under investigation and recording the reflections at the lips, the area profile of the airway could be calculated. The calculation, although mathematically complex, did not require any assumption about vocal tract length. In their treatment, as in the Ware-Aki treatment, losses in the airway were not taken into account. An attempt to include the effect of losses was made in a later paper [Sondhi and Resnick 1983] but was only applicable if the losses were assumed to have simple forms.

Jackson et al [1977] published area profiles of excised dog tracheas and lungs measured using a pulse reflectometer. They modelled the airway as a series of discontinuously joined cylindrical segments of equal lengths but differing cross-sectional areas (and, hence, differing impedances). The problem of measuring the airway dimensions was thus reduced to one of finding the areas of the individual segments (an analagous problem to that of determining the impedances of the individual layers of rock in the earth's crust). The design of the reflectometer was such that a sound pulse created by a spark discharge was applied to the airway via a source tube. Reflections generated at the boundaries between the cylindrical segments returned from the airway and were recorded by a microphone embedded in the wall of the source tube part of the way along its length. After the first of the airway reflections passed the microphone they carried on up the source tube, were reflected by the sound source and returned to the microphone. The source tube ensured that at the microphone the passages of the input pulse, airway reflections and source reflections did not overlap. The input impulse response of the airway was determined by deconvolving the airway reflections with the input pulse shape (the deconvolution was carried out by performing a frequency domain division). The algorithm developed by Ware and Aki was used to calculate the reflection coefficients of the inter-segment boundaries, from which the segment areas were calculated. Although the Ware-Aki algorithm did not take into account losses in

the airway, good area profiles were still achieved because the airways measured were short enough for losses to be insignificant. An expression for the input impedance of an airway in terms of its input impulse response was also derived in the paper but no results were presented. Area profiles of human airway casts were presented in a subsequent paper [Jackson and Olsen 1980].

The first measurements on human patients were carried out by Fredberg et al [1980]. The reflectometer used to make the measurements was a substantially more complicated version of the one used by Jackson et al. A sound pulse (produced this time by a loudspeaker) was again applied to the airway via a source tube. However, in this case the tube was filled with He/O_2 gas. The resultant reflections returned from the airway and were measured by a microphone in the wall of the source tube. The reflections were processed as before and an area profile was calculated using the Ware-Aki algorithm. Between measurements, a valve was opened and the subject breathed in He/O_2 gas. The reason for filling the apparatus and the subject's lungs with He/O_2 was to increase the speed of sound (sound travels almost twice as fast in He/O_2 as it does in air) thereby increasing the bandwidth of the input pulse. With more information from the frequency range in which airway wall non-rigidity could be neglected (i.e. above approximately 1kHz), Fredberg argued that more accurate results should be achieved. On the whole, profile reconstructions made using the He/O_2-filled reflectometer appeared to compare more favourably with X-ray results than those made using an air-filled reflectometer. However, the results were not conclusive and the need for He/O_2 has not been proved. Nonetheless, the system developed by Fredberg et al was tested further and successfully used in clinical trials during the mid-eighties [Brooks et al 1984, Rubinstein et al 1987, Hoffstein et al 1987].

In a conference paper presented at a meeting of the Acoustical Society of America, Benade and Smith [1981] described an early attempt to measure the input impulse response of a musical wind instrument using acoustic pulse reflectometry. An input

sound pulse was produced by a spark discharge at the mouthpiece of a tuba. The tuba reflections, recorded by a microphone also positioned at the mouthpiece, were considered to be the input impulse response of the tuba. No attempt to remove the effects of the input pulse shape by deconvolution was reported. Ayers et al [1985a, 1985b] presented similar work but used a piezoelectric transducer as the sound source instead of a spark discharge.

A large amount of research into the use of acoustic pulse reflectometry for measuring the acoustical properties of musical wind instruments was carried out at the University of Surrey under the supervision of Bowsher.

The group's earliest work was undertaken by Goodwin [1981] and Duffield [1984]. They developed a reflectometer whose design was very similar to the clinical reflectometer of Jackson et al. The only major difference was the longer section of source tube between the sound source and the microphone. This was necessary because the instruments under investigation were longer than the airways measured by Jackson et al. Consequently, the instrument reflections lasted longer than the airway reflections, requiring the extension of the source tube to ensure that they completely passed the microphone before the arrival of the first source reflections. A spark discharge was again used to produce the sound pulse although Duffield did try replacing it with a loudspeaker. Unlike the spark source, the loudspeaker consistently produced pulses of the same shape, enabling the input pulse and instrument reflections to be averaged to improve the signal-to-noise ratio. Unfortunately, attempts to calculate the input impulse response of the instrument by deconvolving the instrument reflections with the input pulse shape were unsuccessful. As a result, the spark source was used because the pulses it produced were more impulse-like and so, to a good approximation, the resulting instrument reflections could be treated as the input impulse response without the need for deconvolution.

Deane [1986] reported more successful attempts at deconvolution. The production

of impulse-like pulses was therefore less critical and the inconsistent spark source was replaced by a loudspeaker. The consistency of the loudspeaker generated pulses allowed averaging of both the input pulse and the instrument reflections to improve the signal-to-noise ratio. Deconvolution then gave the input impulse response.

The research was continued by Smith [1988], Watson and Bowsher [1987, 1988] and Watson [1989]. They presented bore reconstructions of various brass instruments calculated from input impulse responses measured using a pulse reflectometer. Cylindrical symmetry was assumed and, instead of the area, the radius was calculated as a function of axial distance. The reconstruction algorithms employed were the Sondhi algorithm and a non-comprehensive version of the Ware-Aki algorithm (a version which did not take into account all of the multiple reflections within the instrument), neither of which compensated for losses. Although this had not proved significant when calculating the dimensions of a short airway, when calculating the dimensions of a brass instrument where losses were greater, the reconstructed radii should have been increasingly underpredicted with axial distance. However, this underprediction was masked by the presence of a DC offset in the input impulse response which caused the reconstructed profile to either expand or contract spuriously. The procedure described by Watson to remove this DC offset was inappropriate. It involved repeatedly applying the reconstruction algorithm and adjusting the DC value subtracted from the input impulse response until, at an arbitrary position towards the end of the instrument, the reconstructed radius coincided with the measured radius. However, the reconstructed radius was expected to be underpredicted towards the end of the instrument. To force what should have been an underpredicted radius to be equal to a measured radius required the subtracted DC value to be greater than the actual DC offset. Although this overestimation of the DC offset appeared to provide some compensation for losses in the reconstruction, the treatment was not based on a rigorous analysis. Watson also published input impedance curves for a selection of brass instruments (calculated from

their measured input impulse responses using the expression derived by Jackson et al).

In airway measurement work carried out in Edinburgh, Marshall [1990, 1992a, 1992b] took the reflectometer design used by the Surrey group and attempted to improve its portability by reducing the length of the source tube. By sharpening the shape of the input pulse and thus reducing its duration, it was possible to shorten the source tube section between the microphone and airway whilst still maintaining the time separation of the input pulse and airway reflections. Marshall also investigated a mathematical treatment which removed the need for maintaining the time separation of the airway reflections and source reflections (providing a measurement of the loudspeaker input impulse response was made). Thus, the source tube section between the loudspeaker and microphone could be shortened and the input impulse responses of objects of any length could be measured. A reflectometer was built with a source tube length of a few centimetres. Airway profiles calculated using the Ware-Aki algorithm from input impulse response measurements made using the reflectometer were reasonably accurate. However, the accuracy was not as great as that achieved by the longer source tube reflectometer.

Louis et al [1993] described another method for reducing the source tube length. The introduction of a second microphone into the wall of the source tube enabled the separation of the waves probing the airway (the input pulse and any source reflections) from those returning from the airway (the reflection of the input pulse and any source reflections by the airway). This removed the need for maintaining the time separation of the airway reflections and source reflections. As a result, the source tube section between the loudspeaker and microphone could be shortened and the input impulse responses of objects of any length could be measured. Again, good airway profiles were calculated although a correction procedure was necessary.

Until recently, the algorithms used to reconstruct the bore profile of an object from its measured input impulse response did not take into account the effect of losses in

the object. This had not proved significant when reconstructing a short object such as an airway but when a longer object such as a brass instrument was reconstructed it became more important. Amir et al [1995b] suggested the use of a layer-peeling algorithm which was modified to include the effect of losses. The algorithm should have resulted in reconstructed profiles whose radii were correctly predicted at all axial distances. However, as described earlier, the presence of a DC offset in the input impulse response of an object caused the reconstructed profile to either expand or contract spuriously. In this case, Watson's DC offset removal procedure was appropriate because the reconstructed radius at an arbitrary position towards the end of the object was expected to agree with the measured radius at the same position. Hence, forcing the two radii to coincide required the subtracted DC value to be exactly equal to the DC offset. The trumpet and trombone profiles presented were in very good agreement with direct measurements. The algorithm had successfully reconstructed the profiles of instruments whose lengths were such that losses were significant.

1.2 Aims and outline of thesis

The aims of the present work are:

1. to produce a working reflectometer to accurately measure the input impulse response, bore profile and input impedance of a musical wind instrument or other tubular object without prior knowledge of its dimensions.

2. to apply acoustic pulse reflectometry to the problem of detecting leaks in musical wind instruments and in tubular objects in general.

3. to investigate possible methods of measuring longer tubular objects using acoustic pulse reflectometry.

Chapter 2 contains a detailed discussion of the basic theory behind the technique of acoustic pulse reflectometry. The formation of the input impulse response of a

tubular object from internal reflections is described, along with various algorithms for calculating its bore profile and input impedance.

A practical reflectometer is introduced in chapter 3. The experimental determination of the input impulse response of a tubular object is described and measurements made on a stepped tube are presented.

Chapter 4 discusses the reconstruction of the bore profile of a tubular object from its input impulse response. An experimentally determined input impulse response invariably contains a DC offset which must be removed for accurate bore reconstruction. A new method is introduced which determines the DC offset directly from the input impulse response. The method is superior to previous methods because it does not require the repeated application of a reconstruction algorithm nor prior knowledge of the object's dimensions. The accuracies of the Ware-Aki algorithm and the lossy layer-peeling algorithm (recently introduced to acoustic pulse reflectometry by Amir) are investigated by comparing reconstructed profiles of the stepped tube with a directly measured profile. Finally, various brass instrument profiles, reconstructed using the lossy layer-peeling algorithm, are presented.

Chapter 5 discusses the calculation of the input impedance of a tubular object from its input impulse response. It is shown that the resolution of the reflectometry impedance curve is improved by zero-padding the input impulse response. A reflectometry impedance curve of the stepped tube is compared with a curve measured using a standard frequency domain technique and with a theoretically calculated curve. Finally, various brass instrument impedance curves, measured using acoustic pulse reflectometry, are presented.

The effect of a leak in the wall of a tubular object on the object's bore reconstruction and input impedance is investigated in chapter 6. It is shown that a small leak causes the reconstruction to expand spuriously. Comparing the reconstructed profile with a directly measured radius towards the end of the object leads to a method for

identifying the presence and, in certain cases, the position of a leak. Reconstructions of various leaking musical wind instruments are presented to demonstrate the success of the method. Finally, the size of a leak in the wall of a cylindrical pipe is calculated, using a theoretically derived expression, from its reflectometry impedance curve.

The length of object that can be measured using a standard reflectometer is constrained by the formation of source reflections which return to the microphone and limit the time over which the object reflections can be accurately recorded. In chapter 7, methods for measuring longer objects are discussed. A new method is described which involves driving the loudspeaker in such a way as to absorb the incoming object reflections rather than reflect them (i.e. it stops the formation of source reflections). Good absorption is achieved but the reconstruction of a longer stepped tube (calculated from an input impulse response measurement made using the method) is not so successful. The introduction of a slowly varying low amplitude component into the input impulse response causes the reconstruction to slowly expand and contract.

The thesis concludes with chapter 8 which contains final thoughts on the completed work and some ideas for future development.

Chapter 2

Basic theory of acoustic pulse reflectometry

2.1 Introduction

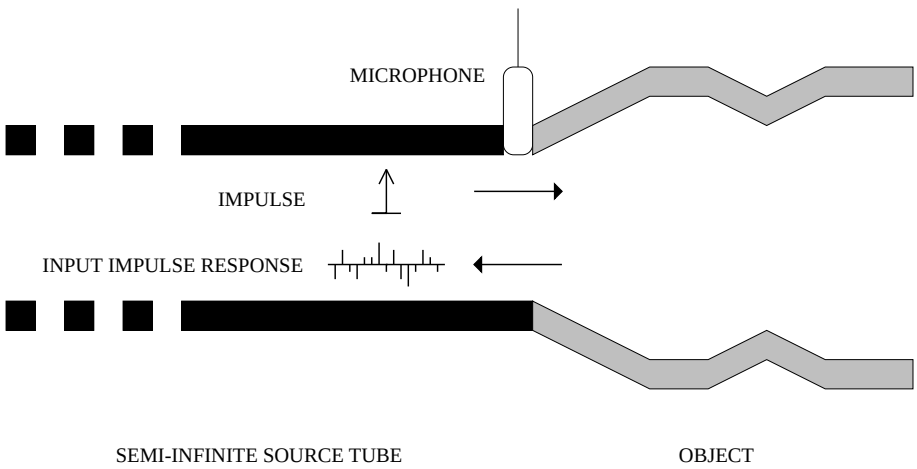

Figure 2.1: *Theoretical pulse reflectometer.*

The idea upon which the technique of acoustic pulse reflectometry is based is relatively simple. Figure 2.1 shows a theoretical reflectometer. An acoustic impulse is fired down a semi-infinite cylindrical tube (called a source tube) into the object under investigation. The impulse undergoes partial reflection and partial transmission at each

change in cross-sectional area along the object's bore, creating a reflection sequence. This sequence returns from the object and travels back up the source tube without further reflection. Its passage is recorded by a microphone embedded in the source tube wall, at the coupling between source tube and object. At this position (the input to the object), the reflection sequence is termed the *input impulse response.* Suitable algorithms enable both the reconstruction of the object's bore profile and the evaluation of its input impedance from the input impulse response.

2.2 Input impulse response

To understand how the bore profile and input impedance of an object can be calculated from its input impulse response, it is first necessary to understand how the input impulse response is created through reflection and transmission within the object.

2.2.1 Single reflection from a single discontinuity

When a planar acoustic pressure wave propagating in an air-filled cylinder encounters a change in cross-sectional area, the associated impedance change causes reflected and transmitted waves to be generated. It can be shown that the ratios of the pressure amplitudes of the reflected and transmitted waves to the pressure amplitude of the incident wave (defined as the reflection and transmission coefficients respectively) depend only on the change in impedance which, in turn, depends only on the change in area [Kinsler et al 1982].

Figure 2.2 shows a semi-infinite cylinder of cross-sectional area S_0 discontinuously joined at $x = 0$ to a second semi-infinite cylinder of cross-sectional area S_1. Consider a pressure wave of frequency ω normally incident on the boundary between the two cylinders,

$$p_0^+ = P_0^+ e^{j(\omega t - kx)} \tag{2.1}$$

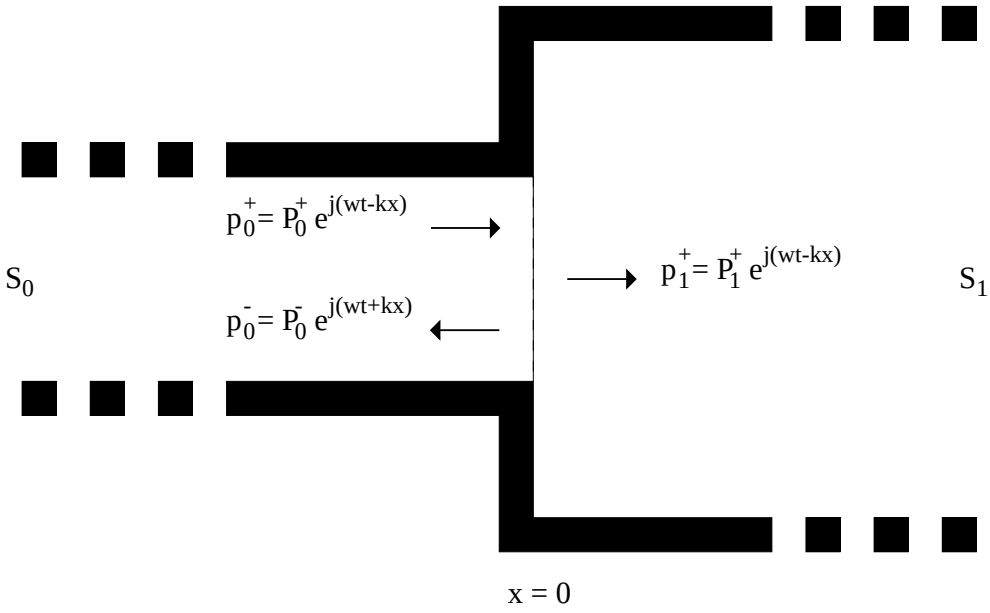

Figure 2.2: *Reflection from a single discontinuity.*

which, when striking the boundary, generates a reflected wave

$$p_0^- = P_0^- e^{j(\omega t+kx)} \tag{2.2}$$

and a transmitted wave

$$p_1^+ = P_1^+ e^{j(\omega t-kx)} \tag{2.3}$$

(where $p_i^{\pm}$ indicates that the pressure wave propagates in cylinder i in the positive/negative x direction). The reflected and transmitted waves propagate down the semi-infinite cylinders without further reflection.

The pressure and velocity must be continuous across the boundary. At $x = 0$, the following continuity equations apply:

$$p_0^+ + p_0^- = p_1^+ \tag{2.4}$$

$$U_0^+ + U_0^- = U_1^+ \tag{2.5}$$

(where $U_i^{\pm}$ is the volume velocity associated with the wave propagating in cylinder i in the positive/negative x direction). The volume velocity has a positive/negative value

depending on whether the wave is propagating in the positive or negative x direction. The volume velocity is defined as the particle velocity $\times$ cross-sectional area.

Dividing equation 2.4 by equation 2.5 gives:

$$\frac{p_0^+ + p_0^-}{U_0^+ + U_0^-} = \frac{p_1^+}{U_1^+} \tag{2.6}$$

The pressure at a surface divided by the volume velocity at that surface is defined as the acoustic impedance Z. Hence,

$$\pm Z = \frac{p}{U^\pm} \tag{2.7}$$

and equation 2.6 becomes:

$$Z_0 \frac{p_0^+ + p_0^-}{p_0^+ - p_0^-} = Z_1 \tag{2.8}$$

(where Z_0 and Z_1 are the acoustic impedances at any cross-section in cylinders 0 and 1 respectively).

Examination of equations 2.1, 2.2 and 2.3 reveals that at $x = 0$, the ratio of the instantaneous pressures of two waves is simply the ratio of their pressure amplitudes. Therefore, rearranging equation 2.8 yields the reflection coefficient $r_{0,1}$ (the ratio of the pressure amplitude of the reflected wave to that of the incident wave).

$$r_{0,1} = \frac{P_0^-}{P_0^+} = \frac{p_0^-}{p_0^+} = \frac{Z_1 - Z_0}{Z_1 + Z_0} \tag{2.9}$$

The transmission coefficient $t_{0,1}$ (the ratio of the pressure amplitude of the transmitted wave to that of the incident wave) can also be expressed in terms of acoustic impedance. Dividing equation 2.4 by p_0^+ gives the transmission coefficient in terms of the reflection coefficient,

$$t_{0,1} = \frac{P_1^+}{P_0^+} = \frac{p_1^+}{p_0^+} = 1 + \frac{p_0^-}{p_0^+} = 1 + r_{0,1} \tag{2.10}$$

and substituting in equation 2.9 gives:

$$t_{0,1} = \frac{2Z_1}{Z_1 + Z_0} \tag{2.11}$$

The ordering of the reflection and transmission coefficient subscripts indicates that the incident wave is propagating from cylinder 0 towards cylinder 1 when it strikes the boundary.

The acoustic impedance at any cross-section S in a cylinder is $Z = \rho c/S$ (where ρ is the density of air and c is the speed of sound in air) so substituting $Z_0 = \rho c/S_0$ and $Z_1 = \rho c/S_1$ into equations 2.9 and 2.11 yields:

$$r_{0,1} = \frac{S_0 - S_1}{S_0 + S_1} \tag{2.12}$$

and

$$t_{0,1} = \frac{2S_0}{S_0 + S_1} \tag{2.13}$$

Thus, the reflection and transmission coefficients depend only on the change in cross-sectional area of the cylinder.

For a wave travelling from cylinder 1 towards cylinder 0, the reflection and transmission coefficients for the boundary can be found by interchanging the subscripts in equations 2.12 and 2.13:

$$r_{1,0} = \frac{S_1 - S_0}{S_0 + S_1} = -r_{0,1} \tag{2.14}$$

and

$$t_{1,0} = \frac{2S_1}{S_0 + S_1} = 1 + r_{1,0} = 1 - r_{0,1} \tag{2.15}$$

Note that the transmission loss experienced by a wave travelling forwards and backwards across the boundary is:

$$t_{0,1}t_{1,0} = (1 + r_{0,1})(1 - r_{0,1}) = 1 - r_{0,1}^2 \tag{2.16}$$

2.2.2 Multiple reflections from multiple discontinuities

In the previous section, the simple case of a sinusoidal pressure wave incident on the discontinuity created by a change in the cross-sectional area of a cylindrical tube, was discussed. In the present section, the case of an impulse incident on a tubular object of varying cross-section will be considered. Plane wave propagation is again assumed.

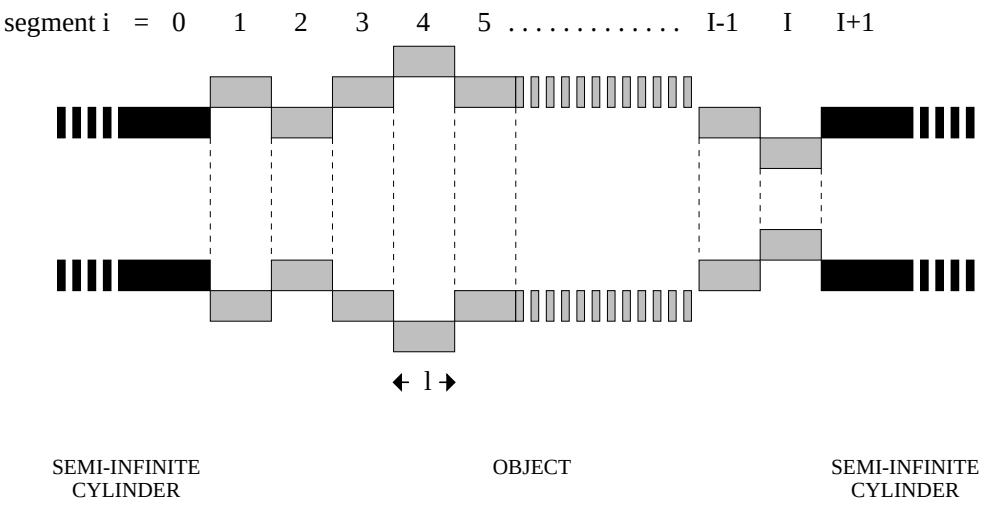

Figure 2.3: *Cylindrically segmented tubular object.*

A tubular object whose cross-sectional area varies with axial distance can be modelled by a series of I discontinuously joined cylindrical segments, each of length l with corresponding two-way travel time $T = 2l/c$ [Marshall et al 1991]. Figure 2.3 shows such an object (consisting of segments numbered from 1 to I), terminated at both ends by semi-infinite cylinders (defined as being the 0th and the (I+1)th cylindrical segments).

Consider a discrete input pressure signal $p^{+}_{0,r}[nT]$ incident on the object from the source tube (i.e. incident on the boundary between segments 0 and 1). The signal will be partially reflected and partially transmitted at each inter-segment boundary. (The notation is similar to the notation used previously, with $p^{\pm}_{i,l/r}[nT]$ representing the contribution, from the wave propagating in the positive/negative x direction, to the total pressure at the left/right end of the ith cylindrical segment at time nT, where

$n = 0, \frac{1}{2}, 1, \frac{3}{2}$ etc.).

For an impulse excitation at the input to the object,

$$p^{+}_{0,r}[nT] = \delta[nT] = \begin{array}{ll} 1 & \text{when } n = 0 \\ 0 & \text{when } n \neq 0 \end{array} \tag{2.17}$$

the signal which returns from the object is the input impulse response,

$$p^{-}_{0,r}[nT] = iir[nT] \tag{2.18}$$

Consider the theoretical situation in which the signal experiences no losses whilst propagating through the cylindrical segments. The input impulse response $iir[nT]$ is then composed of a series of returning impulses, spaced a time T apart (i.e. the input impulse response is only non-zero when n is integer).

For clarity, the history of the impulse as it propagates within the object is displayed in a schematic space-time diagram (figure 2.4). The arrows indicate the direction of propagation (forwards or backwards) in each segment at different times. The pressures at the left and right sides of each segment at different times are also displayed. Note that below the diagonal all points have zero pressure. This is a consequence of the property of causality which states that no backward travelling wave can be present in a segment before a forward travelling wave has reached that segment.

Examination of figure 2.4 reveals that at $t = 0$ the input impulse response $iir[0T]$ is simply the reflected impulse generated when the input impulse strikes the boundary between segment 0 and segment 1 (boundary 0/1). This interface has a reflection coefficient $r_{0,1}$ which (from equation 2.9) implies that

$$iir[0T] = r_{0,1} \tag{2.19}$$

The reflected impulse is referred to as a *primary reflection* because it only undergoes a single reflection before emerging from the object.

At $t = T$, the input impulse response $iir[1T]$ is also composed entirely of a primary reflection. This primary reflection is the fraction of the input impulse which is

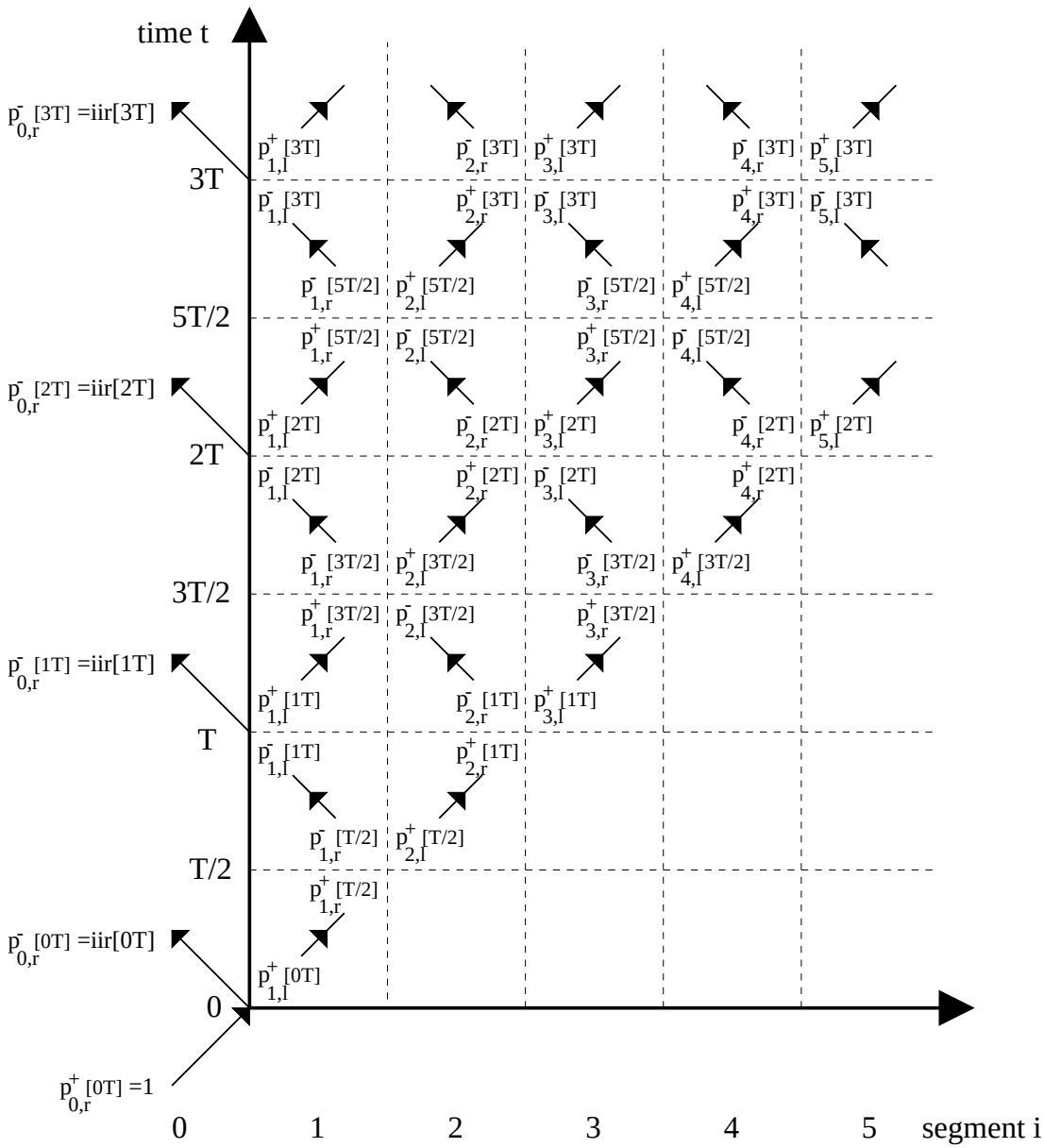

Figure 2.4: *Space-time diagram.*

transmitted forwards across boundary 0/1, reflected at boundary 1/2 and then transmitted backwards across boundary 0/1, before emerging from the object. Hence (using equations 2.9 and 2.16),

$$iir[1T] = r_{1,2}(1 - r_{0,1}^2) \tag{2.20}$$

At $t = 2T$, the input impulse response $iir[2T]$ is composed of both a primary reflection and a *higher order reflection* (a signal which undergoes more than one reflection before emerging from the object). The primary reflection is the fraction of the input impulse which is transmitted forwards across boundaries 0/1 and 1/2, reflected at boundary 2/3 and then transmitted backwards across boundaries 0/1 and 1/2, before emerging from the object. The higher order reflection is the fraction of the input impulse which is transmitted forwards across boundary 0/1, reflected at boundary 1/2, further reflected at boundaries 0/1 and 1/2 and then transmitted backwards across boundary 0/1, before emerging from the object. Hence (using equations 2.9, 2.14 and 2.16),

$$iir[2T] = r_{2,3}(1 - r_{1,2}^2)(1 - r_{0,1}^2) - r_{1,2}^2 r_{0,1}(1 - r_{0,1}^2) \tag{2.21}$$

An expression for the input impulse response at $t = nT$, denoted $iir[nT]$, was derived by Ware and Aki [1969]. Using the z-transform $z = e^{j\omega T}$ to denote the two-way travel time through a segment, they showed that the input impulse response of a tubular object consisting of I segments is given by:

$$iir(I, z) = \frac{z^I B(I, 1/z)}{A(I, z)} \tag{2.22}$$

where A and B are the recursive polynomials:

$$A(I, z) = A(I - 1, z) + r_{I,I+1} z B(I - 1, z) \tag{2.23}$$

$$B(I, z) = r_{I,I+1} A(I - 1, z) + z B(I - 1, z) \tag{2.24}$$

with $A(0, z) = 1$ and $B(0, z) = r_{0,1}$.

$iir(I,z)$ is a polynomial in z representing the impulses, spaced by a time interval T, returning from the object. The coefficient of the nth power of z is the input impulse response at $t = nT$:

$$iir(I,z) = iir[0T] + iir[1T]z + iir[2T]z^2 + ... + iir[nT]z^n + ... \tag{2.25}$$

Using equations 2.22, 2.23 and 2.24 to calculate $iir(I,z)$ and expanding using a binomial series, the individual elements of the input impulse response can be found by gathering together the different powers of z. The complexity of the calculation of $iir(I,z)$ increases rapidly with increasing I.

Clearly, for the theoretical situation of lossless propagation, the input impulse response can be expressed in terms of the segment boundary reflection coefficients, which can be evaluated from the segment areas using the generalised form of equation 2.12:

$$r_{i,i+1} = \frac{S_i - S_{i+1}}{S_i + S_{i+1}} \tag{2.26}$$

That is to say that given the dimensions of the object, its input impulse response can be calculated. This is often termed the *direct problem*.

In a physical situation, the signal will undergo frequency dependent attenuation due to viscous and thermal effects whilst propagating through each cylindrical segment. This causes the impulse to become reduced in amplitude and to develop a tail. Hence, the input impulse response will consist of a series of pulses of finite width. If the pulse widths become greater than T, they will begin to overlap (if the start of a pulse emerges from the object at time t, the end of the pulse will emerge at a time greater than $t+T$ and the pulse will contribute to both $iir[t]$ and $iir[t+T]$). It is thus impractical to extend the Ware-Aki solution of the direct problem to include the effects of signal losses.

2.3 Bore reconstruction

In acoustic pulse reflectometry, it is necessary to solve the *inverse problem* of evaluating the object's dimensions (reconstructing the bore profile) from the measured input impulse response. The input impulse response is digitally recorded at sampling frequency F, resulting in a discrete signal consisting of N samples each spaced a time $T = 1/F$ apart (where N is usually chosen to be a power of 2 to enable the use of FFTs and T is the cylindrical segment two-way travel time described in section 2.2.2). The input impulse response is therefore $iir[nT]$, where n is integer and $0 \leq n < N$.

A solution to the inverse problem for the lossless case, based on the Ware-Aki derivation, is discussed. Like the direct problem, it is impractical to include the effects of losses in this derivation and bore reconstructions made using this method are inaccurate for all but the shortest of objects. In order to compensate for the effect of losses when reconstructing bore profiles, the layer-peeling approach documented by Bruckstein et al [1985] is adopted, as suggested by Amir et al [1995b]. This method is first described for the lossless case (where it gives identical results to those obtained using the Ware-Aki method) and is then extended to incorporate losses.

2.3.1 Ware-Aki method

The Ware-Aki method of bore reconstruction is based on the solution to the direct problem presented in section 2.2.2. Rearrangement of equations 2.19, 2.20 and 2.21 yields expressions for the reflection coefficients of the first three boundaries,

$$r_{0,1} = iir[0T] \tag{2.27}$$

$$r_{1,2} = \frac{iir[1T]}{(1-r_{0,1}^2)} \tag{2.28}$$

$$r_{2,3} = \frac{iir[2T] + r_{0,1}r_{1,2}^2(1-r_{0,1}^2)}{(1-r_{0,1}^2)(1-r_{1,2}^2)} = \frac{iir[2T] + r_{0,1}r_{1,2}iir[1T]}{(1-r_{0,1}^2)(1-r_{1,2}^2)} \tag{2.29}$$

and suitable manipulation of equation 2.22 yields a general expression for the reflection coefficient of the boundary between the ith and $(i+1)$th segments:

$$r_{i,i+1} = \frac{\sum_{m=0}^{i-1} A_m iir[(i-m)T]}{\prod_{m=0}^{i-1}(1-r_{m,m+1}^2)} \tag{2.30}$$

with $r_{0,1} = iir[0T]$. A_m is the coefficient of z^m in the polynomial $A(i-1,z)$, calculated by substituting $i-1$ for I in equation 2.23.

From the calculated reflection coefficients and the source tube area S_0, the cross-sectional areas of the cylindrical segments can be recursively evaluated by rearranging equation 2.26 to give:

$$S_{i+1} = S_i \frac{1-r_{i,i+1}}{1+r_{i,i+1}} \tag{2.31}$$

Assuming cylindrical symmetry, the radii of the cylindrical segments can be calculated and a bore profile constructed. The length of each segment is $l = cT/2$ (the speed of sound in air is taken to be $c = 331.6\sqrt{1+\tau/273}$ m/s, where τ is the air temperature in ^{o}C [Kinsler et al 1982]).

Note that to reconstruct an object consisting of I cylindrical segments, it is only necessary to record the input impulse response up to the time that the primary reflection from the boundary between segments I and $I+1$ arrives back at the microphone. Provided the input impulse response is accurately known over this time period, it is not necessary to wait for all the multiple reflections to decay.

2.3.2 Layer peeling method : Lossless case

The method of reconstruction suggested by Amir et al [1995b] takes a different approach to the Ware-Aki method. It is based on a layer-peeling algorithm which, as the name implies, 'peels off' each cylindrical segment in turn, calculating the segment boundary coefficients as it does so.

The layer peeling algorithm exploits the causality property stated previously; i.e.

no segment can contain a backward travelling wave before a forward travelling wave has reached it. Referring back to figure 2.4, at a boundary between two arbitrary segments i and $i+1$, there are forward and backward travelling waves both to the left of the boundary in segment i, and to the right of the boundary in segment $i+1$. The forward travelling waves in segment $i+1$ are made up of two components; the reflections of the backward travelling waves in segment $i+1$ and the transmissions of the forward travelling waves in segment i. Similarly, the backward travelling waves in segment i are made up of two components; the reflections of the forward travelling waves in segment i and the transmissions of the backward travelling waves in segment $i+1$. However, at time $t = iT/2$ there is no backward travelling wave in segment $i+1$ (causality property). Hence, the backward travelling wave in segment i is simply the reflection of the forward travelling wave in segment i and the boundary reflection coefficient can be calculated. Figure 2.5 shows such an arbitrary inter-segment boundary.

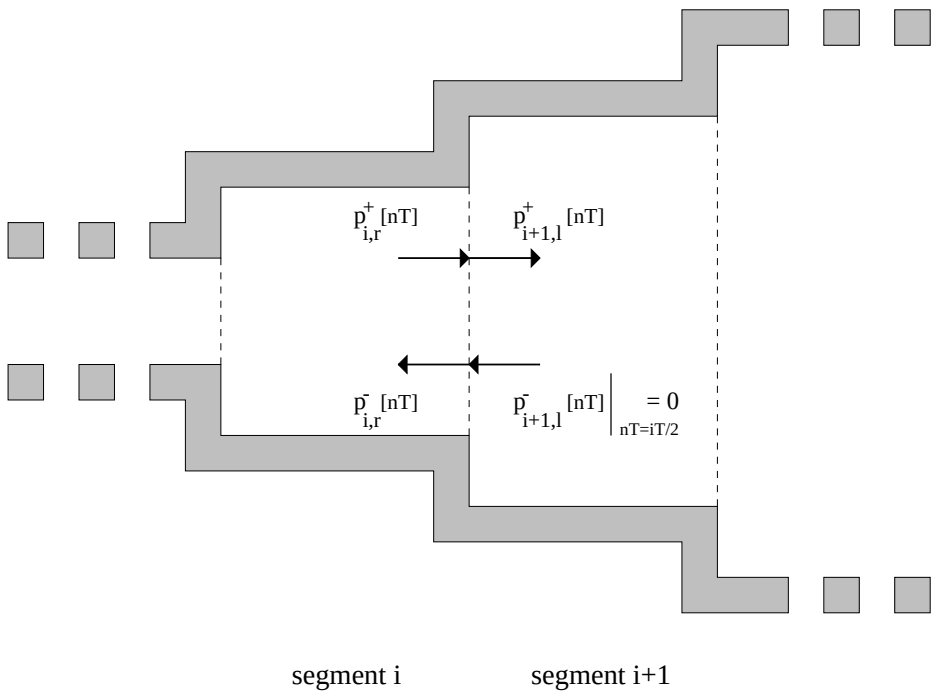

Figure 2.5: *Boundary between two arbitrary segments.*

As before, for an impulse excitation $p^+_{0,r}[nT] = \delta[nT]$ incident on the object from the source tube (i.e. incident on the boundary between segments 0 and 1), the signal which returns is $p^-_{0,r}[nT] = iir[nT]$.

The digitally recorded input impulse response $iir[nT]$ is a discrete signal consisting of N samples spaced a time T apart (i.e. n is integer). For the purposes of the algorithm, the impulse excitation $p^+_{0,r}[nT]$ is also considered to be a discrete signal consisting of N samples spaced a time T apart.

At boundary 0/1, when $t = 0$ there is no backward travelling wave in segment 1. Hence, the backward travelling wave in segment 0 is simply the reflection of the forward travelling wave in segment 0 and:

$$r_{0,1} = \frac{p^-_{0,r}[0T]}{p^+_{0,r}[0T]} = iir[0T] \tag{2.32}$$

Using equation 2.31, the area of the first segment S_1 can be calculated from the reflection coefficient $r_{0,1}$ and the area of the source tube S_0.

Next it is necessary to find the pressures of the forward and backward travelling waves to the right of the boundary. To do this, the following scattering equation is used:

$$\begin{pmatrix} p^+_{1,l}[nT] \\ p^-_{1,l}[nT] \end{pmatrix} = \frac{1}{1-r_{0,1}} \begin{pmatrix} 1 & -r_{0,1} \\ -r_{0,1} & 1 \end{pmatrix} \begin{pmatrix} p^+_{0,r}[nT] \\ p^-_{0,r}[nT] \end{pmatrix} \tag{2.33}$$

Equation 2.33 calculates the pressures of the forward and backward travelling waves at the left side of segment 1 from the pressures of the forward and backward travelling waves at the right side of segment 0. It does so using the previously described relationships between forward and backward travelling waves in adjacent segments, and the reflection and transmission coefficients of boundary 0/1. The calculation is carried out for $n = 0, 1, 2, ..., (N-1)$.

To find the pressures of the forward and backward travelling waves at the right side of segment 1, a delay of $T/2$ must be added to $p^+_{1,l}[nT]$ and subtracted from $p^-_{1,l}[nT]$.

$$p^+_{1,r}\left[\left(n+\frac{1}{2}\right)T\right] = p^+_{1,l}[nT] \tag{2.34}$$

$$p_{1,r}^{-}\left[\left(n-\frac{1}{2}\right)T\right] = p_{1,l}^{-}[nT] \tag{2.35}$$

At boundary 1/2, when $t = T/2$, there is no backward travelling wave in segment 2. Hence, the backward travelling wave in segment 1 is simply the reflection of the forward travelling wave in segment 1 and:

$$r_{1,2} = \frac{p_{1,r}^{-}[T/2]}{p_{1,r}^{+}[T/2]} \tag{2.36}$$

In practice, it is impossible to add a delay of $T/2$ to $p_{1,l}^{+}[nT]$ and subtract a delay of $T/2$ from $p_{1,l}^{-}[nT]$ because both are discrete signals consisting of samples spaced a time T apart. However, except for a shift in the time origin, an equivalent procedure is to subtract a delay of T from $p_{1,l}^{-}[nT]$ and leave $p_{1,l}^{+}[nT]$ unaltered. The time origin shift then requires the reflection coefficient $r_{1,2}$ to be calculated at $t = 0$ rather than at $t = T/2$.

Using equation 2.31, the area of the second segment S_2 can be evaluated from $r_{1,2}$ and the previously calculated area of the first segment S_1.

The process is continued recursively until the entire area profile of the object has been calculated. Assuming cylindrical symmetry, the radii of each of the segments can be calculated and a bore profile produced. Again, the length of each segment is $l = cT/2$ (the speed of sound in air is taken to be $c = 331.6\sqrt{1+\tau/273}$ m/s, where τ is the air temperature in ^{o}C).

As with the Ware-Aki reconstruction method, it is only necessary to know the input impulse response accurately up to the time that the final primary reflection arrives back at the microphone.

2.3.3 Layer peeling method : Lossy case

Amir et al [1995b] showed that extending the layer peeling reconstruction method to incorporate losses is relatively straightforward. In the lossless treatment, each cylindrical segment simply acts as a delay of $T/2$. In reality, whilst propagating through each

segment the signal also experiences frequency dependent attenuation due to viscous and thermal effects. In order to extend the reconstruction algorithm to compensate for these losses, each cylindrical segment is modelled by a digital filter. This filter depends on the length and radius of the segment; at each stage of the algorithm the radius of the following segment is calculated, and the appropriate filter representing losses can be computed. Propagating from the left side of the segment to the right side, the forward travelling wave is simply passed through this filter. The backward travelling wave, however, must be passed through the inverse filter. Hence, to extend the treatment described in section 2.3.2 to include losses the following equations are inserted before equations 2.34 and 2.35 in the reconstruction procedure:

$$p^{+}_{1,l}[nT] = p^{+}_{1,l}[nT] * h_1[nT] \tag{2.37}$$

$$p^{-}_{1,l}[nT] = p^{-}_{1,l}[nT] *^{-1} h_1[nT] \tag{2.38}$$

where the operators $*$ and $*^{-1}$ represent convolution and deconvolution, and $h_1[nT]$ is the time domain filter representing losses in the first segment. To enable convolution and deconvolution, $h_1[nT]$ must have the same sample spacing T and the same total number of samples N as the pressure signals.

Convolution in the time domain is equivalent to multiplication in the frequency domain, so equation 2.37 can be FFT'ed to give:

$$p^{+}_{1,l}[e^{j\theta}] = p^{+}_{1,l}[e^{j\theta}] \times H_1[e^{j\theta}] \tag{2.39}$$

where $p^{+}_{1,l}[e^{j\theta}]$ is the FFT of $p^{+}_{1,l}[nT]$, the frequency domain filter $H_1[e^{j\theta}]$ is the FFT of $h_1[nT]$, and θ is the discretized frequency. This complex multiplication is carried out for each of the N points which make up $p^{+}_{1,l}[e^{j\theta}]$ and $H_1[e^{j\theta}]$. The resultant array is inverse FFT'ed to give the left hand side of equation 2.37.

Deconvolution in the time domain is equivalent to division in the frequency domain, so equation 2.38 can be FFT'ed to give:

$$p^{-}_{1,l}[e^{j\theta}] = \frac{p^{-}_{1,l}[e^{j\theta}]H^{*}_{1}[e^{j\theta}]}{H_1[e^{j\theta}]H^{*}_{1}[e^{j\theta}] + q} \tag{2.40}$$

where $p^-_{1,l}[e^{j\theta}]$ is the FFT of $p^-_{1,l}[nT]$, and $H_1^*[e^{j\theta}]$ is the complex conjugate of $H_1[e^{j\theta}]$. This complex division is carried out for each of the N points which make up $p^-_{1,l}[e^{j\theta}]$ and $H_1[e^{j\theta}]$. The factor q in the denominator is a constraining factor to prevent division by zero. q is set to be small compared with the rest of the denominator. The resultant array is inverse FFT'ed to give the left hand side of equation 2.38.

Problems in calculating a filter to represent losses

The problems associated with finding a time domain filter $h_i[nT]$ to represent the losses in the ith cylindrical segment are discussed in this section. A suitable starting point is the continuous theoretical frequency domain filter representing losses in a cylinder of radius r and length l given by Keefe [1984].

A straight cylinder is characterized by the complex wavenumber γ which determines how a plane wave of frequency ω propagates down the cylinder:

$$\gamma(\omega) = \alpha(\omega) + j\omega/v_p(\omega) \tag{2.41}$$

where $\alpha(\omega)$ is the absorption coefficient and $v_p(\omega)$ is the phase velocity. The absorption coefficent and the phase velocity are given by:

$$\begin{aligned}\alpha &= \frac{\omega}{c}\left[\frac{1}{r_v\sqrt{2}}\left(1+\frac{\gamma-1}{\nu}\right)+\frac{1}{r_v^2}\left(1+\frac{\gamma-1}{\nu}-\frac{1}{2}\frac{\gamma-1}{\nu^2}-\frac{1}{2}\left(\frac{\gamma-1}{\nu}\right)^2\right)\right.\\ &\quad+\frac{1}{r_v^3\sqrt{2}}\left(\frac{7}{8}+\frac{\gamma-1}{\nu}-\frac{1}{2}\frac{\gamma-1}{\nu^2}-\frac{1}{8}\frac{\gamma-1}{\nu^3}-\frac{1}{2}\left(\frac{\gamma-1}{\nu}\right)^2+\frac{1}{2}\frac{(\gamma-1)^2}{\nu^3}\right.\\ &\quad\left.\left.+\frac{1}{2}\left(\frac{\gamma-1}{\nu}\right)^3\right)\right]\end{aligned} \tag{2.42}$$

$$\begin{aligned}\frac{1}{v_p} &= \frac{1}{c}\left[1+\frac{1}{r_v\sqrt{2}}\left(1+\frac{\gamma-1}{\nu}\right)-\frac{1}{r_v^3\sqrt{2}}\left(\frac{7}{8}+\frac{\gamma-1}{\nu}-\frac{1}{2}\frac{\gamma-1}{\nu^2}-\frac{1}{8}\frac{\gamma-1}{\nu^3}\right.\right.\\ &\quad\left.\left.-\frac{1}{2}\left(\frac{\gamma-1}{\nu}\right)^2+\frac{1}{2}\frac{(\gamma-1)^2}{\nu^3}+\frac{1}{2}\left(\frac{\gamma-1}{\nu}\right)^3\right)\right]\end{aligned} \tag{2.43}$$

where $r_v = r\sqrt{\rho\omega/\eta}$, $\nu = \sqrt{\eta C_p/\kappa}$, ρ is the air density, ω is the angular frequency, η is the coefficient of shear viscosity of air, C_p is the specific heat of air at constant

pressure, κ is the thermal conductivity of air, γ is the ratio of the principal specific heats of air, $c = 331.6\sqrt{1+\tau/273}$ m/s is the speed of sound in air and τ is the air temperature.

Assuming that the input is the wave entering the cylinder at the origin, and the output is the wave exiting the cylinder at the far end, the continuous frequency domain filter is given by

$$X(j\omega) = e^{-\gamma(\omega)l} = e^{-\alpha(\omega)l}e^{-j\omega l/v_p(\omega)} \tag{2.44}$$

Inverse Fourier transforming the continuous frequency domain filter $X(j\omega)$ would give a continuous time domain filter $x(t)$ representing the losses along the cylinder. This continuous time domain filter could then be discretized to give the filter $h_i[nT]$ required in equations 2.37 and 2.38. However, analytical calculation of the inverse transform is all but impossible. It is therefore necessary to discretize the frequency domain filter and to use an inverse FFT [Amir et al 1996].

To discretize the frequency domain filter, $X(j\omega)$ is calculated at frequencies $f = 0, \frac{F}{N}, 2\frac{F}{N}, ..., (\frac{N}{2}-1)\frac{F}{N}, \frac{1}{2}F$, creating a vector of $\frac{N}{2}+1$ points. The discrete frequency domain filter must have periodic conjugate symmetry to enable the inverse FFT to be calculated. This is ensured by taking the conjugated reflection of the calculated vector, dropping the first and last points and letting this new vector form the discrete frequency domain filter at frequencies $f = (\frac{N}{2}+1)\frac{F}{N}, (\frac{N}{2}+2)\frac{F}{N}, ..., (N-2)\frac{F}{N}, (N-1)\frac{F}{N}$. Combining the two vectors gives a vector of N samples which is the discrete frequency response $X[e^{j\theta}]$, where θ is the discretized frequency. $X[e^{j\theta}]$ can be inverse FFT'ed resulting in a time domain filter $x[nT]$.

The filter $x[nT]$ is not the filter required in equations 2.37 and 2.38; it is generally beset by ripple. The origin of the ripple can be found by examining equation 2.44 which reveals that the phase is, in general, non-zero at frequency $\frac{1}{2}F$. This results in a phase discontinuity in the discrete frequency domain filter $X[e^{j\theta}]$. Upon transformation, this phase discontinuity causes the ripple in the time domain filter $x[nT]$.

To find a filter suitable for use in equations 2.37 and 2.38, a model must be found which gives a close fit to $x[nT]$ but without the ripple.

Rotating phase model

One solution to the problem of finding a suitable lossy filter model is to apply Papoulis' rotating phase method [Papoulis 1984] to the continuous frequency domain filter $X(j\omega)$ discussed in the previous section. This method is discussed further by Gazengel et al [1995]. The rotating phase method works by forcing the phase to be zero at frequency $f = \frac{1}{2}F$. The following function is formed from the continuous frequency domain filter:

$$X'(j\omega) = X(j\omega) \times e^{j\omega l / v_p(2\pi\frac{1}{2}F)} = e^{-\alpha(\omega)l} e^{(-j\omega l / v_p(\omega) + j\omega l / v_p(2\pi\frac{1}{2}F))} \tag{2.45}$$

Hence, at $f = \frac{1}{2}F$, the phase is zero. If $X'(j\omega)$ is discretized as described in the previous section, the resultant discrete frequency domain filter $X'[e^{j\theta}]$ is continuous in phase and can be inverse FFT'ed to give a ripple-free approximation $x'[nT]$ to the filter $x[nT]$. It should be noted that the rotating phase method does introduce a small time delay. This delay is a fraction of a sample spacing T in duration. Hence, the suitability of $x'[nT]$ as an approximation to $h_i[nT]$ in equations 2.37 and 2.38 improves with the sample rate.

It should also be noted that the rotating phase model is a minimum phase model, meaning that the delay caused by the tube is missing. Other filters may not be minimum phase, in which case the tube delay is already included and need not be added or subtracted explicitly. When using such filters, equations 2.34 and 2.35 should be replaced in the reconstruction procedure by:

$$p^{+}_{1,r}[nT] = p^{+}_{1,l}[nT] \tag{2.46}$$

$$p^{-}_{1,r}[nT] = p^{-}_{1,l}[nT] \tag{2.47}$$

All-pole model

In [Amir et al 1996], Amir proposes using an 40 pole transfer function to model the results (but without the ripple) obtained when Keefe's continuous frequency domain filter is discretized and inverse FFT'ed. The autoregressive method he suggests is documented by Makhoul [1975]. The basic details are outlined below.

1. The autocorrelation of the filter $x[nT]$ (calculated by inverse FFT'ing the filter resulting from the discretization of Keefe's continuous frequency domain filter) is evaluated:

$$R[m] = \sum_{n=0}^{N-1-m} x[nT]x[(n+m)T] \tag{2.48}$$

2. Using the calculated autocorrelation function, the 40 predictor coefficients a_k are found using Durbin's recursive method:

$$E_0 = R[0] \tag{2.49}$$

$$p_m = -\left[R[m] + \sum_{k=1}^{m-1} a_k^{m-1} R[m-k]\right] / E_{m-1} \tag{2.50}$$

$$a_m^{(m)} = p_m \tag{2.51}$$

$$a_k^{(m)} = a_k^{(m-1)} + p_m a_{m-k}^{(m-1)} \qquad 1 \le k \le m-1 \tag{2.52}$$

$$E_m = (1 - p_m^2) E_{m-1} \tag{2.53}$$

Equations are solved recursively for $m = 1, 2, \ldots, 40$. The final solution is given by

$$a_k = a_k^{(40)} \qquad 1 \le k \le 40 \tag{2.54}$$

3. The gain G is calculated next, as follows:

$$G = \sqrt{R[0] + \sum_{k=1}^{40} a_k R[k]} \tag{2.55}$$

4. Finally, the all-pole transfer function,

$$X'(j\omega) = G/(1+\sum_{k=1}^{40} a_k e^{-j\omega k}) \tag{2.56}$$

is calculated on the unit circle (i.e. at frequencies $f = 0, \frac{1}{N}, \frac{2}{N}, \ldots, \frac{N-1}{N}$) to give $X'[e^{j\theta}]$.

5. $X'[e^{j\theta}]$ is then inverse FFT'ed to give $x'[nT]$ a ripple-free approximation to $x[nT]$.

Unlike the rotating phase method, the all-pole method does not introduce a delay of a fraction of a sample spacing, so $x'[nT]$ is a suitable approximation to $h_i[nT]$ in equations 2.37 and 2.38 at all sampling frequencies.

Like the rotating phase model, however, the all-pole model is a minimum phase model, so the delay associated with propagation across the cylindrical segment is not included and must be added explicitly. Hence, equations 2.34 and 2.35 must be included in the reconstruction procedure (not equations 2.46 and 2.47).

2.4 Input impedance

In section 2.2, it was shown that the input impulse response is a series of reflections caused by changes in impedance within the object. It is clear that the input impulse response and the input impedance are closely related. Indeed, the complex input impedance of a tubular object may be directly evaluated from its input impulse response [Jackson et al 1977, Watson and Bowsher 1988, Watson 1989].

Recalling figure 2.1, the microphone records the passage of the impulse (travelling in the positive x direction) and the returning input impulse response (travelling in the negative x direction). Hence, in the time domain,

$$p_{0,r}[nT] = p^+_{0,r}[nT] + p^-_{0,r}[nT] = \delta[nT] + iir[nT] \tag{2.57}$$

and, using equation 2.7,

$$Z_0 \times U_{0,r}[nT] \quad = \quad Z_0 \times \left(U^+_{0,r}[nT] + U^-_{0,r}[nT]\right)$$

$$
\begin{aligned}
&= Z_0 \times \left(\frac{p^+_{0,r}[nT]}{Z_0} - \frac{p^-_{0,r}[nT]}{Z_0} \right) \\
&= p^+_{0,r}[nT] - p^-_{0,r}[nT] \\
&= \delta[nT] - iir[nT] \qquad (2.58)
\end{aligned}
$$

where $p_{0,r}[nT]$ is the total pressure measured by the microphone at time nT, $U_{0,r}[nT]$ is the volume velocity at the microphone at time nT, $p^{\pm}_{0,r}[nT]$ and $U^{\pm}_{0,r}[nT]$ are the pressure and the volume velocity associated with the wave propagating in the positive/negative x direction (at the microphone at time nT), $Z_0 = \rho c/S_0$ is the acoustic impedance at any cross-section in the source tube (ρ is the air density, c is the speed of sound in air, S_0 is the cross-sectional area of the source tube) and $iir[nT]$ is the input impulse response of the object.

In the frequency domain this gives

$$p_{0,r}[e^{j\theta}] = 1 + IIR[e^{j\theta}] \qquad (2.59)$$

$$Z_0 \times U_{0,r}[e^{j\theta}] = 1 - IIR[e^{j\theta}] \qquad (2.60)$$

where $p_{0,r}[e^{j\theta}]$ is the FFT of $p_{0,r}[nT]$, $U_{0,r}[e^{j\theta}]$ is the FFT of $U_{0,r}[e^{j\theta}]$ and $IIR[e^{j\theta}]$ is the FFT of $iir[nT]$. As before, θ represents the discretized frequency.

Hence, the complex input impedance is given by:

$$Z_{in}[e^{j\theta}] = \frac{p_{0,r}[e^{j\theta}]}{U_{0,r}[e^{j\theta}]} = Z_0 \times \frac{1 + IIR[e^{j\theta}]}{1 - IIR[e^{j\theta}]} \qquad (2.61)$$

Chapter 3

Experimental measurement of the input impulse response

3.1 Introduction

The theoretical reflectometer, described in section 2.1 as a means of measuring the input impulse response, clearly cannot be realised practically. Production of an acoustic impulse is impossible and the concept of a semi-infinite tube is solely theoretical. Therefore, to measure the input impulse response of an object, a practical reflectometer must be designed. The construction and operation of such a reflectometer are described in this chapter. Its use in the measurement of input impulse response is discussed and sample results are presented.

3.2 The practical pulse reflectometer and its operation

Figure 3.1 shows a schematic diagram of one of the pulse reflectometers used in the present study. The reflectometer and test object are mounted in an anechoic chamber, with the electronics in an adjoining room (a photograph of the reflectometer and test object is shown in figure 3.2).

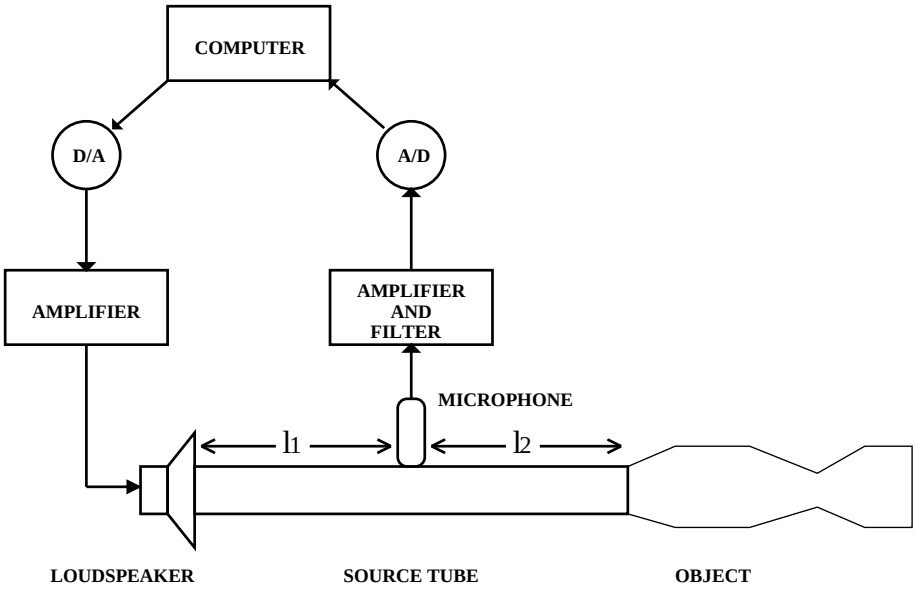

Figure 3.1: *Schematic diagram of pulse reflectometer.*

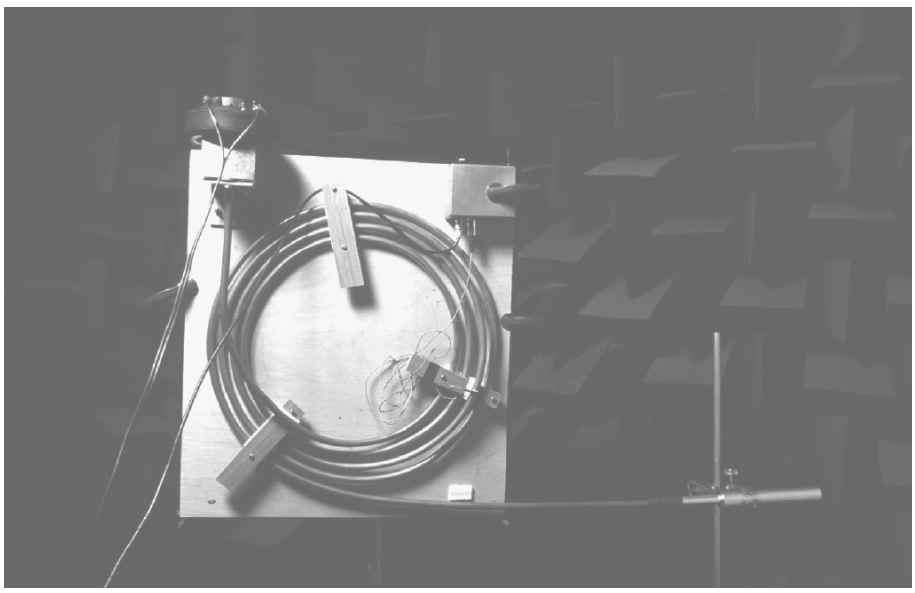

Figure 3.2: *Pulse reflectometer housed in anechoic chamber.*

An electrical pulse of width 80μs and voltage 5V is produced by a 12 bit D/A converter. The D/A converter, which is located on an Iotech DaqBoard 100A data acquisition board inside a Viglen 486DX 66MHz PC, has been customised so that it has a -5V to +5V range instead of a 0V to +5V range. The pulse is amplified by a Pioneer A-119 stereo amplifier, and used to drive a Fane Professional MD2050 compression driver loudspeaker (to reduce 'ringing', the stiffness of the loudspeaker has been increased by inserting a thin layer of cotton wool between the diaphragm and the magnet assembly). The resultant sound pressure pulse travels along a 6.19m long copper source tube (of internal radius 4.8mm and wall thickness 1.2mm) which is clamped to a wooden board in a spiral of approximately 200mm radius. A Knowles microphone embedded part of the way along the tube records the reflections returning from the tubular object under test, which is coupled to the far end of the source tube. The microphone output is amplified by a second amplifier and low-pass filtered (using a Barr and Stroud EF4-03 filter set to 20kHz) to prevent aliasing. The resultant signal is then sampled by a 12 bit A/D converter (using a sampling frequency of 50kHz and a sample length of 1024 points, giving a sample time of 20.48ms) and stored on the PC. The A/D converter is located on the DaqBoard 100A data acquisition board and has a -5V to +5V range.

This procedure is repeated 1000 times and the samples are averaged. This improves the signal-to-noise ratio by a factor of $\sqrt{1000}$, reducing the amplitude of the noise inherent in the signal to less than the resolution of the data acquisition board. Precise time alignment of successive samples is achieved by triggering the sampler from the electrical pulse to the loudspeaker. A delay of 180ms is included before each repetition to ensure all the signal from the previous step has died away.

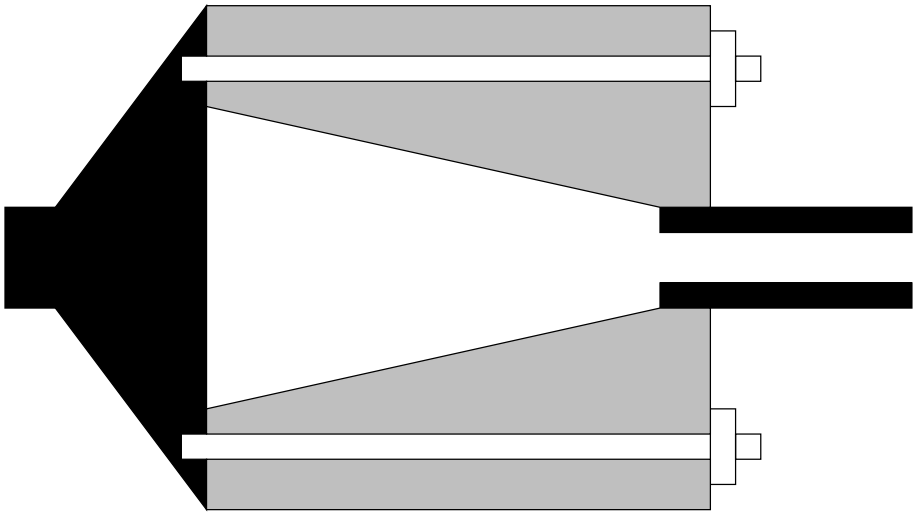

Figure 3.3: *Loudspeaker/source tube coupler.*

3.2.1 Coupling

Loudspeaker/source tube coupling

The loudspeaker is coupled to the source tube with a tapered aluminium coupler (figure 3.3). The downwards taper between the loudspeaker and source tube minimizes reflections within the coupler, reducing any 'ringing' of the input pulse. The coupler is bolted tightly on to the loudspeaker.

Source tube/test object coupling

The source tube is coupled to the test object with a coupler made out of aluminium or black nylon.

Originally a coupler was designed whose internal radius changed sharply from 6.0mm (the external radius of the source tube) to the external radius of the test object (figure 3.4). However, this gave a large discontinuity at the join between the source tube and the object, leading to a large reflection at the start of the object reflections. This reflection should appear as an impulse at the start of the input impulse response.

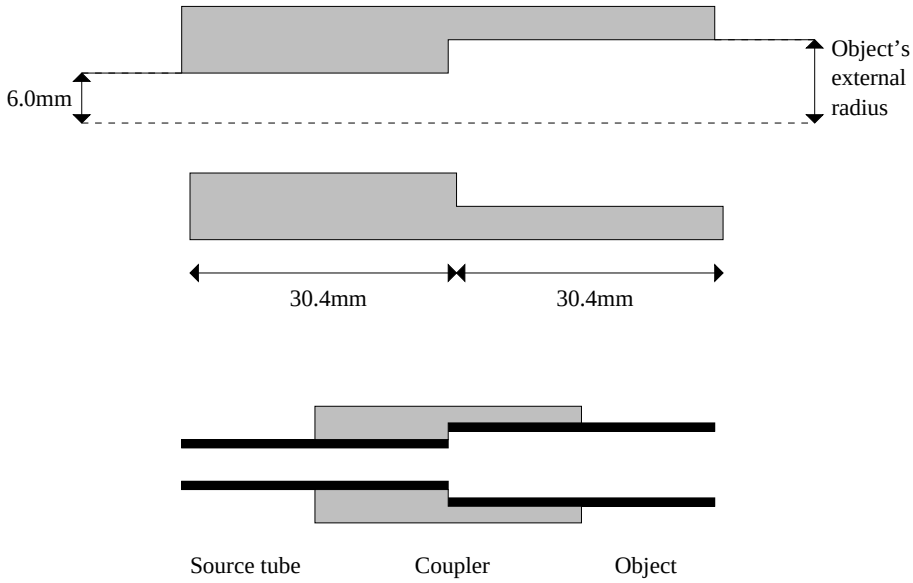

Figure 3.4: *Original source tube/test object coupler.*

However, when the input impulse response of the object was determined experimentally (following the procedure described in section 3.3), the introduction of ripple (an example of the Gibbs phenomenon, also discussed later in the chapter) caused the energy of this impulse to be spread over a finite time. The energy spreading resulted in a 'wrap around' effect with the finite duration reflection divided between the beginning and end of the response.

To ensure the reflection associated with the discontinuous joining of the test object was completely included within the input impulse response, the coupler was redesigned. The modified coupler has an internal radius which changes from 6.0mm to 4.8mm (the internal radius of the source tube), remains at 4.8mm for a length of 50mm and then changes to the external radius of the test object (figure 3.5). This coupler ensures that the large reflection occurs a short time after the object reflections start. Therefore, when the input impulse response is determined, this large reflection is completely included towards the start of the response. Note that the object being measured is now effectively the system consisting of the 50mm long coupler tube and the original test object. This system is smoothly joined to the source tube.

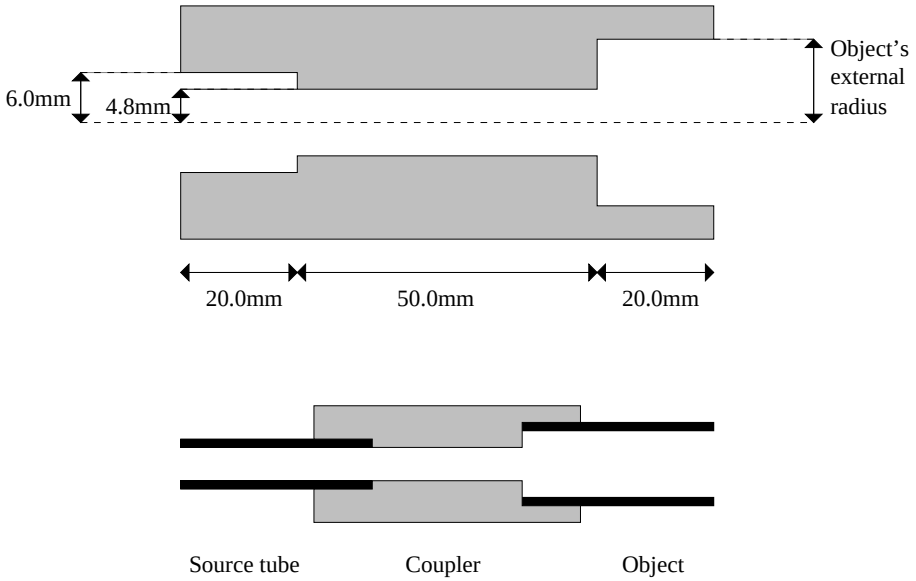

Figure 3.5: *Modified source tube/test object coupler.*

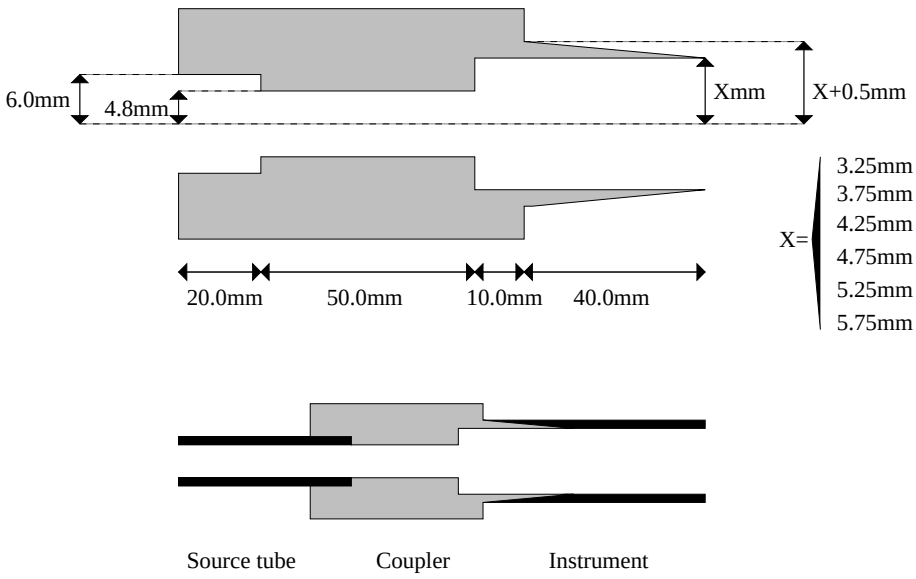

Figure 3.6: *Source tube/brass instrument coupler.*

As it was desired to measure a variety of brass instruments on the reflectometer, it was impractical to make individual couplers for each instrument to be tested. A set of six couplers was designed for use over the whole range of brass instruments (figure 3.6). As before, each of the couplers has an internal radius which changes from 6.0mm to 4.8mm, and remains 4.8mm for a length of 50mm. The internal radius of each of the couplers then changes to one of six different values and remains at this value for a further 50mm, while the external radius tapers downwards. When the correct coupler is chosen, the tapered end enables it to be fitted tightly into the instrument to be measured. Again, note that the object being measured is now effectively the system consisting of the 100mm long coupler and the instrument.

3.3 Deconvolution

For an ideal delta function sound pressure pulse, the reflections measured by the microphone would be the input impulse response of the object under test (or to be more precise, the input impulse response of the source tube section l_2 and the test object). However, the sound pressure pulse is not ideal; to obtain the input impulse response, the reflections are deconvolved with the input pulse shape. The input pulse shape is measured by terminating the source tube with a flat perspex plate of thickness 5mm and recording the reflected pulse [Sondhi and Resnick 1983]. This ensures that both the object reflections and the input pulse have travelled the same path in the source tube and have therefore experienced the same source tube losses. It also ensures that deconvolution yields the input impulse response of the test object alone, without a section of source tube included.

The deconvolution is carried out by performing an FFT on both the sample containing the object reflections and the sample containing the input pulse (each of length 1024 points). To prevent leakage in the frequency domain, both samples must be self-windowing; i.e. the signal must have decayed to zero by the end of the sample. A

complex division of the object reflections by the input pulse is then carried out in the frequency domain. A constraining factor q is added to the denominator to prevent division by zero.

$$IIR[e^{j\theta}] = \frac{R[e^{j\theta}]I^*[e^{j\theta}]}{I[e^{j\theta}]I^*[e^{j\theta}] + q} \tag{3.1}$$

where $R[e^{j\theta}]$ is the transformed object reflections measured at the microphone, $I[e^{j\theta}]$ is the transformed input pulse measured at the microphone, $I^*[e^{j\theta}]$ is the complex conjugate of $I[e^{j\theta}]$ and $IIR[e^{j\theta}]$ is the transformed input impulse response. θ is the discretized frequency.

$IIR[e^{j\theta}]$ is then inverse FFT'ed to give the input impulse response $iir[nT]$ of the object.

3.3.1 Reflectivity of perspex

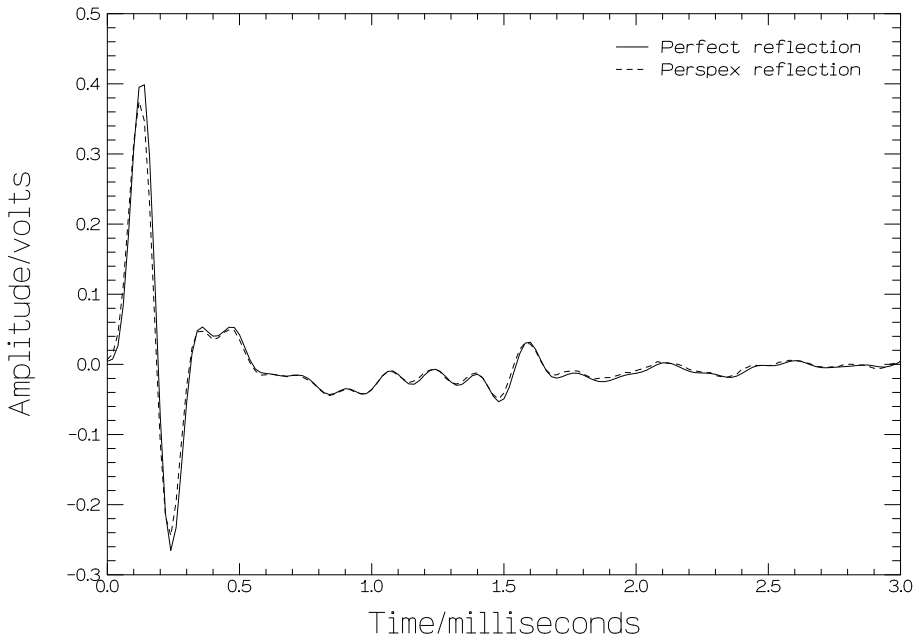

Figure 3.7: *Perspex reflectivity test results.*

The measurement of the input pulse shape through reflection by a flat perspex plate assumes that the perspex provides a perfect reflecting surface. To test that this is the case, the input pulse was sampled both before reflection (as it first passed the

microphone) and after reflection (as it returned to the microphone). The pulse recorded after reflection had travelled an extra $2l_2$=6.18m of source tube. Each sample was 512 points in length and was recorded at a sampling frequency of 50kHz. Convolving the unreflected pulse with an all-pole model filter representing the losses associated with a cylinder of radius 4.8mm and length 6.18m (see section 2.3.3) gave the input pulse shape expected at the microphone after a perfect reflection. Comparing this with the pulse shape measured at the microphone after reflection by the perspex (figure 3.7), it can be concluded that the perspex plate does provide a good approximation to a perfect reflecting surface.

3.4 Physical constraints on the source tube

Comparison of the practical reflectometer described in section 3.2 with the theoretical reflectometer reveals two major differences. The practical reflectometer produces a pulse of finite width (rather than an impulse) which is passed down a finite length source tube (rather than a semi-infinite source tube). These differences explain the required configuration of the practical reflectometer's source tube, where the microphone is embedded a distance l_1=3.10m away from the loudspeaker and a distance l_2=3.09m away from the source tube/object coupling.

The source tube section l_2=3.09m is necessary to ensure that the input pulse has fully passed the microphone before the first of the returning object reflections reaches it. The minimum duration of the input pulse is in practice limited by the requirement that the pulse carries sufficient energy to ensure a good signal-to-noise ratio in the measured reflections.

(The theoretical reflectometer, has l_2=0m because the input pulse is an impulse and takes an infinitesimal amount of time to pass the microphone).

After the object reflections pass the microphone they are further reflected by the loudspeaker. The source tube section l_1=3.10m is necessary to separate the object

reflections from these source reflections. It ensures that once the object reflections reach the microphone, they can be recorded for up to $2l_1/c$ seconds (the time taken to travel the distance from the microphone to the loudspeaker and back) before the source reflections return and contaminate the signal. For the reflectometer under discussion, this time period is approximately 18ms. (In the procedure described in section 3.2, the longer sample time of 20.48ms is possible because the sampling is started 3ms before the arrival of the object reflections at the microphone).

Although the reconstruction algorithms only require the input impulse response to be known up until the final primary reflection from the object, it was stated in section 3.3 that to accurately determine the input impulse response, all of the object reflections must be recorded (i.e. the object reflections must completely pass the microphone within the $2l_1/c$ time period). Hence, although it first appears that the maximum length of object (including coupler) which can be measured is 3.10m, the actual maximum length is shorter than this.

(The theoretical reflectometer has a semi-infinite source tube so after the object reflections pass the microphone they undergo no further reflection and can be recorded accurately for an indefinite period of time).

3.5 Input impulse response measurements of a stepped tube

A series of measurements were made using the stepped tube shown in figure 3.8 as the test object. The tube consists of two cylindrical sections; a 129mm long section of 6.20mm radius discontinuously joined to a 177mm long section of 9.25mm radius. Calipers were used to measure the cylindrical section diameters leading to a reading error of $\pm$0.1mm on each and, hence, an error of $\pm$0.05mm on the radius of each section. The lengths of the cylindrical sections were measured by ruler leading to a

Figure 3.8: *Stepped tube.*

reading error of ± 0.5mm on each. The coupler depicted in figure 3.5 was used to join the stepped tube and the source tube.

Figure 3.10 shows the reflections returning from the stepped tube and coupler when the input pulse (shown in figure 3.9) was injected. Both the input pulse and the reflections were measured at a temperature of $\tau = 16.5^{o}$C. The small reflection at approximately 3.4ms is from the small discontinuity at the joining of source tube and coupler, the reflection at approximately 3.7ms is from the start of the stepped tube, the reflection at approximately 4.4ms is from the step in the tube, and the reflection at approximately 5.5ms is from the end of the tube (which is open to the air). All these reflections are primary reflections; all later reflections are multiple reflections.

To find the input impulse response of the stepped tube and coupler, it is necessary to transform both the input pulse and the stepped tube/coupler reflections to the frequency domain. Figures 3.11 and 3.12 show the spectra of the input pulse and the stepped tube/coupler reflections respectively.

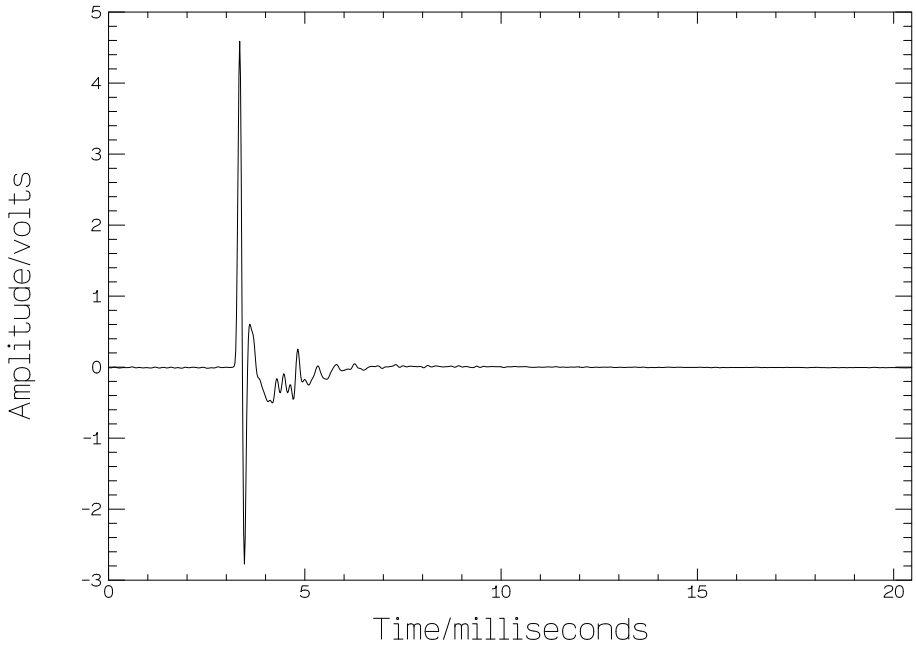

Figure 3.9: *Input pulse.*

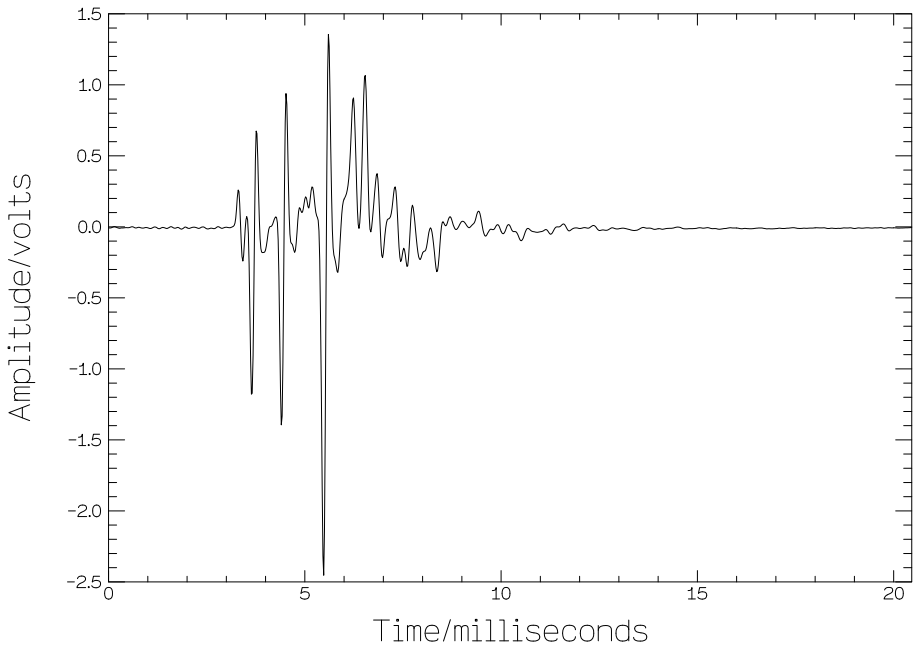

Figure 3.10: *Reflections from stepped tube and coupler.*

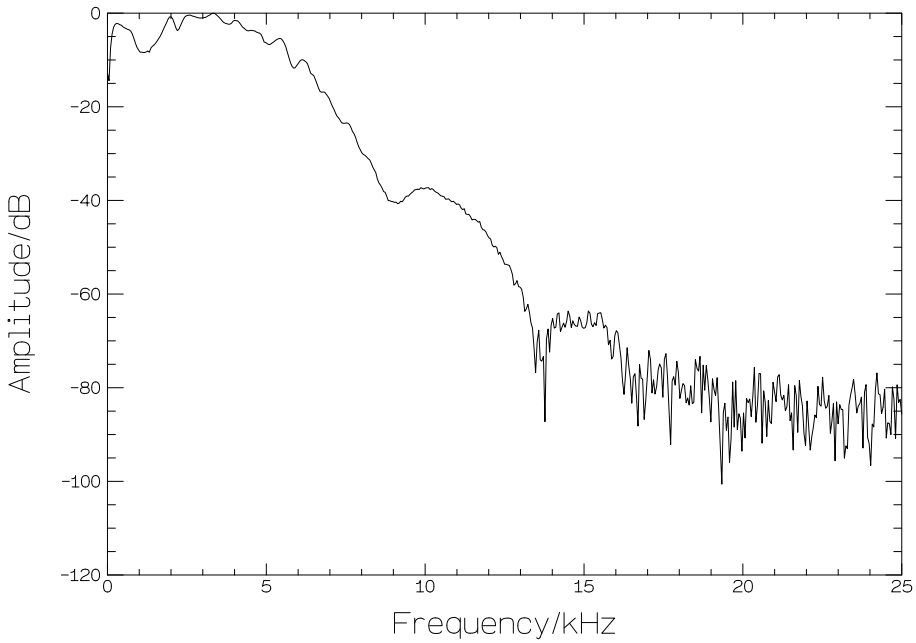

Figure 3.11: *Spectrum of input pulse.*

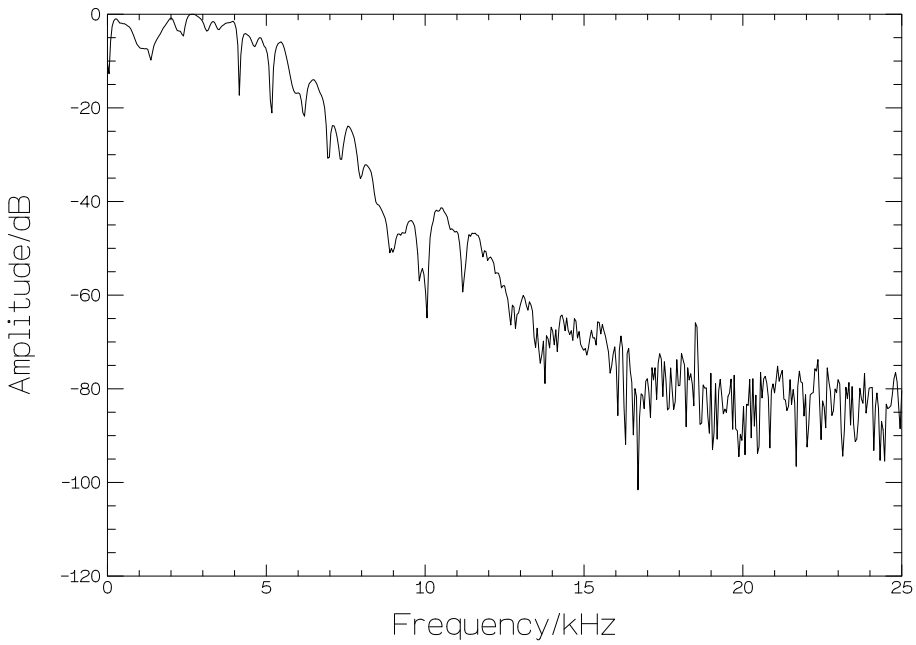

Figure 3.12: *Spectrum of reflections from stepped tube and coupler.*

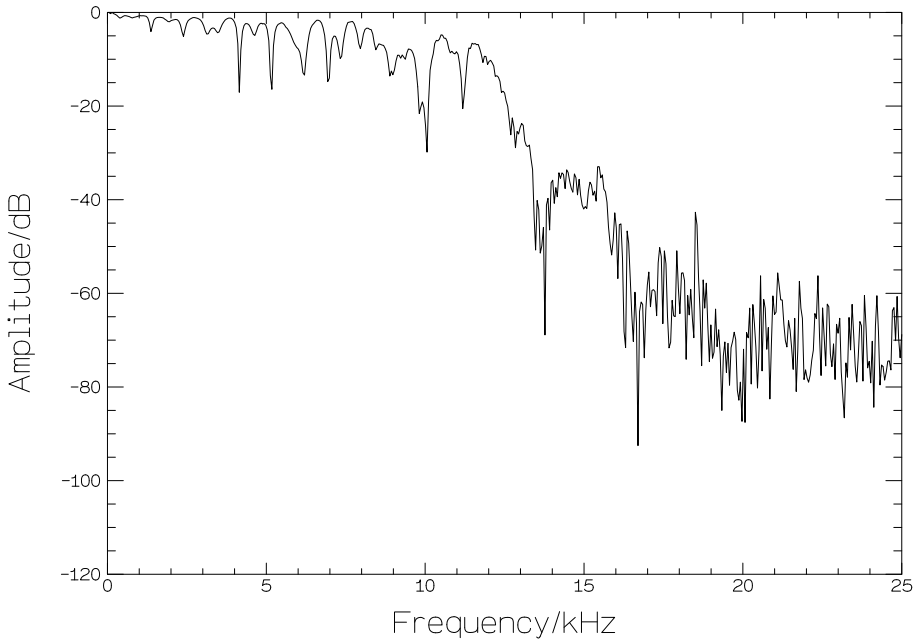

Figure 3.13: *Spectrum of input impulse response of stepped tube and coupler.*

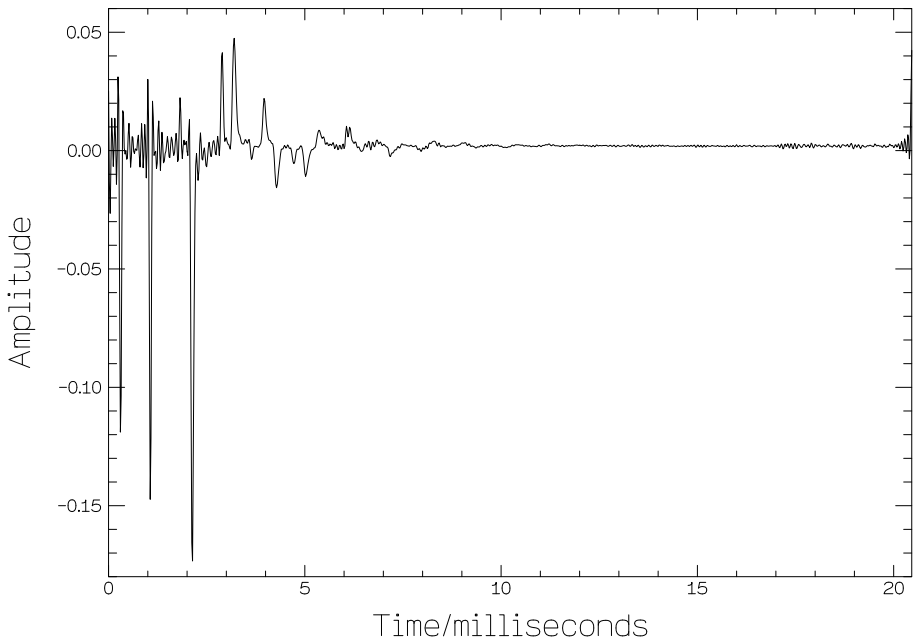

Figure 3.14: *Input impulse response of stepped tube and coupler.*

Applying equation 3.1 to these frequency domain values yields the spectrum of the input impulse response of the stepped tube and coupler (figure 3.13) and inverse FFT'ing yields the input impulse response (figure 3.14).

Again, the source of each reflection in the input impulse response can be identified. The reflection at approximately 0ms is from the small discontinuity at the joining of source tube and coupler, the reflection at approximately 0.3ms is from the start of the stepped tube, the reflection at approximately 1.1ms is from the step in the tube, and the reflection at approximately 2.2ms is from the end of the tube.

3.5.1 Anti-aliasing filter

The anti-aliasing filter is included to remove any frequencies above 20kHz from the signal, thus preventing them from being aliased upon sampling (aliasing gives rise to frequencies above the 25kHz Nyquist frequency being transposed down in to the 0-25kHz band). However, examination of figures 3.11 and 3.12 suggests that the input pulse and the stepped tube/coupler reflections only contain frequencies up to approximately 15kHz. Hence, the purpose of the anti-aliasing filter is to filter out any high frequency noise which may be introduced by external sources, and which would get aliased into the 0-25kHz band. With the possible exception of electrical noise, it is unlikely that there will be such noise in the anechoic chamber.

Spectra of the input pulse and the stepped tube/coupler reflections, along with the resultant input impulse response, measured without the filter in place (figures 3.15, 3.16 and 3.17) are virtually identical to the spectra and the input impulse response measured with the filter in place (figures 3.11, 3.12 and 3.14). Again, both measurements were made at a temperature of $\tau = 16.5^{o}$C. It is clear that under the conditions of the anechoic chamber the anti-aliasing filter is unnecessary and only serves to add extra noise to the signal. For this reason, most of the measurements in the present study were made without the anti-aliasing filter in position. However, under less controlled

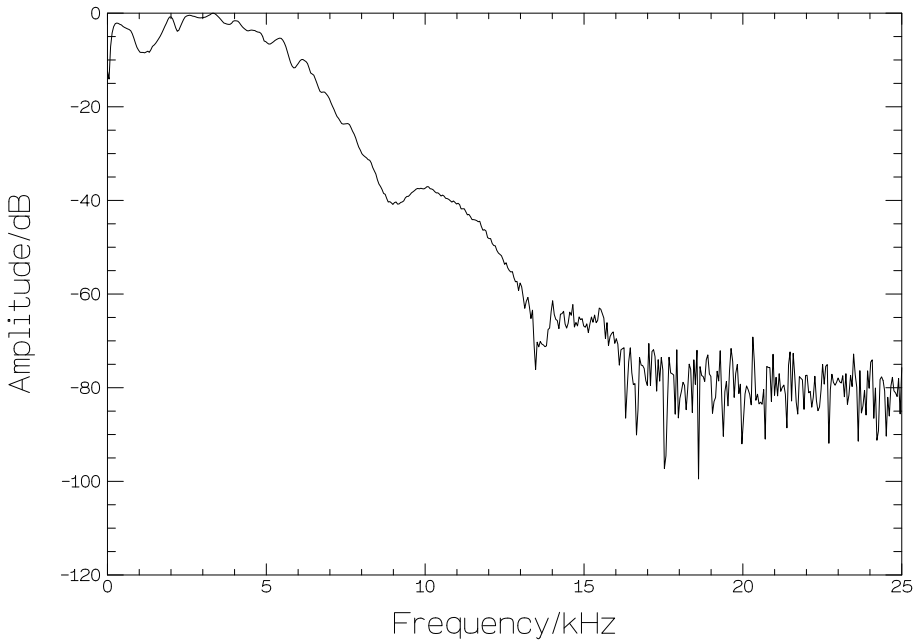

Figure 3.15: *Spectrum of input pulse (no anti-aliasing filter).*

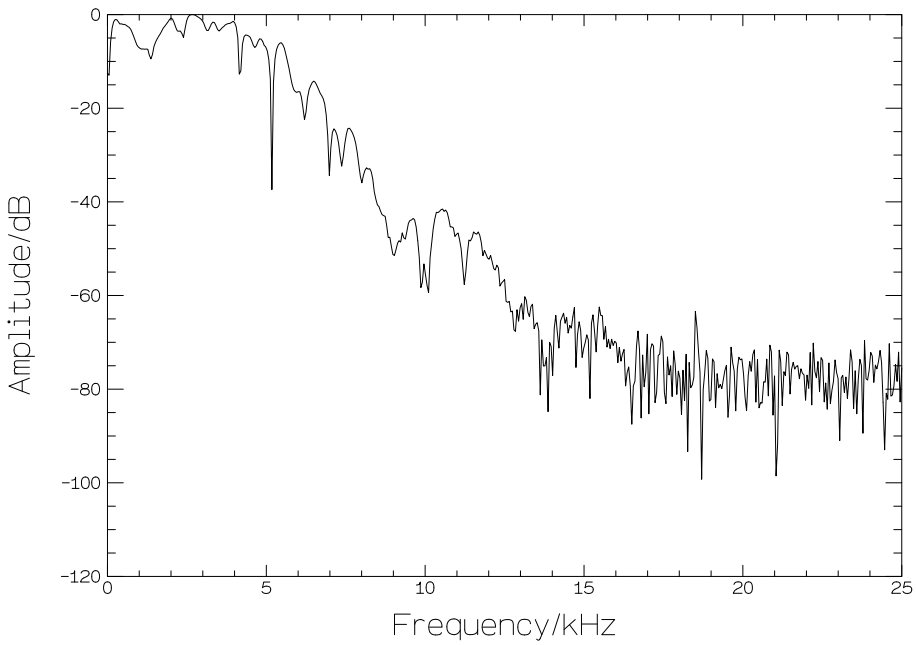

Figure 3.16: *Spectrum of reflections from stepped tube and coupler (no anti-aliasing filter).*

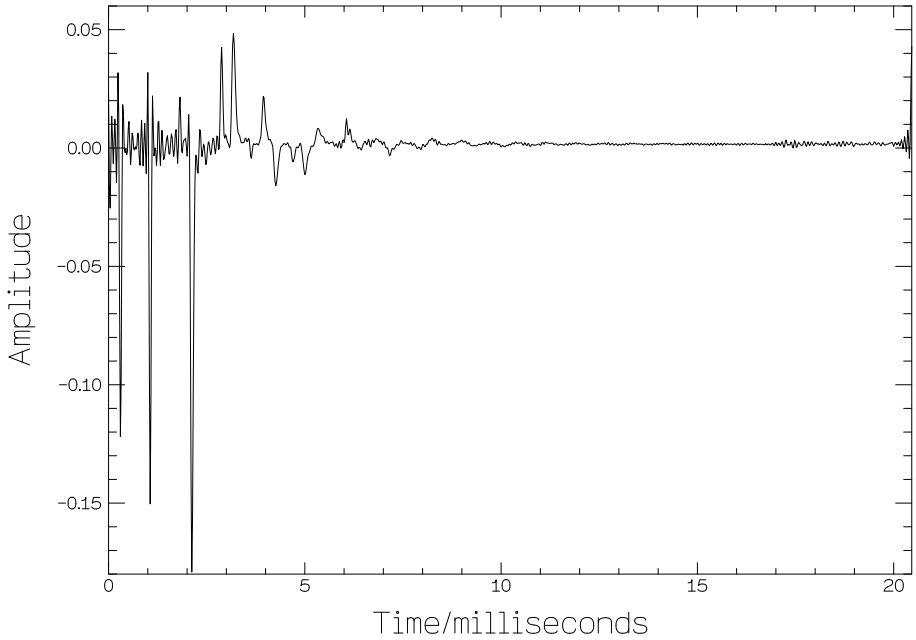

Figure 3.17: *Input impulse response of stepped tube and coupler (no anti-aliasing filter).*

conditions where external noise sources may be present, the anti-aliasing filter must be included.

3.5.2 Gibbs phenomenon

Close examination of figures 3.14 and 3.17 reveals that each reflection in the input impulse response is rippled. This ripple is an example of the Gibbs phenomenon and cannot be avoided. It is introduced because both the input pulse and object reflections have a finite bandwidth determined by the 50kHz sampling frequency. Both signals only contain frequencies up to approximately 15kHz (which is well below the Nyquist frequency of 25kHz) so this finite bandwidth only becomes a problem when the signals are FFT'ed and divided to give the spectrum of the input impulse response. The input impulse response should contain all frequencies but the experimentally determined value is bandwidth limited to 25kHz (figure 3.13). The lack of higher frequencies is the source of the ripple when the spectrum is inverse transformed to the time domain.

The experimentally determined input impulse response is only an approximation to the actual input impulse response.

Chapter 4

Bore reconstruction

4.1 Introduction

Direct application of one of the bore reconstruction algorithms of section 2.3 to an object's experimentally determined input impulse response should yield the bore profile of that object. In this chapter, it is shown that a DC component must first be removed from the input impulse response before a profile resembling that of the object can be produced. Two methods for evaluating this DC offset are discussed. The accuracies of the algorithms are then compared and reconstructed profiles of three brass instruments are presented.

4.2 DC offset problem

Figure 4.1 shows the bore reconstruction resulting from the direct application of the Ware-Aki algorithm to the input impulse response (figure 3.17) of the stepped tube/coupler system described in section 3.5. The input impulse response was determined from measurements made at $\tau = 16.5^{o}$C.

Similarly, figure 4.2 shows the bore reconstruction resulting from the direct application of the lossy layer-peeling algorithm (using the rotating phase model lossy filter)

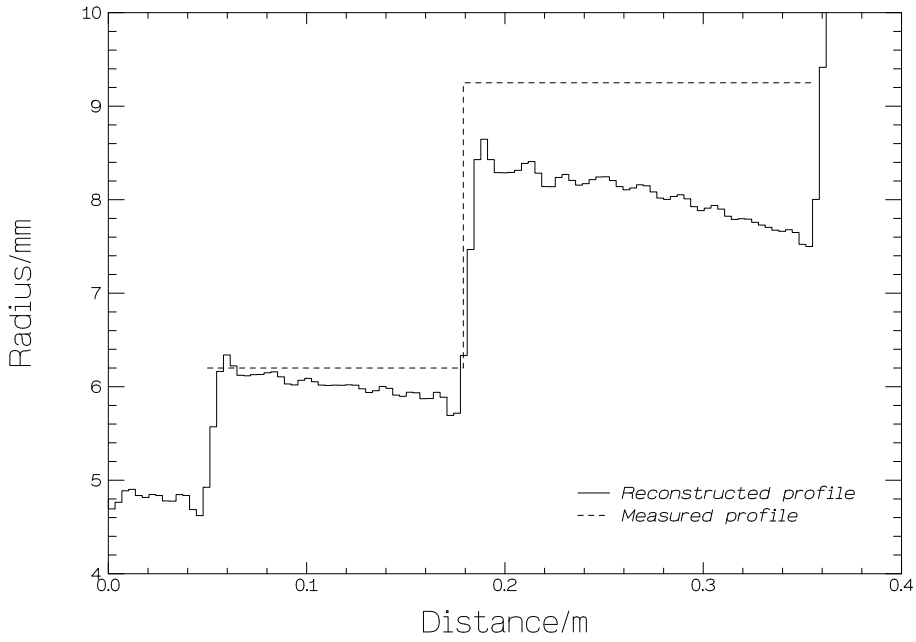

Figure 4.1: *Ware-Aki reconstruction of stepped tube. DC offset not removed from input impulse response.*

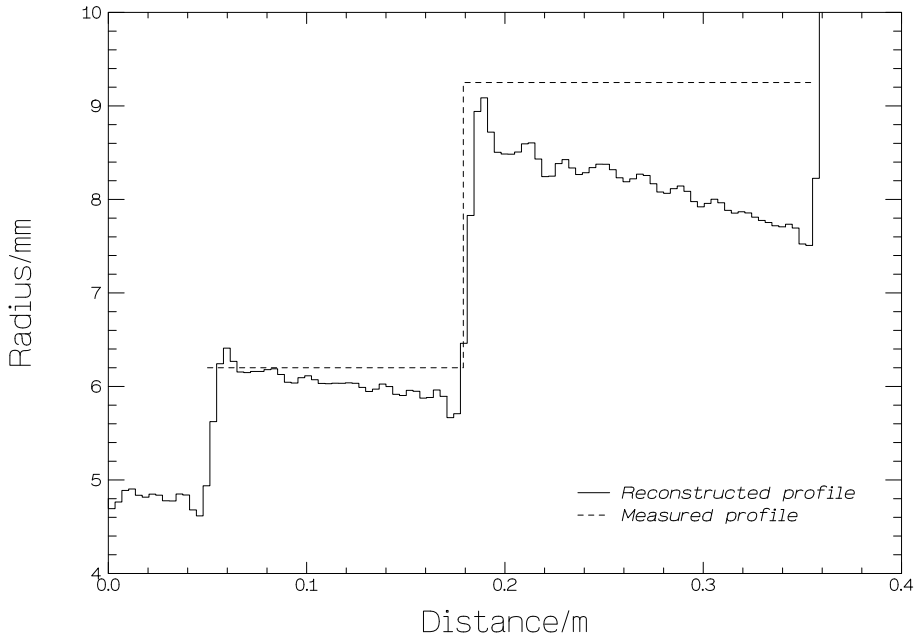

Figure 4.2: *Lossy layer-peeling reconstruction of stepped tube (using the rotating phase model lossy filter). DC offset not removed from input impulse response.*

to the same input impulse response.

In each case, the reconstructed profile bears little resemblance to the measured profile. The radius of each section of the stepped tube decreases with increasing distance along the tube, instead of remaining constant.

Close examination of figure 3.17 reveals that the poor reconstructions are not a fault of the algorithms but a consequence of the presence of a DC offset in the input impulse response of the stepped tube and coupler. The experimentally determined input impulse response of an object generally contains a DC offset. This offset, which is a result of DC offsets in the input pulse and object reflections, manifests itself by causing reconstructions to expand or contract too rapidly. For accurate reconstruction, the offset must be completely removed before application of a reconstruction algorithm.

4.2.1 Iterative method of DC offset removal

Originally, the DC offset value was obtained by an iterative process of estimation and adjustment, until a section of the object known to be cylindrical was reconstructed as cylindrical, or (only when using the more accurate lossy reconstruction algorithms) until the bore reconstruction coincided with a directly measured diameter, at a known point towards the end of the object. The method therefore required some prior knowledge of the object's profile. It also proved very time consuming, requiring a reconstruction to be calculated at each step of the iteration. However, reasonable reconstructions could be achieved.

Figure 4.3 shows the stepped tube bore reconstruction resulting from the application of the Ware-Aki algorithm to the input impulse response of figure 3.17, with the DC offset removed. The DC offset was obtained by iteratively applying the algorithm and adjusting the value removed from the response until the reconstruction had approximately parallel sides.

Figure 4.4 shows the stepped tube bore reconstruction resulting from the applica-

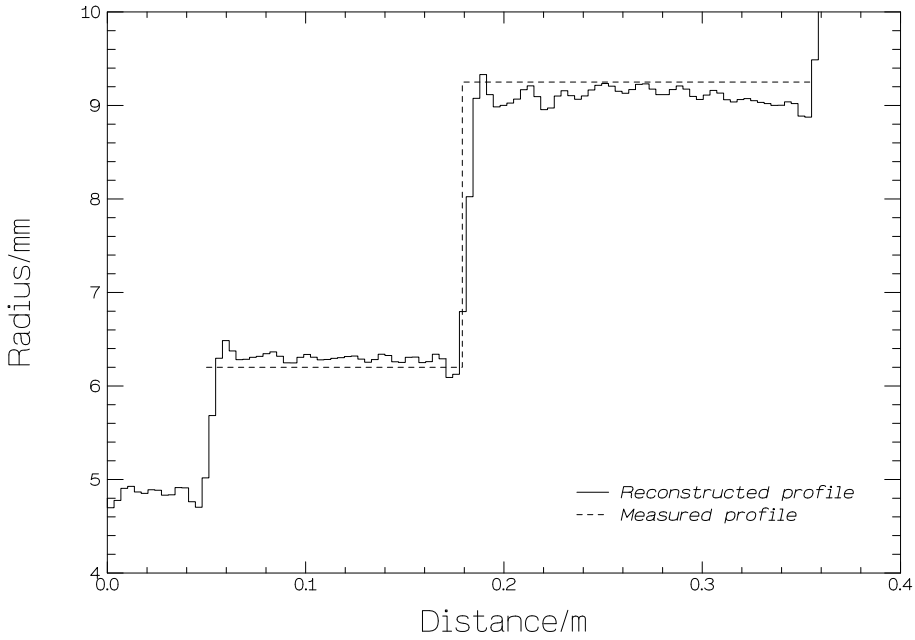

Figure 4.3: *Ware-Aki reconstruction of stepped tube. DC offset removed from input impulse response using iterative method.*

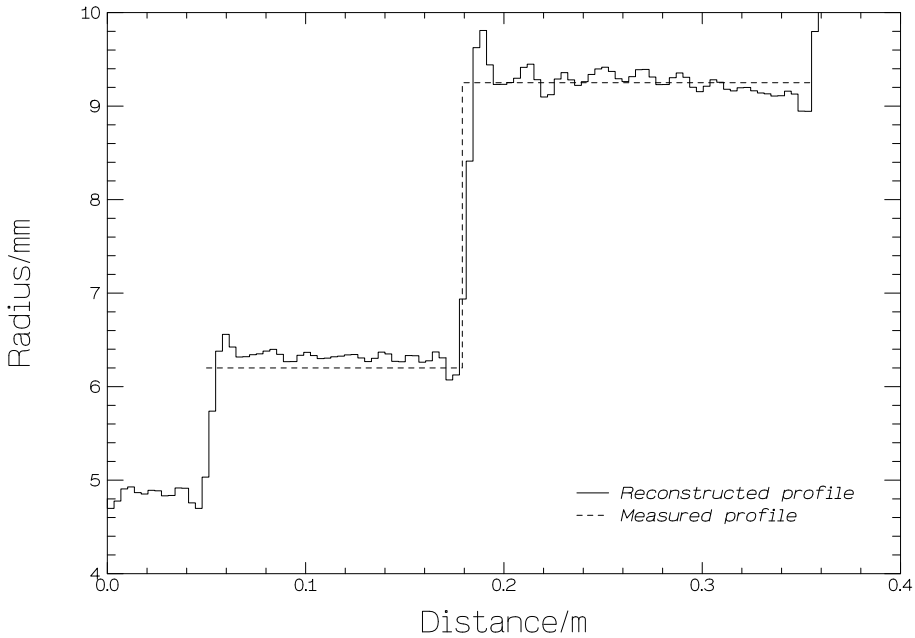

Figure 4.4: *Lossy layer-peeling reconstruction of stepped tube (using the rotating phase model filter). DC offset removed from input impulse response using iterative method.*

tion of the lossy layer-peeling algorithm (using the rotating phase model lossy filter) to the same input impulse response with, again, the DC offset removed. In this case, the DC offset was obtained by iteratively applying the algorithm and adjusting the value removed from the response until the reconstructed radius agreed with the directly measured radius of 9.25 ± 0.05mm, at 310mm along the tube.

In both figure 4.3 and figure 4.4, note the presence of the coupler at the start of the reconstruction. The overshoots at each expansion in the reconstruction are due to the ripple in the input impulse response (the cause of which was discussed in section 3.5.2).

4.2.2 Cylindrical connector method of DC offset removal

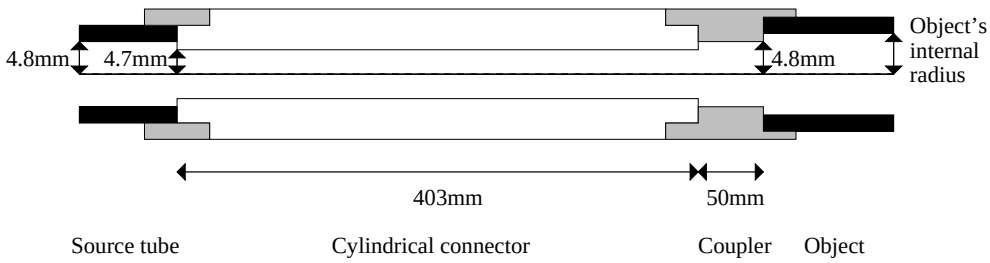

Figure 4.5: *Cylindrical connector inserted between source tube and coupler.*

An alternative method was devised which allowed the DC offset to be determined directly from the input impulse response rather than from examination of bore reconstructions. This method of finding the DC offset value involves the insertion of a 403mm long steel cylindrical tube (of wall thickness 1.7mm and internal radius 4.7mm, approximately the same as that of the source tube) between the source tube and the coupling to the object under investigation (figure 4.5). Since there should be no signal reflected back from this cylindrical connector, the input impulse response should be zero. The average value of the measured input impulse response (of the connector coupled to the test object) over this range thus gives the DC offset value. The method proves much quicker than the iterative method; the DC offset can be found

without a reconstruction having to be calculated. Another advantage of this method is that it does not require any prior knowledge of the profile of the object being measured.

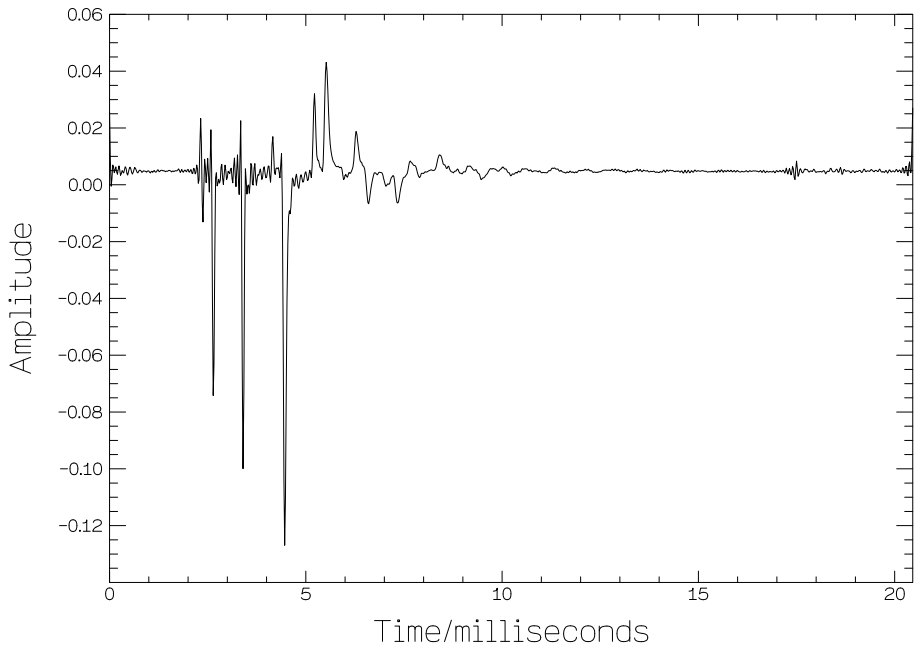

Figure 4.6: *Input impulse response of stepped tube and coupler with cylindrical connector.*

Figure 4.6 shows the input impulse response of the stepped tube and coupler with the cylindrical connector in place. The DC offset is evaluated over the 1ms-2ms range and is removed prior to the application of a reconstruction algorithm. Again, the input impulse response was determined from measurements made at $\tau = 16.5^{o}$C.

More accurate reconstructions can be obtained by removing the small reflections present in the first millisecond of the input impulse response (caused by the slight discontinuity in area between source tube and cylindrical connector). This is done by zeroing the first millisecond of the input impulse response (after the DC offset has been removed). When applying a reconstruction algorithm to this modified input impulse response, the connector area should be used instead of the source tube area as the initial area S_0.

Figure 4.7 shows the input impulse response of the stepped tube and coupler with

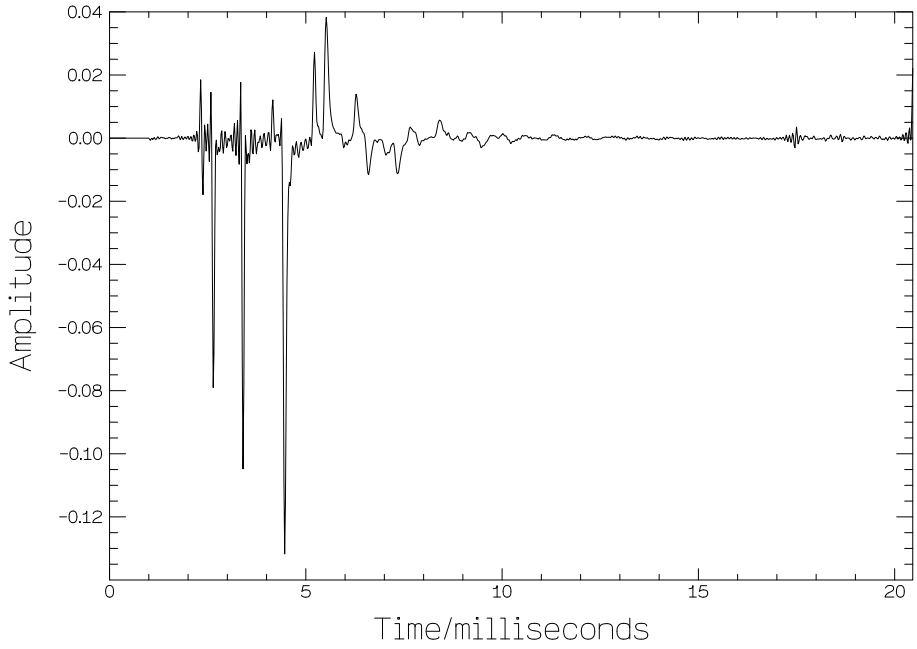

Figure 4.7: *Input impulse response of stepped tube and coupler with cylindrical connector. DC offset removed and first millisecond zeroed.*

cylindrical connector after the DC offset has been removed and the first millisecond zeroed.

Figure 4.8 shows the stepped tube bore reconstruction resulting from the application of the Ware-Aki algorithm to the modified input impulse response of figure 4.7.

Figure 4.9 shows the stepped tube bore reconstruction resulting from the application of the lossy layer-peeling algorithm (using the rotating phase model lossy filter) to the same modified input impulse response.

In both figure 4.8 and figure 4.9, note the presence of the cylindrical connector and the coupler at the start of the reconstruction. Again, the overshoots at each expansion in the reconstruction are due to the ripple in the input impulse response.

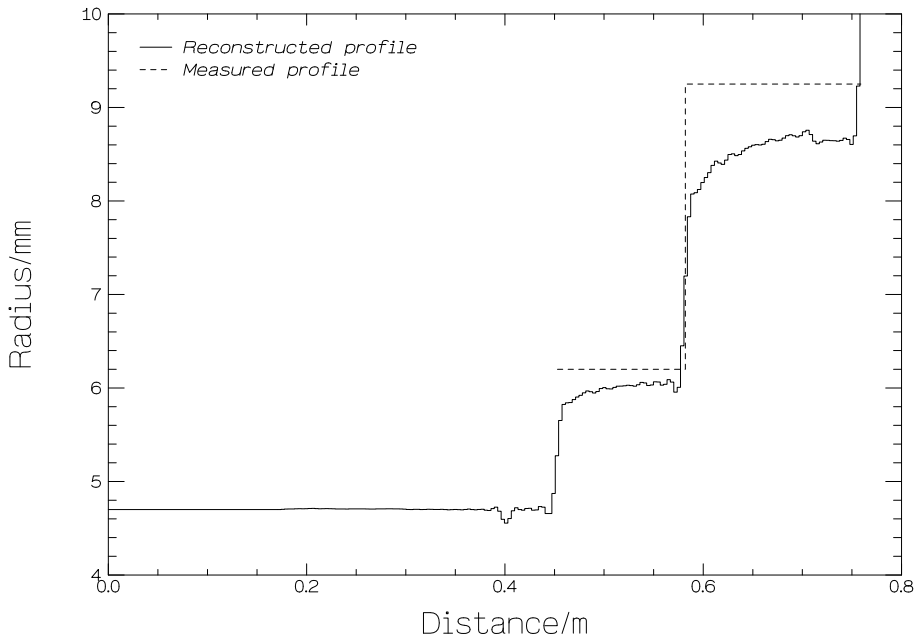

Figure 4.8: *Ware-Aki reconstruction of stepped tube. DC offset removed from input impulse response using cylindrical connector method.*

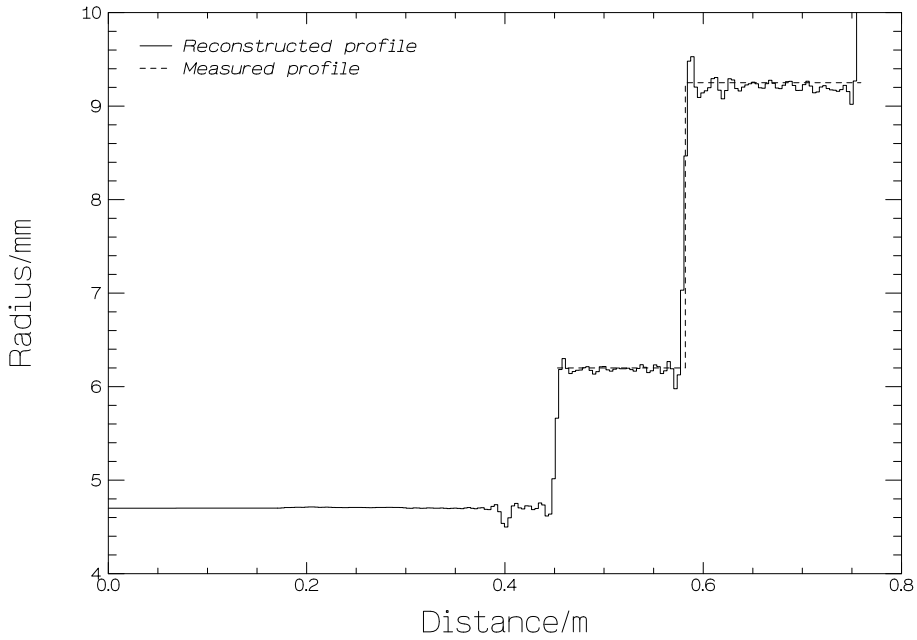

Figure 4.9: *Lossy layer-peeling reconstruction of stepped tube (using the rotating phase model lossy filter). DC offset removed from input impulse response using cylindrical connector method.*

4.3 Comparison of reconstruction algorithms

In the previous section, reconstructions made using the Ware-Aki algorithm and the lossy layer-peeling algorithm (using the rotating phase model lossy filter) were presented.

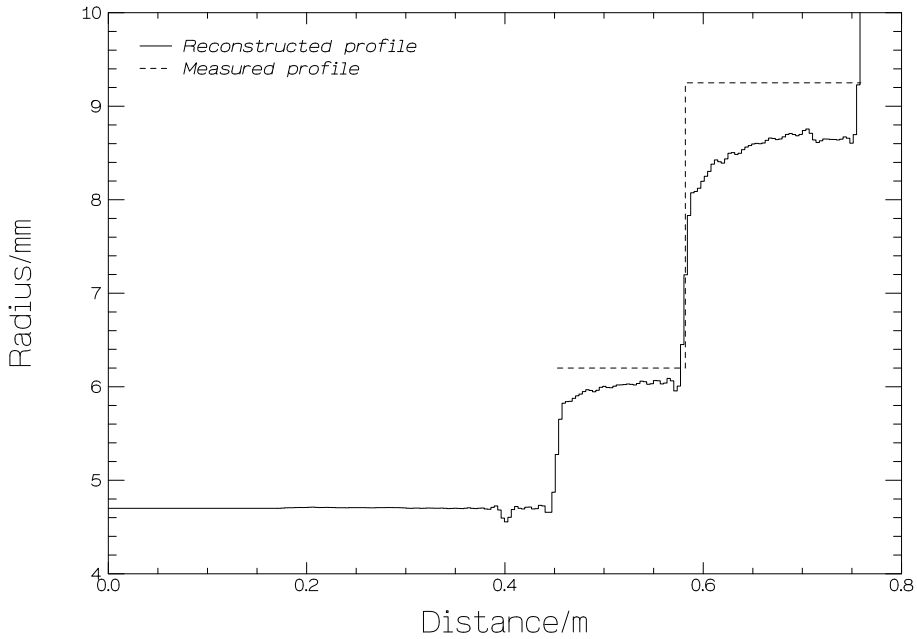

Figure 4.10: *Lossless layer-peeling reconstruction of stepped tube. DC offset removed from input impulse response using cylindrical connector method.*

Figure 4.10 shows the stepped tube bore reconstruction resulting from the application of the lossless layer-peeling algorithm to the modified input impulse response of figure 4.7.

Figure 4.11 shows the stepped tube bore reconstruction resulting from the application of the lossy layer-peeling algorithm (using the all-pole model lossy filter) to the same modified input impulse response.

It is now possible to compare reconstructions made using the four algorithms described in chapter 2 (the Ware-Aki algorithm, the lossless layer-peeling algorithm, the lossy layer-peeling algorithm using the rotating phase model lossy filter, and the lossy layer-peeling algorithm using the all-pole model lossy filter). In this way, the most

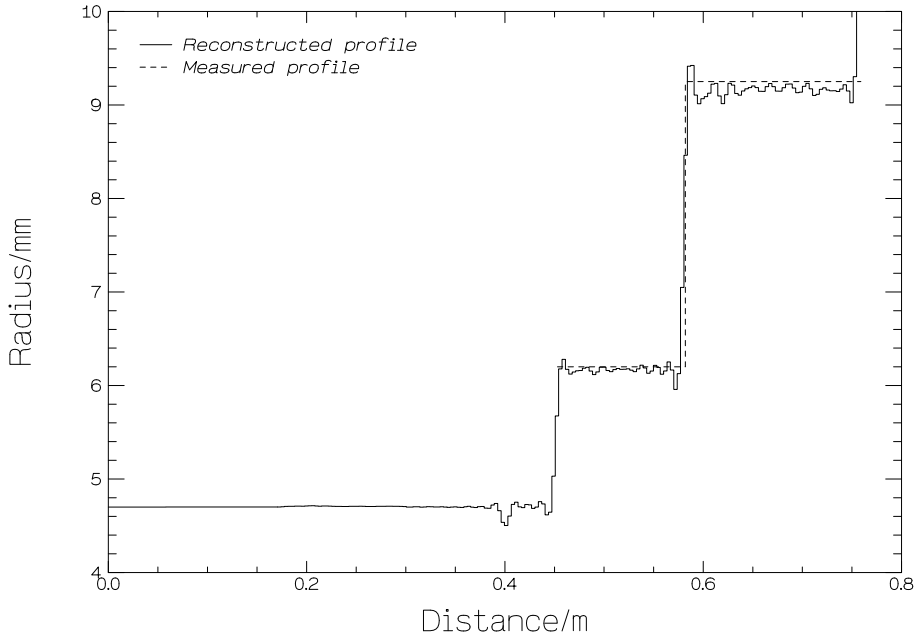

Figure 4.11: *Lossy layer-peeling reconstruction of stepped tube (using all-pole model lossy filter). DC offset removed from input impulse response using cylindrical connector method.*

suitable reconstruction algorithm can be found.

The axial resolution of all the bore reconstructions is the same. The two input impulse responses to which the reconstruction algorithms were applied, were both determined from measurements made at $\tau = 16.5^{o}$C. Hence, the length of the cylindrical segments making up the reconstructions is $l = cT/2 = (331.6\sqrt{1+16.5/273}) \times (2 \times 10^{-5})/2$=0.0034m=3.4mm. However, the error on the reconstructed axial position is not $\pm$3.4mm, it is simply -3.4mm. To understand why this is the case, consider a pulse reflected back to the microphone from the open end of a cylindrical pipe. The microphone output is sampled. If the passage of the start of the pulse coincides with a sample point (figure 4.12), the pulse will be recorded at the correct time. However, if the passage of the start of the pulse occurs between two sample points (figure 4.13), the pulse will not be recorded until the second of the sample points, some time $< T$ later. If a reconstruction of the cylindrical pipe was made, its length would be anything up

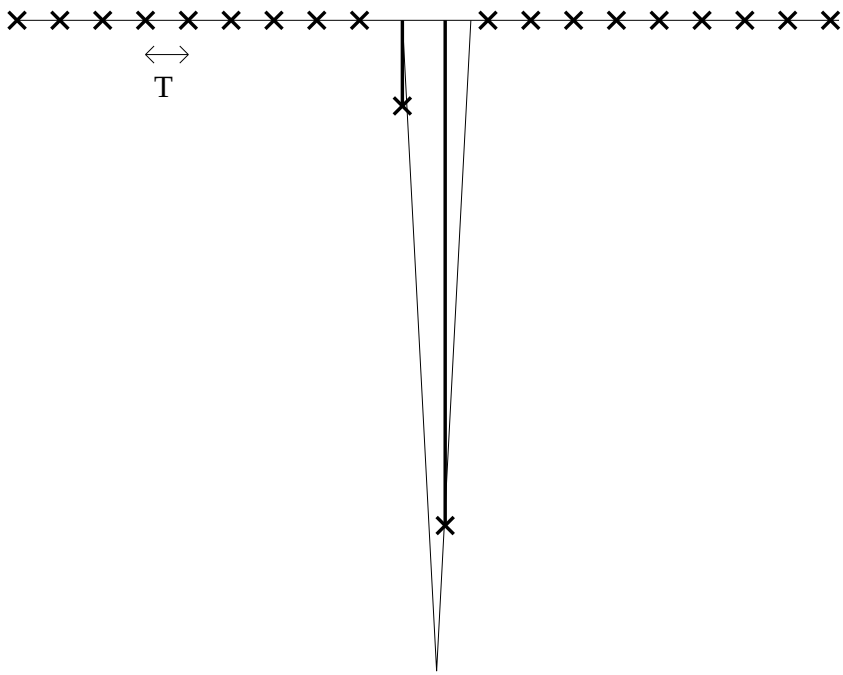

Figure 4.12: *Start of pulse coincides with a sample point.*

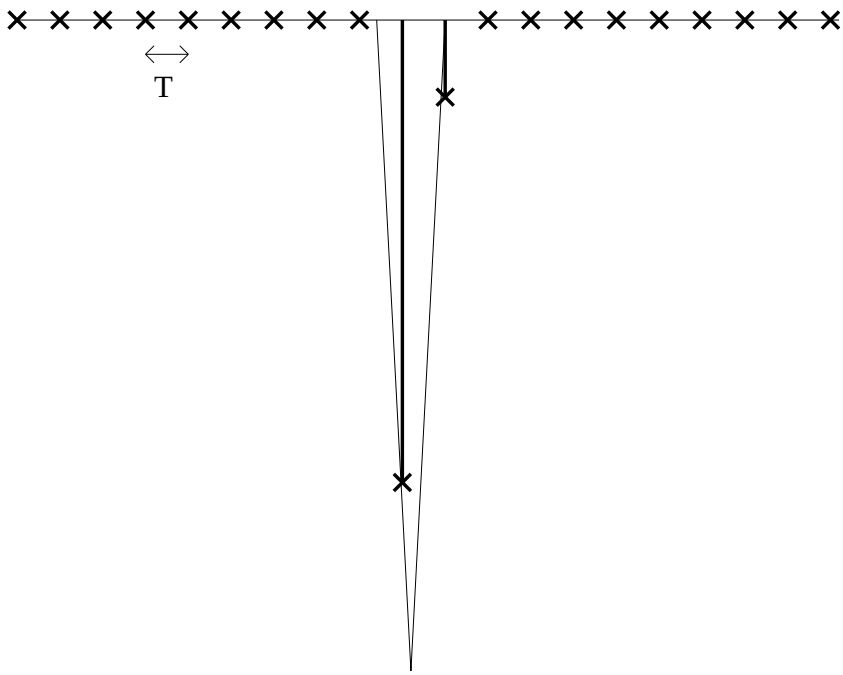

Figure 4.13: *Start of pulse lies between two sample points.*

to $cT/2$ too long; i.e. the error on the reconstructed length would be $-cT/2$. Thus, for measurements made at $\tau = 16.5^{o}$C, the error on the reconstructed length is -3.4mm.

Examination of figures 4.8 and 4.10 reveals that, as one would expect, the Ware-Aki algorithm and the lossless layer-peeling algorithm (neither of which compensate for losses) give identical bore reconstructions. These reconstructions are in poor agreement with the measured profile of the stepped tube. The first section of the reconstructed stepped tube increases in radius from 5.8mm to 6.0mm (rather than remaining constant at the measured radius of 6.20 ± 0.05mm), whilst the second section of the reconstructed stepped tube increases in radius from 8.1mm to 8.7mm (rather than remaining constant at the measured radius of 9.25 ± 0.05mm). This is because the losses experienced by the signal whilst propagating through the cylindrical connector, the coupler, the stepped tube and back again, are large. These losses also cause the expansions in the reconstructed profile to be sloped rather than sharp. The axial position of each expansion can be estimated as the axial position of the midpoint of the slope. Using this method, the axial positions of the start of the tube, the step and the end of the tube can be estimated as $454-3.4$mm, $584-3.4$mm and $762-3.4$mm respectively. These values agree with the measured positions of 453 ± 0.5mm, 582 ± 0.5mm and 759 ± 0.5mm to within the error bounds.

Examination of figure 4.3 shows that when the signal only has to propagate through the coupler, the stepped tube and back again, the losses experienced are smaller and the reconstructed profile is in better agreement with the measured profile. The first section of the reconstructed stepped tube has a radius which remains approximately constant at 6.3mm (which is actually slightly larger than the measured radius of 6.20 ± 0.05mm), whilst the second section of the reconstructed stepped tube has a radius which remains approximately constant at 9.1mm (compared with the measured radius of 9.25 ± 0.05mm). Although the reconstructed profile is in better agreement, the radii still do not fall within the error bounds of the measured radii. The smaller losses cause the recon-

structed profile to have sharper expansions. The start of the tube, the step and the end of the tube can be estimated as $51-3.4$mm, $181-3.4$mm and $359-3.4$mm respectively. These values agree with the measured positions of 50 ± 0.5mm, 179 ± 0.5mm and 356 ± 0.5mm to within the error bounds.

Inclusion of the cylindrical connector is necessary to enable measurement of the DC offset, so the signal experiences significant losses in most measurement cases. In order to obtain good reconstructions of lengthy systems, an algorithm must be used which compensates for the losses experienced by the signal whilst propagating through the object; i.e. the lossy layer-peeling algorithm using either the rotating phase method or the all-pole method to model the filter representing losses.

Examination of figures 4.9 and 4.11 shows that the reconstructed profiles (made using the two different lossy layer-peeling algorithms) are in good agreement with the measured profile and with each other. The radii of both reconstructed profiles agree with the measured radii of 6.20 ± 0.05mm and 9.25 ± 0.05mm to within the error bounds. The reconstructed profiles both have sharp expansions and, in both cases, the start of the tube, the step and the end of the tube can be estimated as $454-3.4$mm, $584-3.4$mm and $762-3.4$mm respectively. These values agree with the measured positions of 453 ± 0.5mm, 582 ± 0.5mm and 759 ± 0.5mm to within the error bounds.

At a sampling frequency of 50kHz, the delay introduced when using the rotating phase model lossy filter is insignificant and both lossy layer-peeling algorithms give similar results. However, the lossy layer-peeling algorithm takes approximately ten times as long to calculate a profile when the all-pole model lossy filter is used as it does when the rotating phase model lossy filter is used. Therefore, at a sampling frequency of 50kHz, the lossy layer-peeling algorithm using the rotating phase model lossy filter proves the most suitable reconstruction algorithm to use.

4.4 Reproducibility of reconstructions

In the previous section, radii at different points along a typical stepped tube reconstruction (made using the lossy layer-peeling algorithm with the rotating phase model lossy filter) were found to agree with the corresponding directly measured radii to within the direct measurement reading error of 0.05mm.

To find out how reproducible the stepped tube reconstruction is (i.e. how much the reconstructed radii vary between one reconstruction and the next), the input impulse response of the tube was measured five times. The lossy layer-peeling algorithm was applied to each of the responses (after the DC offset had been calculated and removed using the cylindrical connector method), resulting in five bore reconstructions. Halfway along the first and second cylindrical sections, the radii of the five reconstructions were found. Mean values for the radius of the stepped tube at these two positions were calculated, along with the standard deviations.

Halfway along the first cylindrical section of the stepped tube, the radii of the five reconstructions were 6.173mm, 6.176mm, 6.189mm, 6.187mm and 6.165mm. The mean value for the reconstructed radius at this position is therefore 6.178mm with a standard deviation of 0.01mm. Halfway along the second cylindrical section of the stepped tube, the radii of the five reconstructions were 9.213mm, 9.205mm, 9.219mm, 9.238mm and 9.191mm. The mean value for the reconstructed radius at this position is therefore 9.213mm with a standard deviation of 0.017mm.

The standard deviation provides a measure of the reproducibility of the reconstruction. It is clear that the reproducibility worsens with axial distance along the reconstructed profile. However, even towards the end of the stepped tube, the standard deviation of 0.017mm is less than the reading error of 0.05mm on the directly measured radius.

4.5 Brass instrument reconstructions

The combined length of the cylindrical connector, coupler and stepped tube was 759mm. All the reflections returning from this system completely passed the microphone before the first source reflection arrived at the microphone (i.e. they completely passed the microphone within the time taken to travel $2l_1$=6.20m, the distance from the microphone to the loudspeaker and back).

When measuring objects which are longer than the 306mm stepped tube, a longer source tube section l_1 is necessary to ensure that the reflections returning from the cylindrical connector, coupler and object are completely recorded before the first source reflection arrives.

In this section, reconstructions of three brass instruments (whose lengths, when combined with the cylindrical connector and one of the couplers shown in figure 3.6, are all greater than 1.6m), are presented. These instruments were measured on a second reflectometer, identical to the reflectometer described in section 3.2 except for the source tube section l_1, which at 7.37m is over twice as long. The instrument reflections were sampled using the same 50kHz sampling frequency but with a sample length of 2048 points, giving a sample time of 40.96ms.

4.5.1 Amati-Kraslice trumpet

Figure 4.14 shows a bore reconstruction of an Amati-Kraslice trumpet in B♭ with no valves depressed (calculated from measurements made at τ=17.5^{o}C). This instrument is a 'standard model' trumpet. Direct measurements are superimposed on the graph to provide a comparison with the reconstructed profile.

Although the DC offset was calculated and removed using the cylindrical connector method, the connector is not displayed on the graph. For the first 100mm, the graph shows the coupler which was chosen (from the six couplers displayed in figure 3.6) to give a good fit to the trumpet. The coupler penetrated a distance of 13.5mm into the

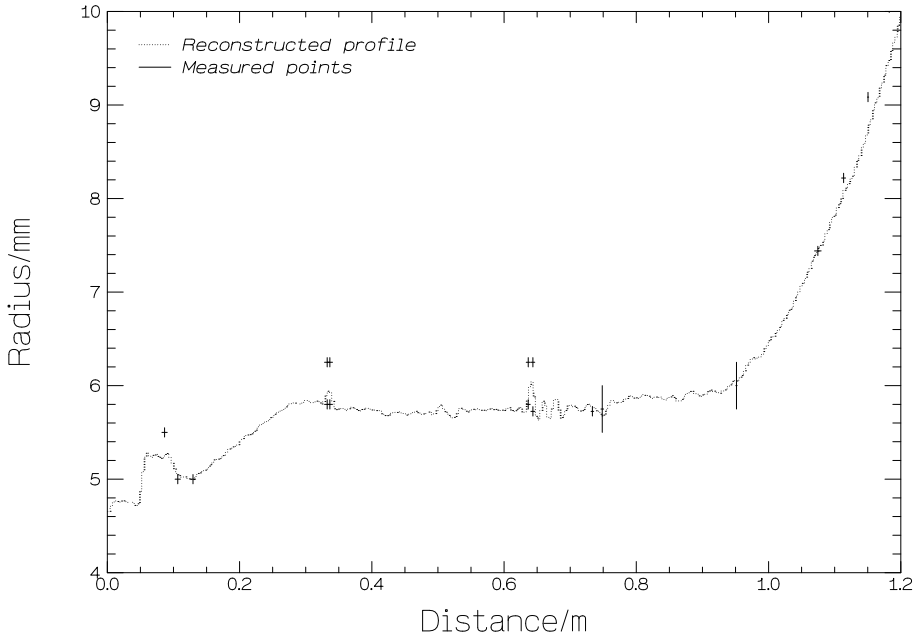

Figure 4.14: *Reconstructed and measured bore profiles of Amati-Kraslice trumpet. No valves depressed.*

instrument. From 100mm to 1200mm, the graph shows the trumpet profile (starting from a position 13.5mm in from the mouthpiece end).

After an initial widening of the bore (between 130mm and 300mm on the graph), the profile of the trumpet is approximately cylindrical. The radius remains fairly constant through the valve section (between 650mm and 750mm), which is situated towards the end of the instrument. This constant bore radius is characteristic of a 'standard model' trumpet. At the bell (which begins at approximately 930mm), the radius begins to increase rapidly.

Comparison of the reconstructed profile with radii directly measured at various accessible positions along the trumpet bore reveals that the reconstruction breaks down in regions where the bore radius changes rapidly. Prime examples of such regions are the steps at the beginning and end of the main tuning-slide (at graph positions 340mm and 630mm respectively) and the flaring bell (from approximately 930mm onwards). In these regions, the reconstructed profile has a smaller radius than the directly measured

radius. The most likely explanation for this breakdown is the fact that the reconstruction algorithm assumes plane wave propagation within the instrument. At regions of rapid change in the bore profile, spherical wave propagation becomes predominant.

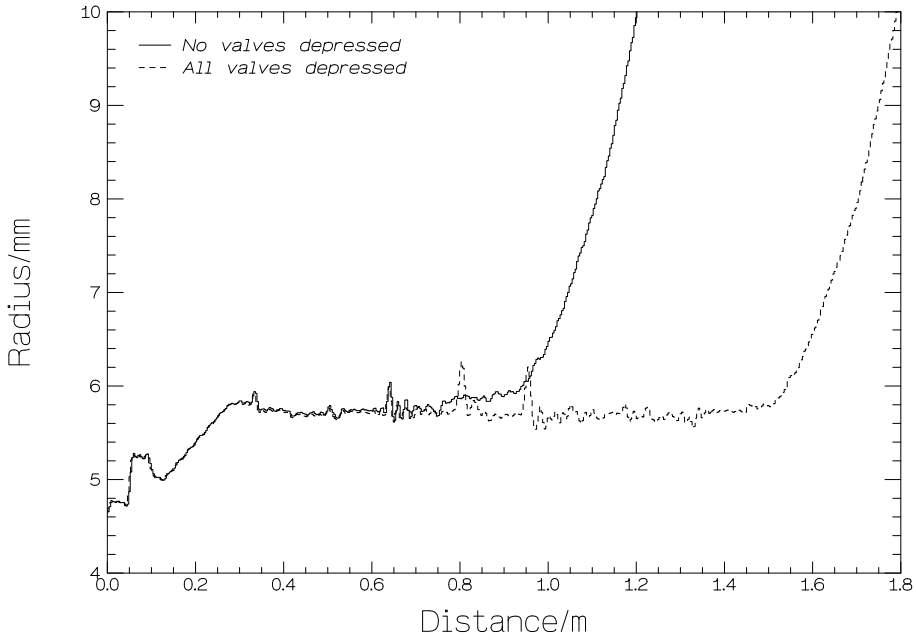

Figure 4.15: *Reconstructed bore profiles of Amati-Kraslice trumpet. No valves depressed and all three valves depressed.*

Figure 4.15 shows bore reconstructions of the Amati-Kraslice trumpet with no valves depressed and with all three valves depressed (calculated from measurements made at τ=17.5^{o}C and τ=18.2^{o}C respectively). Depressing all the valves introduces three extra sections of cylindrical tubing (the first, second and third valve tuning-slides), thus increasing the length of the valve section (which now occupies the region between 650mm and 1350mm on the graph). The total length of the instrument is increased by approximately 600mm.

Figure 4.16 shows a 3D representation of the Amati-Kraslice trumpet with no valves depressed, created from the reconstruction data displayed in figure 4.14.

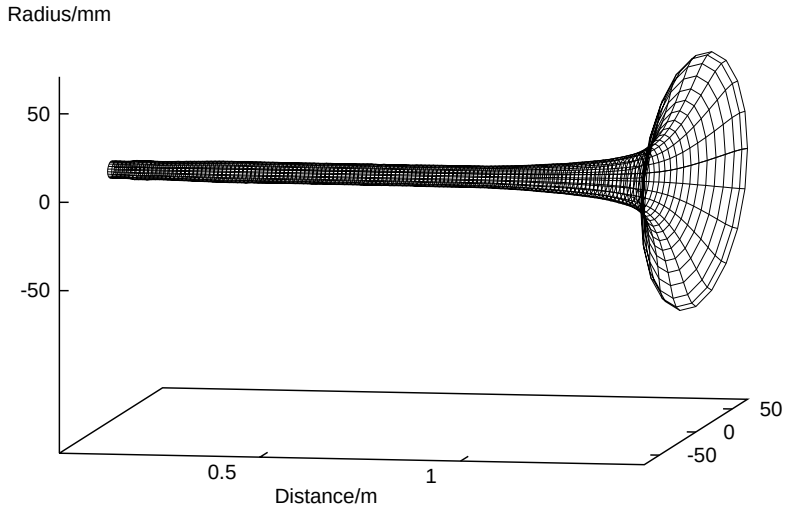

Figure 4.16: *3D representation of Amati-Kraslice trumpet. No valves depressed.*

4.5.2 Rudall Carte 'Webster' trumpet

Figure 4.17 shows a bore reconstruction of a Rudall Carte 'Webster' trumpet in B♭ with no valves depressed (calculated from measurements made at τ=17.7^{o}C). This instrument is a 'conical bore model' trumpet. Direct measurements are superimposed on the graph to provide a comparison with the reconstructed profile.

Again, although the DC offset was calculated and removed using the cylindrical connector method, the connector is not displayed on the graph. For the first 100mm, the graph shows the coupler which was chosen (from the six couplers displayed in figure 3.6) to give a good fit to the trumpet. The coupler penetrated a distance of 10.0mm into the instrument. From 100mm to 1200mm, the graph shows the trumpet profile (starting from a position 10.0mm in from the mouthpiece end).

After an initial widening of the bore (between 150mm and 310mm on the graph), the profile of the trumpet continues to expand slowly. The radius increases through the valve section (between 310mm and 410mm) which, unlike the Amati-Kraslice trumpet, is situated towards the start of the instrument. This continual slow expansion of the bore is characteristic of a 'conical bore' trumpet. At the bell (which begins at

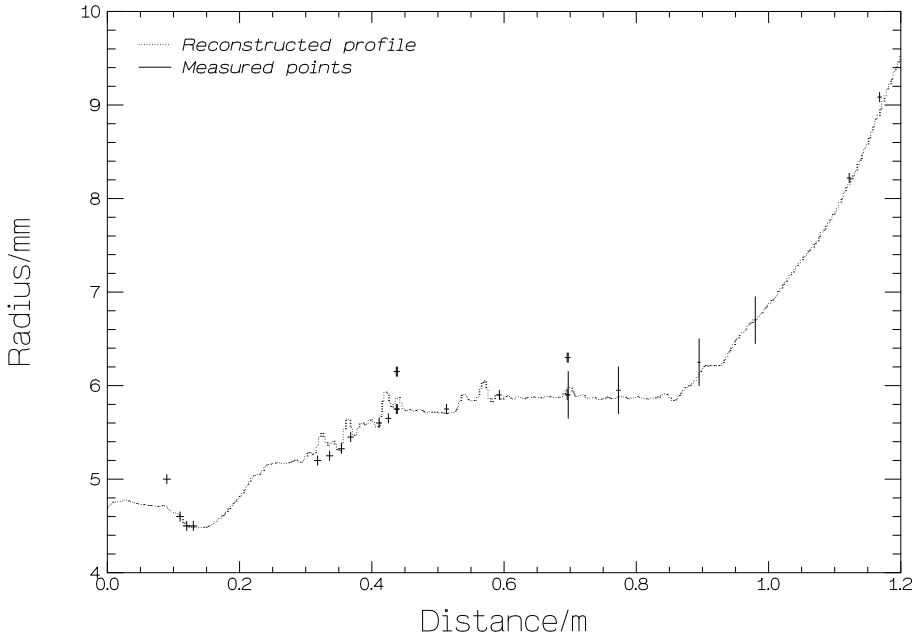

Figure 4.17: *Reconstructed and measured bore profiles of Rudall Carte 'Webster' trumpet. No valves depressed.*

approximately 860mm), the radius increases more rapidly.

Again, comparison of the reconstructed profile with radii directly measured at various accessible positions along the trumpet bore reveals that the reconstruction breaks down in regions where the bore radius changes rapidly (such as the steps at the ends of tuning-slides and the flaring bell).

Figure 4.18 shows bore reconstructions of the Rudall Carte 'Webster' trumpet with no valves depressed and with all three valves depressed (calculated from measurements made at τ=17.7^{o}C and τ=17.3^{o}C respectively). Depressing all the valves introduces three extra sections of cylindrical tubing (the first, second and third valve tuning-slides), thus increasing the length of the valve section (which now occupies the region between 310mm and 930mm on the graph). Unlike the Amati-Kraslice trumpet, the extra tubing is added towards the start of the trumpet rather than towards the end. The total length of the instrument is increased by approximately 520mm.

Figure 4.19 shows a 3D representation of the Rudall Carte 'Webster' trumpet with

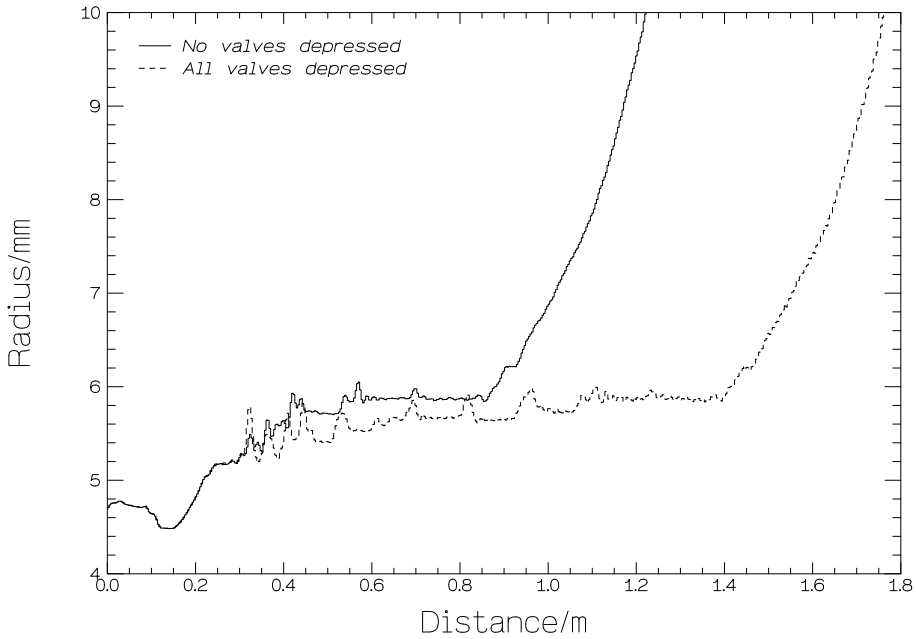

Figure 4.18: *Reconstructed bore profiles of Rudall Carte 'Webster' trumpet. No valves depressed and all three valves depressed.*

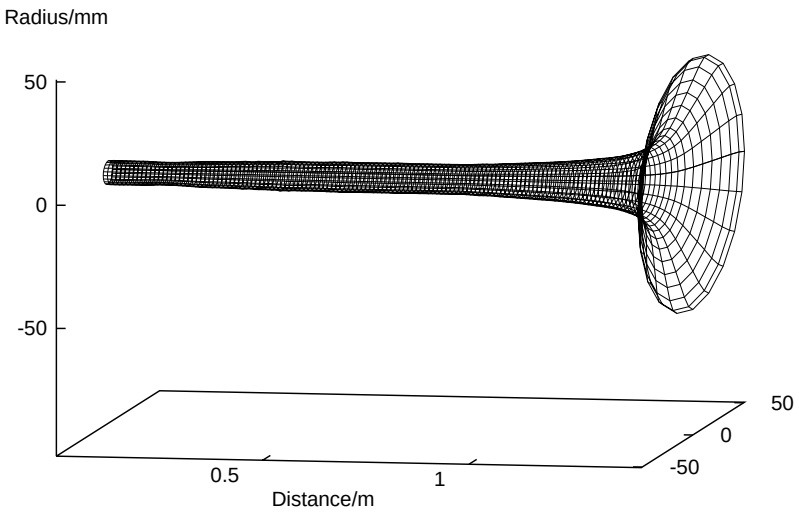

Figure 4.19: *3D representation of Rudall Carte 'Webster' trumpet. No valves depressed.*

no valves depressed, created from the reconstruction data displayed in figure 4.17.

4.5.3 Morhange Posthorn

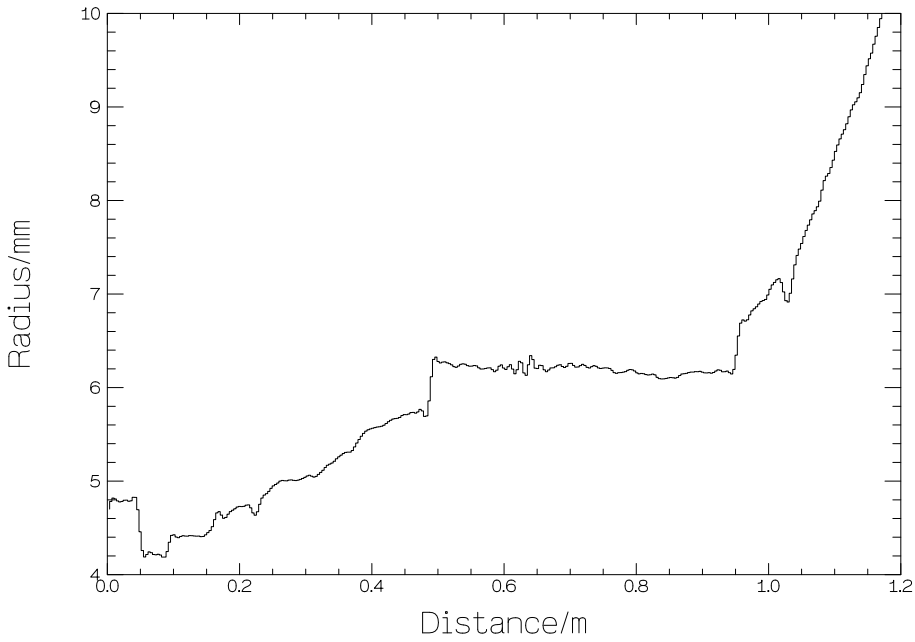

Figure 4.20: *Reconstructed bore profile of a Morhange posthorn.*

Figure 4.20 shows a bore reconstruction of a B♭ posthorn probably by Morhange (calculated from measurements made at τ=16.5^{o}C). For instruments such as these, it is not possible to make direct measurements as there are no tuning-slides and therefore no access points. Acoustic pulse reflectometry proves very useful in the measurement of such instruments.

Again, although the DC offset was calculated and removed using the cylindrical connector method, the connector is not displayed on the graph. For the first 100mm, the graph shows the coupler which was chosen (from the six couplers displayed in figure 3.6) to give a good fit to the posthorn. The coupler penetrated a distance of 21.9mm into the instrument. From 100mm to 1200mm, the graph shows the posthorn profile (starting from a position 21.9mm in from the mouthpiece end).

After an initial widening of the bore (between 170mm and 480mm on the graph),

the profile of the posthorn is approximately cylindrical. At the bell (which begins at approximately 950mm), the radius begins to increase rapidly. The sudden decrease in radius at 1030mm is caused by a dent in the bell of the instrument.

Figure 4.21 shows a 3D representation of the posthorn, created from the reconstruction data displayed in figure 4.20.

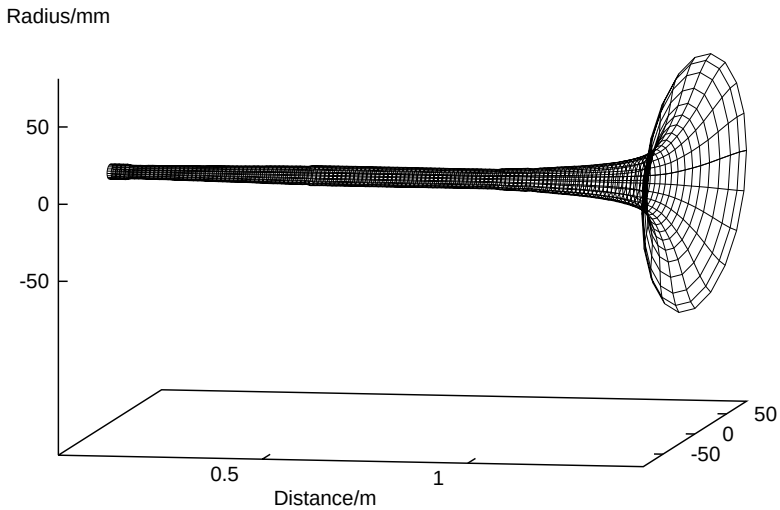

Figure 4.21: *3D representation of Morhange posthorn.*

Chapter 5

Input impedance

5.1 Introduction

In this chapter, impedance curves calculated from input impulse response measurements made using acoustic pulse reflectometry, are presented. An alternative, frequency domain method (the swept sine wave method) of measuring the input impedance of an object is described and sample results are shown. Impedance curves measured using the time domain technique of acoustic pulse reflectometry and using the frequency domain swept sine wave technique are compared both with each other and with a theoretical curve. Finally, impedance curves of two brass instruments are presented and discussed.

5.2 Acoustic pulse reflectometry method of measuring input impedance

Applying equation 2.61 to the complex spectrum of the input impulse response of an object yields the object's complex input impedance over a range of frequencies. This information is normally displayed as an impedance curve; the magnitude of the input

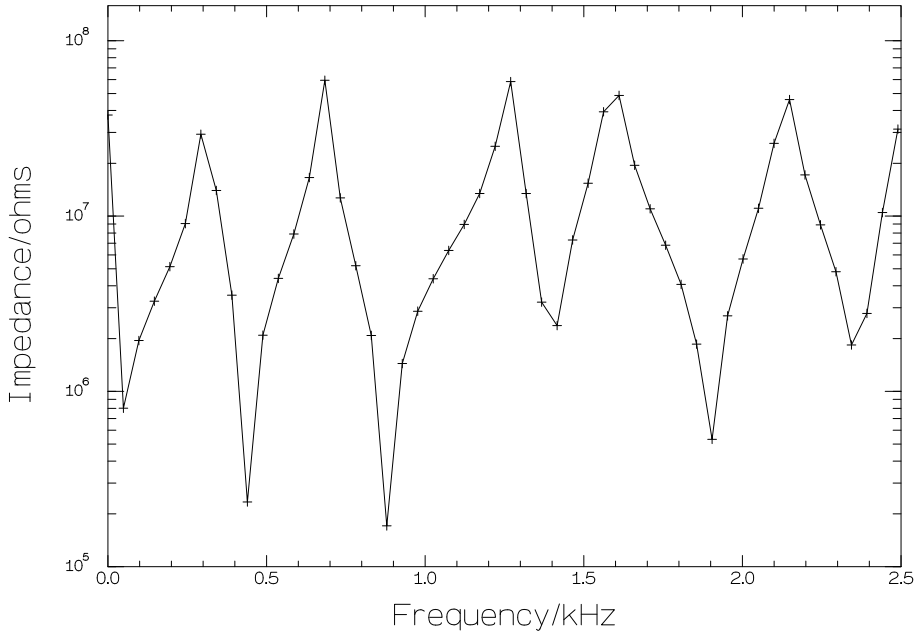

Figure 5.1: *Reflectometry impedance curve for stepped tube and coupler.*

impedance versus the frequency.

Figure 5.1 shows the impedance curve resulting from the application of equation 2.61 to the complex spectrum of the experimentally determined input impulse response (figure 3.17) of the stepped tube and coupler. The input impulse response was determined from measurements made at τ=16.5^{o}C. The crosses on the impedance curve indicate the frequencies at which the impedance was calculated.

When evaluating the input impedance directly from the input impulse response, it is not necessary to calculate the DC offset. The DC offset only changes the impedance at 0Hz.

5.2.1 Impedance curve frequency resolution

The frequency resolution of an impedance curve is the same as that of the input impulse response spectrum from which it was calculated. The frequency resolution of the input impulse response spectrum is determined by the sample length of the object reflections, which is constrained by the reflectometer source tube section l_1.

The reflectometer used to measure the input impulse response of the stepped tube and coupler has l_1=3.10m. To avoid source reflections being recorded, object reflections are sampled over a time period shorter than the time taken for the signal to travel the distance $2l_1$ (from the microphone to the loudspeaker and back). At a sampling frequency of 50kHz, when sampling is started 3ms before the object reflections reach the microphone, the maximum number of points over which object reflections can be sampled without recording any source reflections is 1024. This results in the spectrum of the object reflections and, therefore, the spectrum of the input impulse response having a maximum frequency resolution of 50000/1024=48.83Hz. This is the frequency resolution of the impedance curve shown in figure 5.1.

To improve the impedance curve frequency resolution by sampling the object reflections over a longer time period either the source tube section l_1 must be lengthened or the source reflections removed. Although increasing l_1 improves the resolution of the impedance measurement, the object reflections experience extra losses. This attenuation of the higher frequencies means that the impedance can only be accurately calculated at low frequencies. Chapter 7 details work concerned with removing the source reflections (by driving the loudspeaker in such a way as to absorb the incoming object reflections).

5.2.2 Zero-padding to improve impedance curve frequency resolution

Recalling the constraint that the input pulse and object reflections must be self-windowing, by the end of the input impulse response the signal should have decayed to zero. Therefore, the length of the response can be artificially increased, without having to increase the sample length of the object reflections, by padding it out to the desired length with zeroes (at 20μs intervals, corresponding to the sampling frequency of 50kHz). By padding the response out from 1024 points to 8192 points, the frequency resolution of

the impedance curve is improved to 50000/8192=6.10Hz. Note that before the input impulse can be zero-padded, the DC offset must first be removed. The offset need not be determined as accurately as for bore reconstruction; an estimation from visual examination of the input impulse response is usually sufficient.

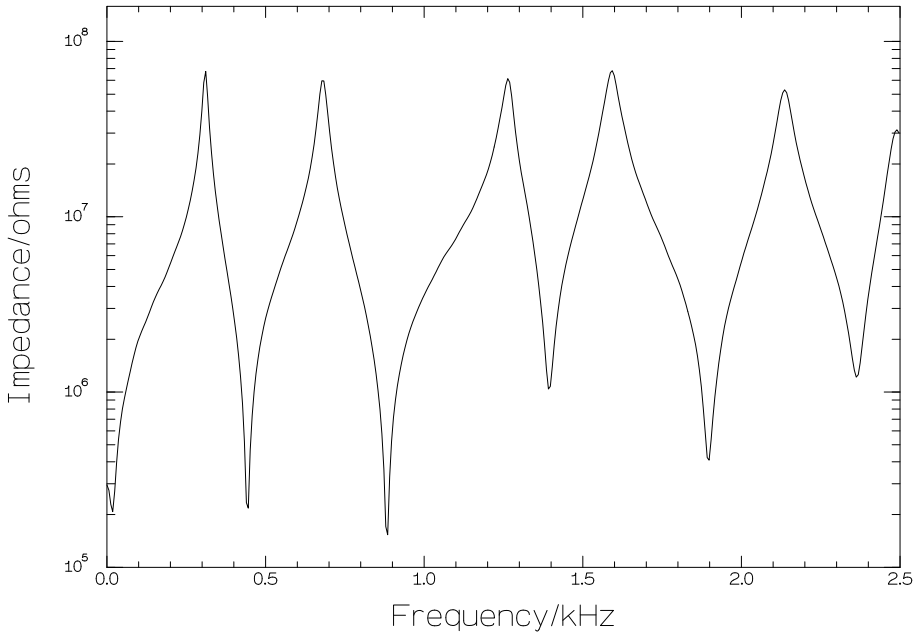

Figure 5.2: *Improved resolution reflectometry impedance curve for stepped tube and coupler.*

Figure 5.2 shows the impedance curve resulting from the application of equation 2.61 to the spectrum of the zero-padded input impulse response of the stepped tube and coupler. The input impulse response was zero-padded from 1024 points to 8192 points (after an estimated DC offset of 0.0016 had been removed) resulting in an impedance curve frequency resolution of 6.10Hz.

5.3 Swept sine wave method of measuring input impedance

A standard frequency domain method of measuring the input impedance of a tubular object is the swept sine wave method. As the name suggests, the object is excited

at its input by a sinusoidal pressure wave. The frequency of the excitation wave is increased and the pressure response at each frequency is recorded. Provided the excitation wave has a constant volume velocity, the pressure response is proportional to the input impedance of the object. No phase information is gained using this technique; only the magnitude of the input impedance is measured.

5.3.1 Experimental apparatus and its operation

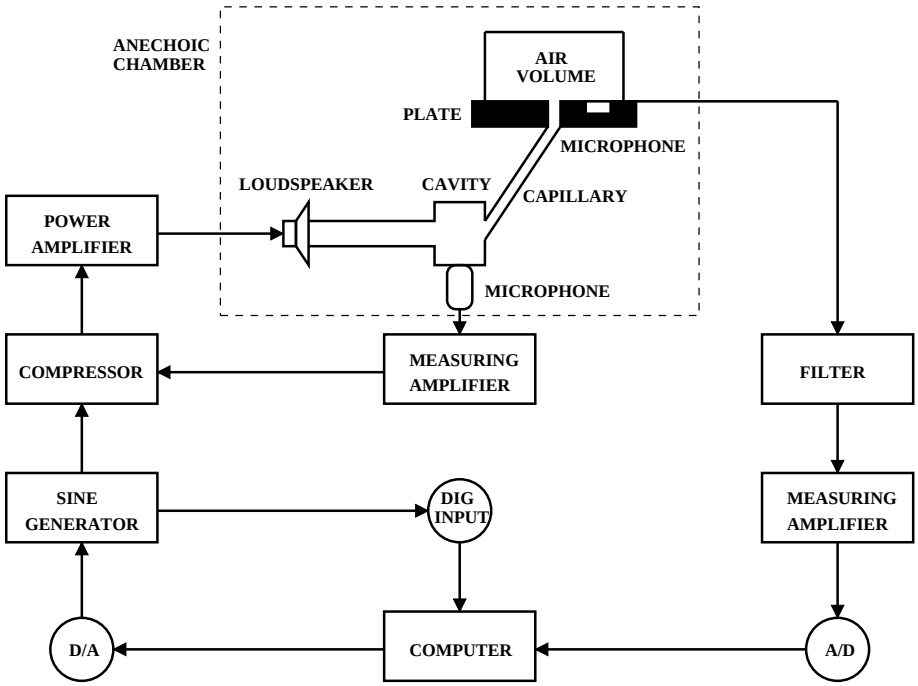

Figure 5.3: *Schematic diagram of experimental apparatus.*

Figure 5.3 shows a schematic diagram of the experimental apparatus used in the frequency domain measurement of the input impedance of a volume of air. The loudspeaker, the microphones and the volume of air under test are all mounted in an anechoic chamber, with the electronics in an adjoining room. This apparatus is similar to that described by Backus [1974, 1975].

A voltage is produced by a 12 bit D/A converter (located on an Iotech DaqBook data acquisition system controlled by a Viglen 486DX 66MHz PC) and used to drive

a Bruel and Kjaer Type 1023 sine generator. An inbuilt frequency meter measures the exact frequency of the generated wave and outputs the result, in binary coded decimal form, to the PC (via a digital input on the DaqBook). The generated sine wave is amplified by a Bruel and Kjaer Type 2706 power amplifier and used to feed a horn loudspeaker driver. A Bruel and Kjaer $\frac{1}{2}$ inch microphone monitors the pressure generated in a cavity mounted in front of the horn driver. This pressure signal is measured using a Bruel and Kjaer Type 2608 measuring amplifier and passed back to a compressor circuit in the sine generator. The compressor adjusts the sine generator output so as to maintain the cavity pressure constant. In the present case, this feedback loop ensures that the cavity pressure remains 130 ± 0.5 dB SPL over the frequency range 50Hz-4kHz. An annular capillary of length 69mm connects the cavity to a plate on which the volume of air under test is mounted. On the assumption that the impedance of the capillary is independent of frequency and much larger than the impedance of the volume of air, the system can be considered a source of constant volume velocity. The pressure in the volume of air, measured by a Knowles microphone embedded in the plate, is then directly proportional to the input impedance of the volume of air. This pressure signal is passed through a Kemo VBF/3 high pass filter set to 50Hz to remove low frequency noise (such as the noise associated with the closing of doors outside the laboratory). The filtered signal is displayed on a Bruel and Kjaer Type 2609 measuring amplifier which outputs a voltage proportional to the pressure displayed. This voltage is sampled by a 12 bit A/D converter located on the DaqBook and converted back to a pressure value by the PC. Given the volume velocity, the PC also calculates the input impedance according to equation 2.7.

A higher voltage is then produced by the D/A converter and used to drive the sine generator. A wave of higher frequency is generated and the above process is repeated. In this way, the input impedance of the volume of air under test is measured over a frequency range of approximately 1Hz to 4kHz. However, because the pressure in

the cavity is not constant below 50Hz, the range over which the input impedance is accurately measured is 50Hz-4kHz.

Determination of the volume velocity

To determine the volume velocity, a small volume was placed on the experimental apparatus. The impedance Z of a small volume V is given by:

$$Z = \frac{\rho c^2}{\omega V} \tag{5.1}$$

where ρ is the density of air, $c = 331.6\sqrt{1+\tau/273}$ m/s is the speed of sound in air, τ is the air temperature in ^{o}C, and ω is the angular frequency of the excitation sound wave.

Substituting equation 2.7 into equation 5.1 and rearranging gives:

$$U = \frac{p\omega V}{\rho c^2} = \frac{(10^{\frac{p(dB)}{20}}) \times 2 \times 10^{-5} \times \omega V}{\rho c^2} \tag{5.2}$$

where U is the volume velocity, p is the pressure in Pascals and $p(dB)$ is the pressure in decibels.

Thus by measuring the pressure at different frequencies, values of the volume velocity can be found.

For a volume of air $V = 1 \times 10^{-5}\text{m}^3$, the volume velocity measured at frequencies over the 50Hz-2.5kHz range was found to have an average value of $3 \times 10^{-8}\text{m}^3\text{s}^{-1}$.

5.3.2 Swept sine wave impedance curve

Figure 5.4 shows the impedance curve resulting from the measurement of the stepped tube and coupler using the swept sine wave method. The object measured essentially consisted of three cylindrical sections; a cylinder of length 50mm and radius 4.8mm (the coupler), a cylinder of length 129mm and radius 6.20mm (the first section of the stepped tube), and a cylinder of length 177mm and radius 9.25mm (the second section

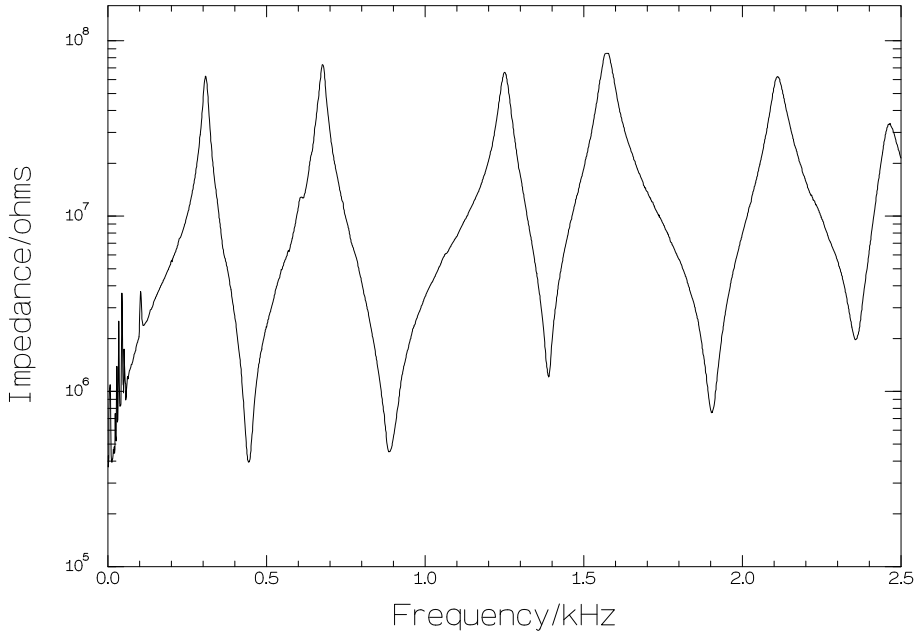

Figure 5.4: *Swept sine wave impedance curve for stepped tube and coupler.*

of the stepped tube). The measurement was made at an air temperature of 18.5^{o}C. The resolution of the impedance curve is approximately 1Hz.

5.4 Theoretical expression for input impedance

A theoretical expression for the complex input impedance of a cylinder is well known [Kinsler et al 1982]. Using this expression, a theoretical expression for the complex input impedance of an object consisting of three cylindrical sections can be derived.

5.4.1 Input impedance of a cylinder

Assuming plane wave propagation but including losses, the complex input impedance of a cylinder of length l and radius r is given by:

$$Z_{in} = \frac{\rho\omega}{\underline{k}\pi r^2}\left(\frac{\frac{Z_{load}\underline{k}\pi r^2}{\rho\omega} + j\tan\underline{k}l}{1 + j\frac{Z_{load}\underline{k}\pi r^2}{\rho\omega}\tan\underline{k}l}\right) \tag{5.3}$$

where Z_{in} is the complex input impedance, Z_{load} is the complex load impedance (at the end of the cylinder), ρ is the air density, ω is the angular frequency, and $\underline{k}$ is the complex propagation constant.

The complex propagation constant is defined as being:

$$\underline{k} = k - j\alpha = \frac{\omega}{c} - j\frac{1}{rc}\left(\sqrt{\frac{\eta\omega}{2\rho}} + (\gamma - 1)\sqrt{\frac{\kappa\omega}{2\rho C_p}}\right) \tag{5.4}$$

where γ is the ratio of the principal specific heats of air, C_p is the specific heat of air at constant pressure, η is the coefficient of shear viscosity of air, κ is the thermal conductivity of air, $c = 331.6\sqrt{1+\tau/273}$ m/s is the speed of sound in air, and τ is the air temperature.

For an open-ended cylinder, the load impedance is the radiation impedance which, for an unflanged end, is given by:

$$Z_{load} = 0.25\frac{\rho\omega}{\underline{k}\pi r^2}\underline{k}^2 r^2 + j0.6\frac{\rho\omega}{\underline{k}\pi r^2}\underline{k}r \tag{5.5}$$

Substituting equation 5.5 into equation 5.3 gives the input impedance of an open-ended cylinder:

$$Z_{in} = \frac{\rho\omega}{\underline{k}\pi r^2}\left(\frac{0.25\underline{k}^2 r^2 + j(0.6\underline{k}r + \tan\underline{k}l)}{(1 - 0.6\underline{k}r\tan\underline{k}l) + j0.25\underline{k}^2 r^2 \tan\underline{k}l}\right) \tag{5.6}$$

5.4.2 Input impedance of a stepped tube

Figure 5.5 shows a schematic diagram of a stepped tube consisting of three cylindrical sections, the last of which is open-ended.

The complex impedance Z_C at the boundary between the second and third cylindrical sections is simply the input impedance of the open-ended third cylinder (of length

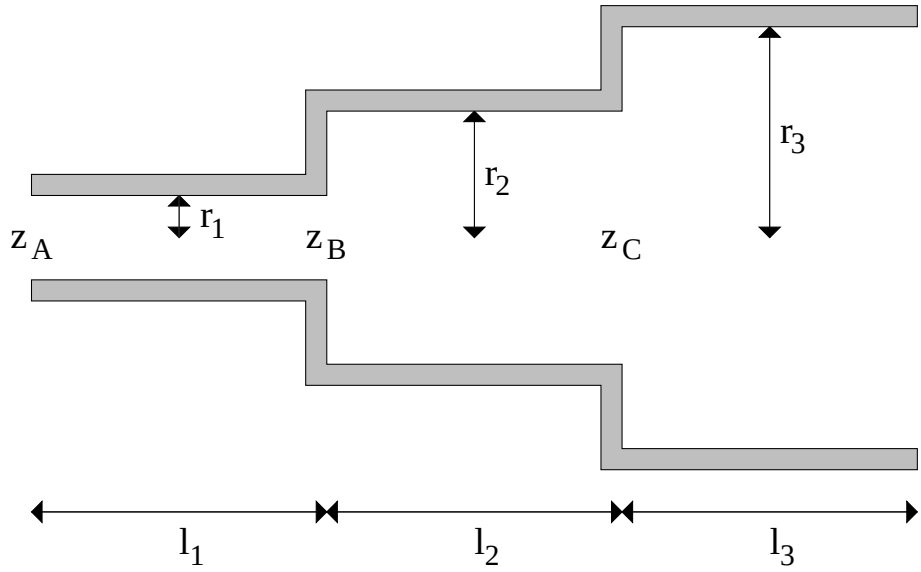

Figure 5.5: *Schematic diagram of a stepped tube consisting of three cylindrical sections.*

l_3 and radius r_3). Hence, substituting Z_C for Z_{in}, l_3 for l and r_3 for r in equation 5.6 gives:

$$Z_C = \frac{\rho\omega}{\underline{k}\pi r_3^2}\left(\frac{0.25\underline{k}^2 r_3^2 + j(0.6\underline{k}r_3 + \tan\underline{k}l_3)}{(1 - 0.6\underline{k}r_3\tan\underline{k}l_3) + j0.25\underline{k}^2 r_3^2\tan\underline{k}l_3}\right) \tag{5.7}$$

Note that in this case $\underline{k}$ is the propagation constant for the third cylindrical section which has radius r_3. Hence, $\underline{k}$ is found by substituting r_3 for r in equation 5.4.

The complex impedance Z_B at the boundary between the first and second cylindrical sections is simply the input impedance of the second cylinder (with a load impedance equal to Z_C). Hence, substituting Z_B for Z_{in}, l_2 for l, r_2 for r and Z_C (calculated using equation 5.7) for Z_{load} in equation 5.3 gives:

$$Z_B = \frac{\rho\omega}{\underline{k}\pi r_2^2}\left(\frac{\frac{Z_C\underline{k}\pi r_2^2}{\rho\omega} + j\tan\underline{k}l_2}{1 + j\frac{Z_C\underline{k}\pi r_2^2}{\rho\omega}\tan\underline{k}l_2}\right) \tag{5.8}$$

Again, note that in this case $\underline{k}$ is the propagation constant for the second cylindrical section which has radius r_2. Hence, $\underline{k}$ is found by substituting r_2 for r in equation 5.4.

Finally, the input impedance of the stepped tube Z_A is simply the input impedance of the first cylinder (with a load impedance equal to Z_B). Hence, substituting Z_A for

Z_{in}, l_1 for l, r_1 for r and Z_B (calculated using equation 5.8) for Z_{load} in equation 5.3 gives:

$$Z_A = \frac{\rho\omega}{\underline{k}\pi r_1^2}\left(\frac{\frac{Z_B\underline{k}\pi r_1^2}{\rho\omega} + j\tan\underline{k}l_1}{1 + j\frac{Z_B\underline{k}\pi r_1^2}{\rho\omega}\tan\underline{k}l_1}\right) \tag{5.9}$$

Again, note that in this case $\underline{k}$ is the propagation constant for the first cylindrical section which has radius r_1. Hence, $\underline{k}$ is found by substituting r_1 for r in equation 5.4.

5.4.3 Theoretical impedance curve

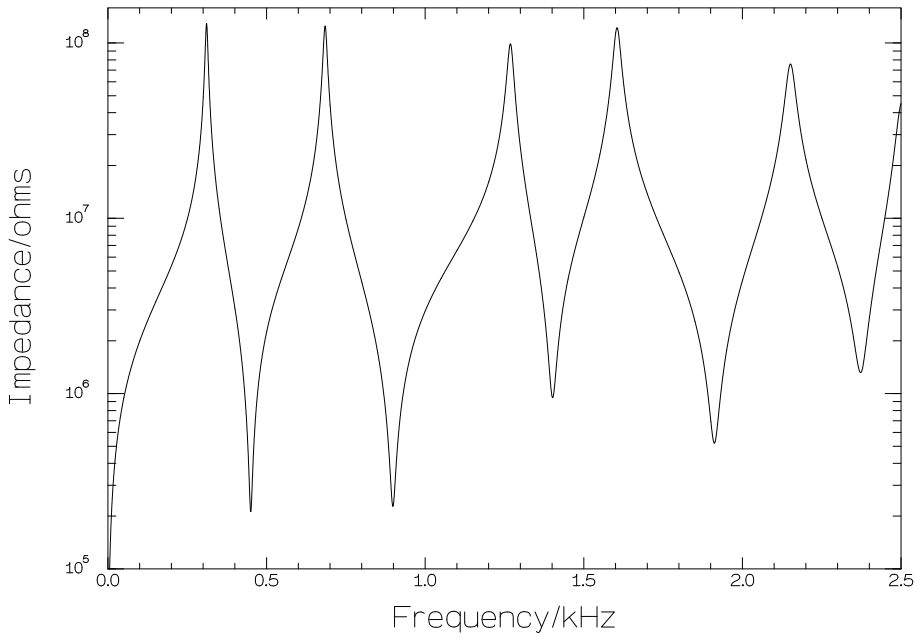

Figure 5.6: *Theoretical impedance curve for stepped tube and coupler.*

Figure 5.6 shows the impedance curve resulting when the dimensions of the stepped tube and coupler are substituted in to equations 5.7, 5.8 and 5.9 (i.e. l_1=50mm, r_1=4.8mm, l_2=129mm, r_2=6.20mm, l_3=177mm, r_3=9.25mm). A temperature of τ=16.5^{o}C was used.

5.5 Comparison of the different methods of measuring input impedance

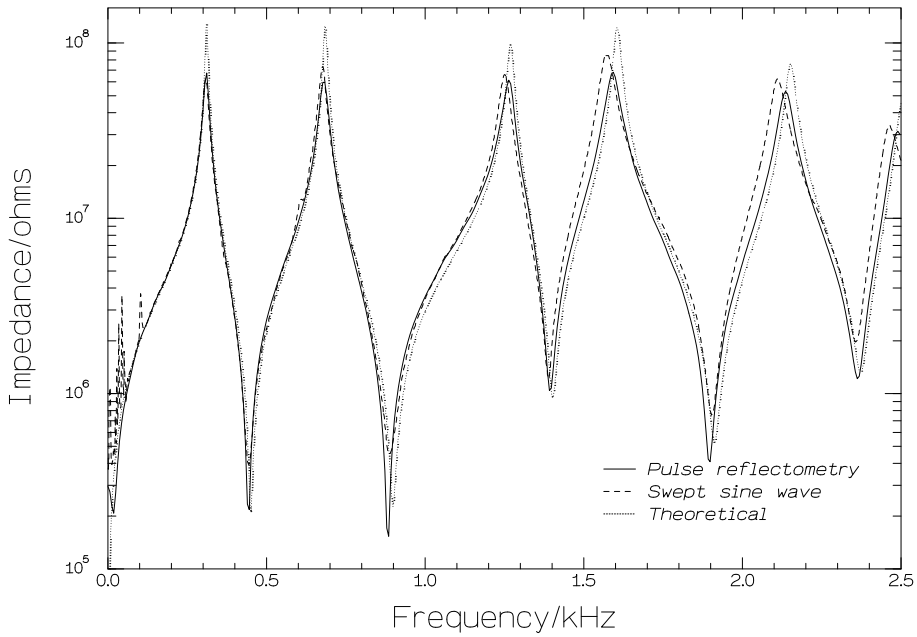

Figure 5.7: *Reflectometry, swept sine wave and theoretical impedance curves for stepped tube and coupler.*

Figure 5.7 compares impedance curves for the stepped tube and coupler obtained using the three different methods described. The impedance curves can be seen separately in figures 5.2, 5.4 and 5.6.

The basic shape of the three impedance curves is the same. However, at the maxima and minima, the curves diverge in both frequency and amplitude.

Comparing the reflectometry impedance curve with the theoretical curve, the frequencies of the maxima and minima of the reflectometry curve are all approximately 0.7% lower than those of the theoretical curve. This implies that, when measured on the pulse reflectometer, the stepped tube and coupler behave 0.7% longer than the directly measured length of 356mm (i.e. they behave 2.5mm longer). However, it was shown in section 4.3 that an object measured on the reflectometer can behave anything

up to $cT/2 = 3.4$mm longer than its directly measured length. Hence, the frequencies of the reflectometry impedance curve agree with those of the theoretical curve to within experimental error. The amplitudes of the peaks of the theoretical curve are uniformly higher than those of the reflectometry impedance curve.

Comparing the swept sine wave impedance curve with the theoretical curve, the frequencies of the maxima and minima of the swept sine wave curve are all lower than those of the theoretical curve. The frequencies are approximately 1% lower for the first two resonances and approximately 2% lower for the next three resonances, whilst the frequencies are approximately 1.5% lower for the first two antiresonances and approximately 1% lower for the next three antiresonances. One explanation for these differences is the false assumption that the source capillary has a frequency independent impedance. The capillary is 69mm long. When the wavelength of the excitation sine wave is reduced to four times this length (when the frequency reaches approximately 1.2kHz), resonance effects begin to occur. Hence, by the third of the stepped tube resonance peaks, capillary resonance effects are also significant. Another explanation is the false assumption that the stepped tube impedance is negligible compared with the capillary impedance. The impedance at the stepped tube resonance peaks is of the order of 10^8 ohms. This is significant when compared with the capillary impedance which is of the order of 10^9 ohms. The amplitudes of the peaks of the theoretical curve are uniformly higher than those of the swept sine wave impedance curve.

5.6 Brass instrument impedance curves

To illustrate the use of impedance curves in the study of musical instruments, results for two 19th century cornets (measured using both the acoustic pulse reflectometry method and the swept sine wave method) are presented and discussed.

The reflectometry impedance curves were calculated (using equation 2.61) from zero-padded input impulse response measurements made using the longer source tube

reflectometer described in section 4.5. The longer source tube ensured that the reflections returning from each of the cornets were completely recorded. The two sets of reflections were sampled at 50kHz with sample lengths of 2048 points; the resultant input impulse responses were both zero-padded up to 8192 points.

Each cornet was connected to the reflectometer source tube or the microphone and sine wave source capillary using one of the couplers shown in figure 3.6. These couplers have approximately the same volume as a cornet mouthpiece. Thus, the impedance curves measured for each cornet are a good approximation to the impedance curves for each cornet with its mouthpiece attached.

5.6.1 Boosey & Co. cornet

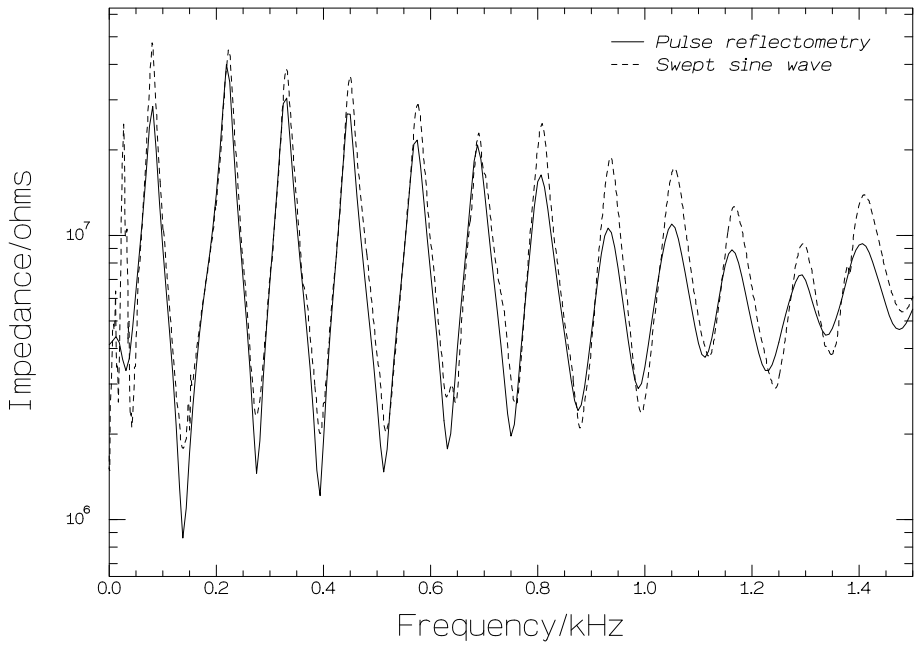

Figure 5.8: *Impedance curves for Boosey & Co. cornet with no valves operated.*

Figure 5.8 shows impedance curves for a Boosey & Co. 'Acme' model cornet in B♭ with no valves operated (both impedance curves were measured at 17.4^{o}C). This instrument is a 'standard model' cornet with a cylindrical bore through the valves. It has had its playing pitch lowered from A_4=452.5 Hz to 440 Hz by the insertion of

cylindrical extension pieces in its main tuning-slide.

5.6.2 Rudall Carte cornet

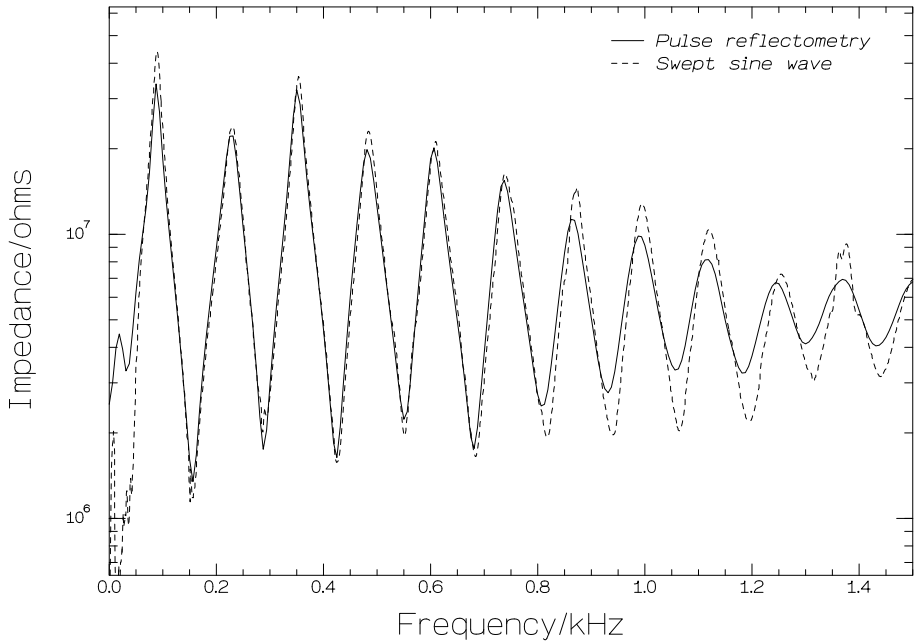

Figure 5.9: *Impedance curves for Rudall Carte cornet with no valves operated.*

Figure 5.9 shows impedance curves for a Rudall Carte 'conical bore model' cornet in B♭ with no valves operated (again, both impedance curves were measured at 17.4^{o}C). This instrument is characterised by its bore which increases in diameter through the valve pistons and between one valve and the next. Compared with the Boosey & Co. cornet, the bore is narrower and more gently tapering between the mouthpiece receiver and the valves. The instrument was designed to play at a pitch of A_4=452.5Hz.

5.6.3 Discussion

Comparison of the curves for the two cornets clearly shows that all the peaks are at slightly lower frequencies for the Boosey & Co. cornet which has had its pitch lowered.

In fact, analysis of the frequencies of the peaks suggest that the Boosey & Co. cornet should play at a pitch around 116 cents lower than the Rudall Carte instrument. Playing tests confirm that this is indeed the case. It appears that the high pitch instrument was designed to play at A_4 with its tuning-slides partially withdrawn.

Chapter 6

Investigation of leaks

6.1 Introduction

In this chapter, the effect of a leak in the wall of a tubular object on the object's bore reconstruction and input impedance is investigated. Examination of the bore reconstruction leads to a method for detecting the presence of a leak in any tubular object. It is shown that for certain objects, the axial position of a leak can also be determined. Finally, for the special case of a cylindrical pipe, a method for determining the size of a leak from input impedance measurements is described.

6.2 Detecting the presence of a leak in a tubular object

A small leak in the wall of an object being measured on the pulse reflectometer can be thought of as a side branch of complex impedance (discussed further in section 6.4). The side branch presents a reduction in the magnitude of the impedance seen by the incoming pulse. This change is similar to the change in impedance caused by a widening of the bore. Hence, the leak appears as an expansion in the bore reconstruction.

A leaking object reconstructed using the cylindrical connector method of DC offset removal (described in section 4.2.2) will give a reconstruction which is correct as far

as the position of the leak, after which it will expand to an extent dependent on the size of the leak.

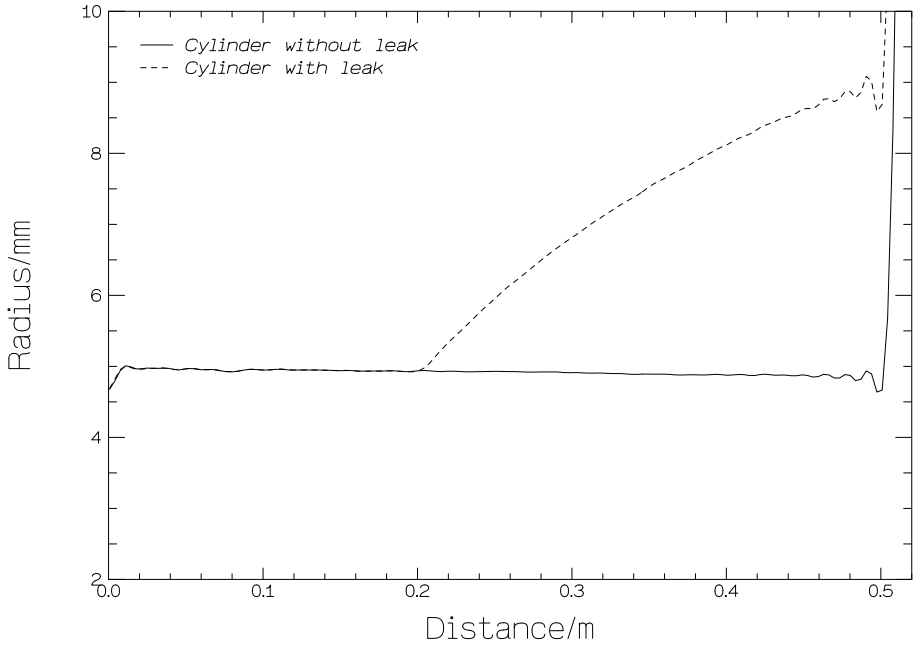

Figure 6.1: *Reconstructed bore profile of cylindrical pipe with and without leak.*

As an example, figure 6.1 shows two bore reconstructions of a 501mm long cylindrical pipe of internal radius 5.0mm and wall thickness 1.0mm. In the first case the pipe wall contains a 0.5mm radius circular hole (drilled 200mm from the start of the pipe) whilst in the second case this hole is sealed. The reconstructions were calculated from measurements made at $\tau = 17.0^{o}$C using the original pulse reflectometer (described in section 3.2).

Although the DC offset was calculated and removed using the cylindrical connector method, the connector is not displayed on the graph. The start of the graph corresponds to the start of the cylindrical pipe, with the expansion from the 4.7mm radius cylindrical connector clearly visible. The two reconstructions coincide until the position of the leak, 200 mm from the start of the pipe. From this point onwards, the reconstruction of the leaking pipe expands, reaching a final radius of 9.0mm, whilst the reconstruction of the airtight pipe continues at the correct radius of 5.0mm.

From these observations, a method of leak detection was devised. If the bore reconstruction of an object passes through a directly measured radius at a known point towards the end of the object, the object can be considered airtight. If, however, the reconstruction has a larger radius than the directly measured radius, there must be a spurious expansion present and the object must contain a leak.

Of course, for the simple case of a cylindrical pipe, the mere presence of a reflection must indicate the presence of a leak. Hence, the method of leak detection described above, whilst valid, is not strictly necessary. However, when dealing with more complicated tubular objects whose bores expand and contract, reflections can result from either a change in bore radius or from a leak in the object's wall. For such objects, the method of leak detection described above must be used.

6.2.1 Rudall Carte cornet

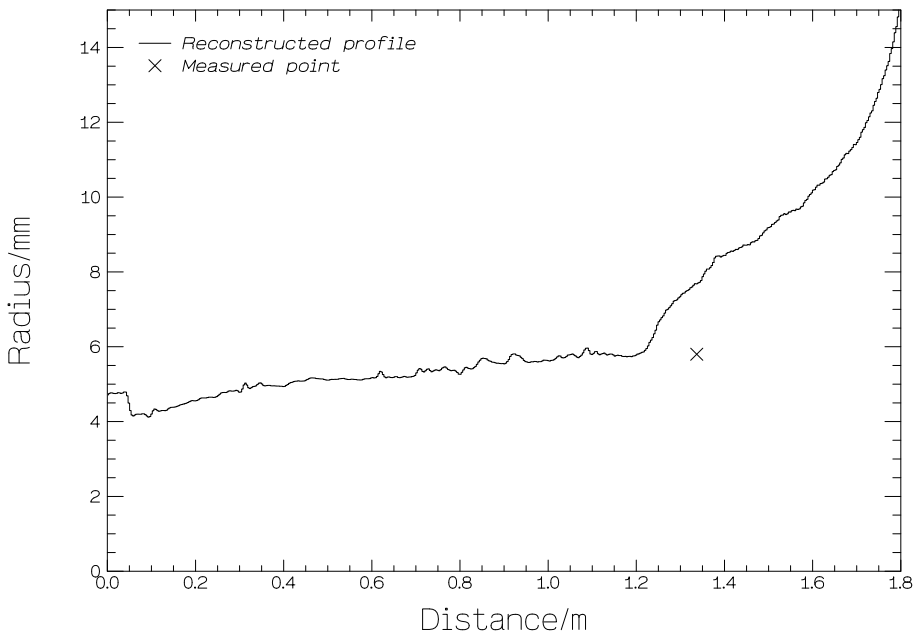

Figure 6.2: *Reconstructed bore profile of Rudall Carte cornet with leak. All three valves operated.*

Figure 6.2 shows a bore reconstruction of a Rudall Carte 'conical bore model'

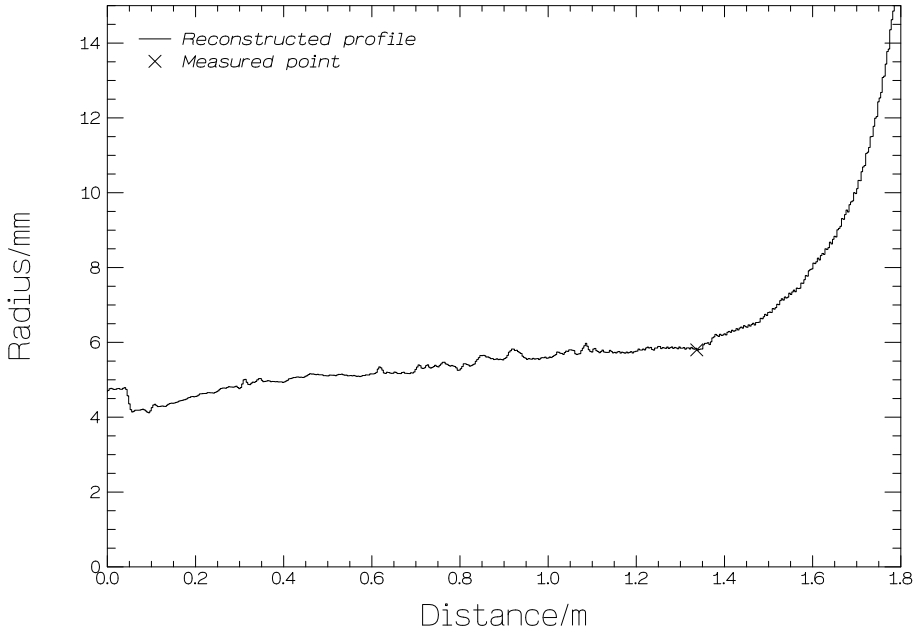

Figure 6.3: *Reconstructed bore profile of Rudall Carte cornet with leak sealed. All three valves operated.*

cornet in B♭ with all three valves operated (calculated from measurements made at $\tau = 19.9^{o}$C using the longer source tube reflectometer described in section 4.5).

For the first 100mm, the graph shows the coupler which was chosen (from the six couplers displayed in figure 3.6) to couple the cornet to the cylindrical connector with the best possible fit. The coupler penetrated a distance of 35.3mm into the instrument. From 100mm to 1800mm, the graph shows the cornet profile (starting from a position 35.3mm in from the mouthpiece end).

At a tuning slide towards the end of the instrument (the axial position on the graph is 1337mm), the internal radius of the cornet was measured using calipers. The directly measured radius of 5.8 ± 0.05mm was found to be significantly less than the reconstructed radius of 7.7mm at the same axial position, suggesting the presence of a leak before this position.

Examination of the cornet revealed a crack in the third valve tuning slide. Figure 6.3 shows a second bore reconstruction of the same Rudall Carte ‘conical bore

model' cornet, made this time with the crack sealed (calculated from measurements made at $\tau = 17.2^{o}$C using the longer source tube reflectometer). The reconstruction and the direct measurement are now seen to be in good agreement, showing that there are no other significant leaks in the section of the instrument before the measured point.

6.2.2 King trombone bell section

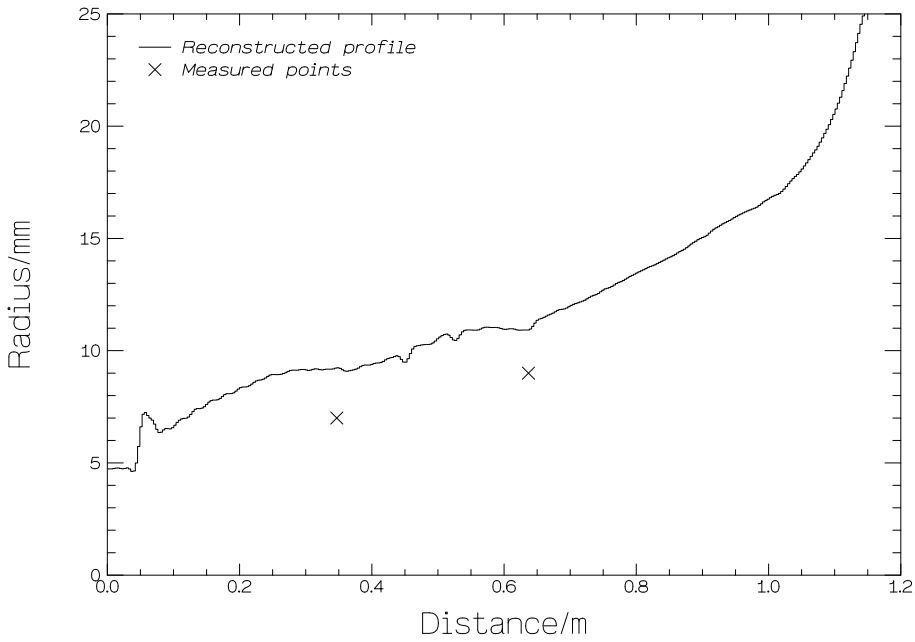

Figure 6.4: *Reconstructed bore profile of King trombone bell section with leak.*

Figure 6.4 shows a bore reconstruction of the bell section of a King trombone (calculated from measurements made at $\tau = 18.3^{o}$C using the longer source tube reflectometer).

The coupler shown in figure 3.5 was used to couple the cylindrical connector to the trombone bell section. The graph shows the coupler for the first 50mm and then the trombone profile from 50mm to 1200mm.

At two different positions (347mm and 637mm on the graph), the internal radius of the trombone was measured using calipers. The directly measured radii of 7.0 ± 0.05mm and 9.0 ± 0.05mm were found to be significantly less than the recon-

structed radii of 9.2mm and 10.9mm at the same axial positions, suggesting the presence of a leak before these positions.

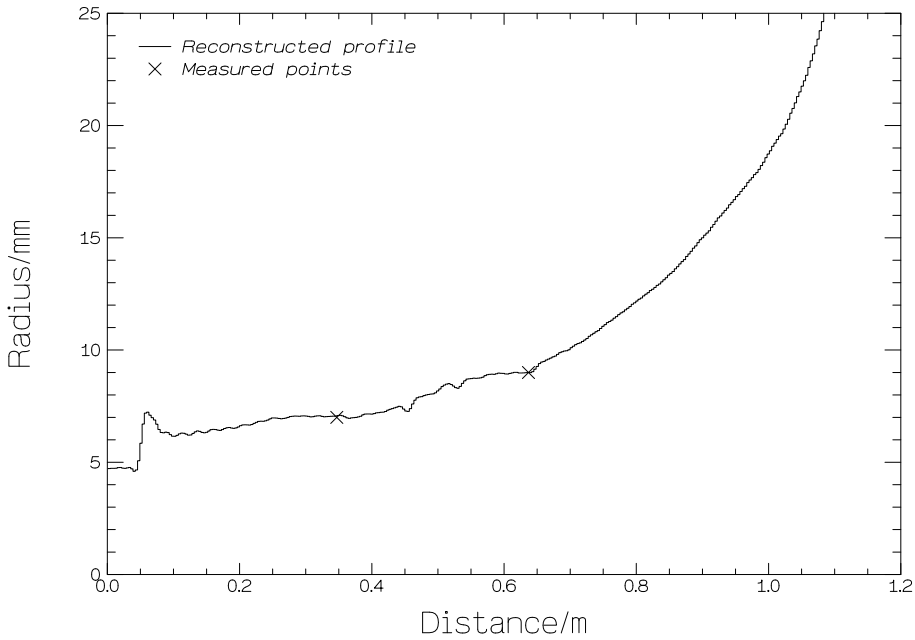

Figure 6.5: *Reconstructed bore profile of King trombone bell section with leak sealed.*

Examination of the trombone revealed a crack near the start of the bell section. Figure 6.5 shows a second bore reconstruction of the same King trombone bell section, made this time with the crack sealed (calculated from measurements made at $\tau = 18.3^{o}$C using the longer source tube reflectometer). The reconstruction and the direct measurement are now seen to be in good agreement, showing that there are no other significant leaks in the section of the instrument before the measured point.

6.3 Identifying the axial position of a leak in a tubular object

For an object whose entire bore profile is not known in advance, an expansion in the reconstruction due to a leak may be indistinguishable from an actual widening of the bore. Hence, although the presence of a leak can be detected, its axial position may be

difficult to determine.

For certain objects, the axial position of a leak can be found by producing two bore reconstructions: the first with the object's input coupled to the cylindrical connector, the second with the object reversed so that its output is coupled to the cylindrical connector.

The first reconstruction starts at the object's input, ends at the object's output and is correct up until the position of the leak (after which it expands to an extent dependent on the leak size). The second reconstruction starts at the object's output, ends at the object's input and is correct up until the position of the leak (after which it expands to an extent dependent on the leak size). Therefore, reversing the second reconstruction and superimposing it on the first yields a complete profile. The axial position of the centre of the leak is the point where the two reconstructions coincide.

This method can be applied directly to a leaking object whose output and input radii are similar (e.g. a flute) but not to one whose output radius is significantly larger (e.g. an instrument with a flaring bell). The reason for this is the practical difficulty of coupling the output end of the object to the cylindrical connector.

The method can, however, be used to examine a section of an object such as a trumpet. The first reconstruction is produced, as before, by coupling the mouthpiece end of the trumpet to the cylindrical connector. The second reconstruction is produced in a slightly different way to that described previously. Instead of attempting to couple the trumpet bell to the cylindrical connector, the coupling is made at a tuning-slide towards the output end of the instrument. In this way, the instrument bell is bypassed, removing the coupling problem but restricting the examination of the trumpet to the section between the mouthpiece end and the tuning-slide.

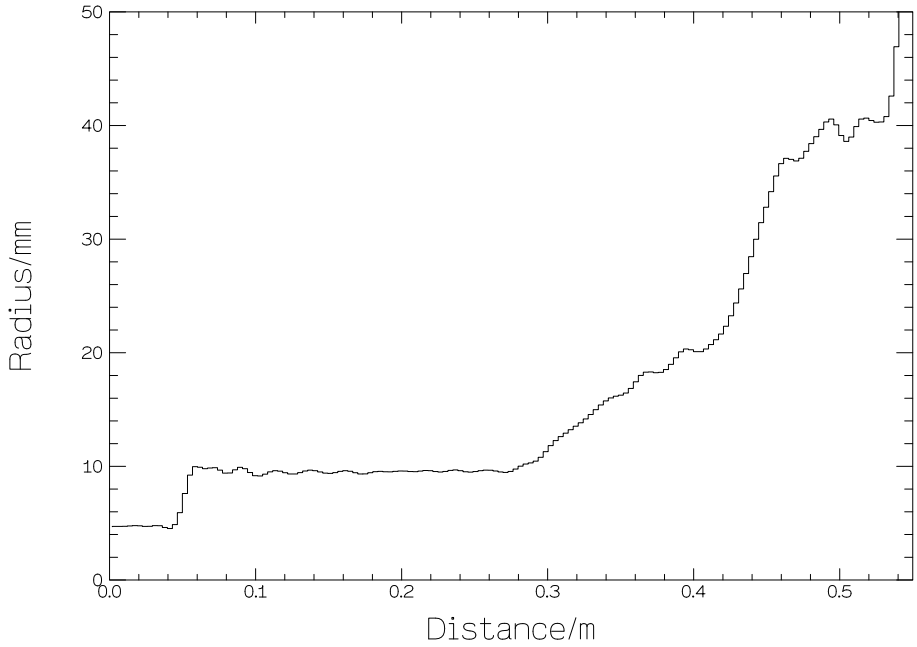

Figure 6.6: *Reconstructed bore profile of Blessing flute with leaking pad. Input end of flute coupled to cylindrical connector.*

6.3.1 Blessing flute

Figure 6.6 shows a bore reconstruction of a Blessing flute with all the pads closed but with one leaking. To measure the flute, the head joint was removed and the input end of the flute body was coupled to the cylindrical connector (using the coupler shown in figure 3.5). The reconstruction was calculated from measurements made at $\tau = 19.3^{o}$C using the longer source tube reflectometer.

The graph shows the coupler for the first 50mm and then the flute profile, starting with the input end of the flute at 50mm and ending with the output end at 535mm. The reconstruction is correct up until the position of the leak, after which it expands spuriously.

Figure 6.7 shows a bore reconstruction of the same Blessing flute. In this case, the flute was turned around and the output end of the flute body was coupled to the cylindrical connector. The reconstruction was calculated from measurements made at $\tau = 19.5^{o}$C using the longer source tube reflectometer.

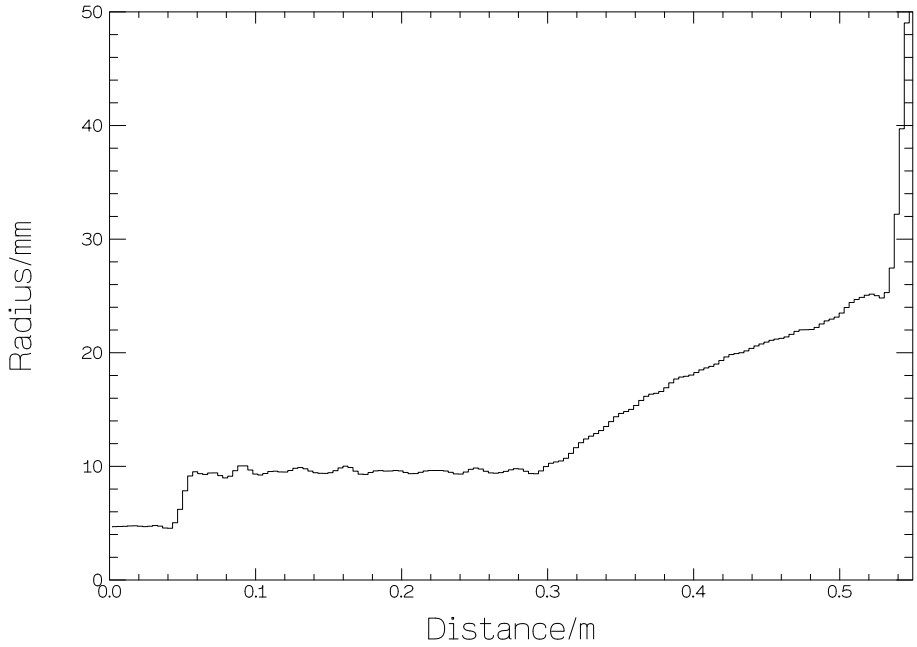

Figure 6.7: *Reconstructed bore profile of Blessing flute with leaking pad. Output end of flute coupled to cylindrical connector.*

The graph shows the coupler for the first 50mm and then the flute profile, starting with the output end of the flute at 50mm and ending with the input end at 535mm. The reconstruction is correct up until the position of the leak, after which it expands spuriously.

Figure 6.8 shows the bore reconstruction of figure 6.7 reversed and superimposed on the bore reconstruction of figure 6.6. The result is a complete reconstructed profile of the Blessing flute with its head joint removed and all the pads closed. On the graph, the axial position of the centre of the leaking pad is approximately 285mm. The first 50mm of the graph is the coupler so the leaking pad is 235mm from the input end of the flute. Direct measurement confirmed this prediction.

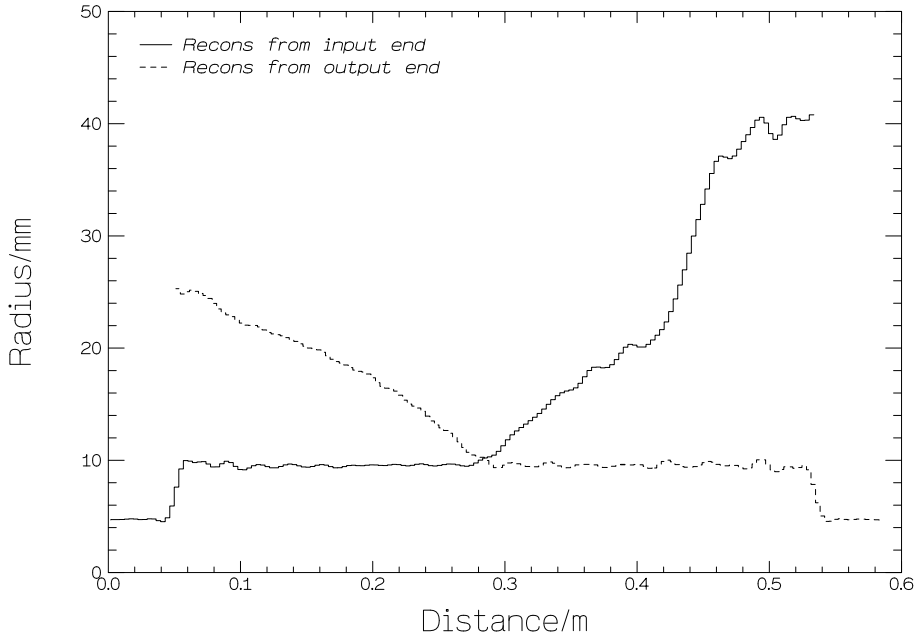

Figure 6.8: *Reconstructed bore profiles of Blessing flute with leaking pad.*

6.4 Evaluating the size of a leak in a cylindrical pipe

The size of a hole in a leaking cylindrical pipe can be calculated from the pipe's complex input impedance (measured using the pulse reflectometer), provided the axial position of the leak and the radius, length and wall thickness of the cylinder are known.

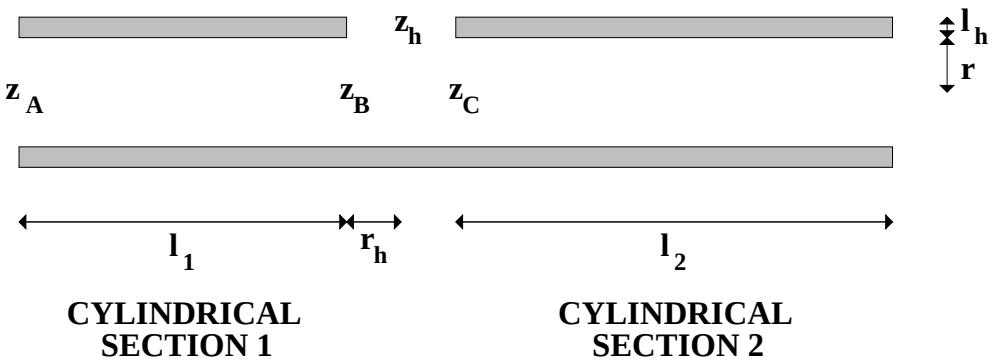

Figure 6.9: *Schematic diagram of a leaking cylindrical pipe*

Figure 6.9 shows a schematic diagram of a cylindrical pipe with a sidehole. To find the radius of the hole r_h, the leaking pipe (of radius r and wall thickness l_h) is thought of as consisting of two non-leaking cylinders of lengths l_1 and l_2, one before

the hole and one after. From this model, an expression is found for the impedance of the hole Z_h in terms of the input impedance of the leaking pipe Z_A. The calculated hole impedance is then substituted into a theoretical expression which gives the radius of a hole in terms of its impedance.

6.4.1 Impedance of the hole in a leaking cylinder

Expression of Z_h in terms of Z_B and Z_C

The impedance at the end of the first cylindrical section, Z_B, is made up of contributions from the input impedance of the second cylindrical section, Z_C, and the impedance of the hole, Z_h. The impedance of the hole can be expressed in terms of the impedances Z_B and Z_C:

$$\frac{1}{Z_h} = \frac{1}{Z_B} - \frac{1}{Z_C} = \frac{Z_C - Z_B}{Z_B Z_C} \tag{6.1}$$

Thus

$$Z_h = \frac{Z_B Z_C}{Z_C - Z_B} \tag{6.2}$$

Evaluation of Z_B

Equation 5.3 gives the input impedance of a non-leaking cylinder with a load impedance at its end. The first cylindrical section, of length l_1, is such a cylinder, with an input impedance Z_A and a load impedance Z_B. By substituting Z_A for Z_{in}, l_1 for l, Z_B for Z_{load} and rearranging, an expression is obtained for the load impedance Z_B (i.e. the impedance at the end of the first cylindrical section) in terms of the input impedance of the leaking pipe, Z_A:

$$Z_B = \frac{Z_A - j\dfrac{\rho\omega}{\underline{k}\pi r^2}\tan \underline{k} l_1}{1 - j\dfrac{Z_A \underline{k}\pi r^2}{\rho\omega}\tan \underline{k} l_1} \tag{6.3}$$

where $\underline{k}$ is the propagation constant given by equation 5.4.

Evaluation of Z_C

Equation 5.6 gives the input impedance of a open-ended non-leaking cylinder. The second cylindrical section, of length l_2, is such a cylinder, with an input impedance Z_C. Substituting Z_C for Z_{in} and l_2 for l in equation 5.6 gives:

$$Z_C = \frac{\rho\omega}{\underline{k}\pi r^2}\left(\frac{0.25\underline{k}^2r^2 + j(0.6\underline{k}r + \tan\underline{k}l_2)}{(1 - 0.6\underline{k}r\tan\underline{k}l_2) + j0.25\underline{k}^2r^2\tan\underline{k}l_2}\right) \tag{6.4}$$

where $\underline{k}$ is the propagation constant given by equation 5.4.

Evaluation of Z_h

Substituting equations 6.3 and 6.4 into equation 6.2 gives the complex impedance of the hole, Z_h, in terms of the complex input impedance of the leaking pipe, Z_A (measured using the pulse reflectometer).

6.4.2 Radius of the hole in a leaking cylinder

The complex impedance of a sidehole can also be expressed theoretically in terms of its depth l_h and radius r_h [Kinsler et al 1982]:

$$Z_h = \frac{\rho\omega\underline{k}_h}{4\pi} + j\frac{\rho\omega(l_h + Er_h)}{\pi r_h^2} \tag{6.5}$$

where $E = 1.595 - 0.58\left(\frac{r_h}{r}\right)^2$ is the sum of the inner and outer end corrections for a hole set flush with the cylinder wall. The inner end correction $E_i = \frac{\pi}{4}\left(1 - 0.74\left(\frac{r_h}{r}\right)^2\right)$ is derived in [Keefe 1982]. The outer end correction E_o is also discussed in the paper by Keefe where measured values [Benade and Murday 1967] ranging from 0.8 to 0.82 are quoted. A value of E_o=0.81 is used here.

The propagation constant is denoted by $\underline{k}_h$ because the wave is propagating in the hole. $\underline{k}_h$ is found by substituting the hole radius r_h for r in equation 5.4.

Expanding equation 6.5 gives:

$$Z_h = \frac{\rho\omega k}{4\pi} - j\frac{\rho\omega}{4\pi r_h c}\left(\sqrt{\frac{\eta\omega}{2\rho}} + (\gamma-1)\sqrt{\frac{\kappa\omega}{2\rho c_p}}\right) + j\frac{\rho\omega}{\pi r_h^2}\left(l_h + 1.595 r_h - 0.58\frac{r_h^3}{r^2}\right) \tag{6.6}$$

Considering only the imaginary part of the impedance Z_h:

$$Z_{himag} = \frac{\rho\omega l_h}{\pi r_h^2} + \frac{1.595\rho\omega}{\pi r_h} - \frac{0.58\rho\omega r_h}{\pi r^2} - \frac{\rho\omega}{4\pi r_h c}\left(\sqrt{\frac{\eta\omega}{2\rho}} + (\gamma-1)\sqrt{\frac{\kappa\omega}{2\rho c_p}}\right) \tag{6.7}$$

Rearranging gives a cubic equation for r_h:

$$r_h^3 + \left[\frac{1.724\pi r^2 Z_{himag}}{\rho\omega}\right] r_h^2 + \left[\frac{0.431 r^2}{c}\left(\sqrt{\frac{\eta\omega}{2\rho}} + (\gamma-1)\sqrt{\frac{\kappa\omega}{2\rho c_p}}\right) - 2.75 r^2\right] r_h - \left[1.724 l_h r^2\right] = 0 \tag{6.8}$$

Substituting the value for Z_{himag} calculated in the previous section from the leaking pipe's complex input impedance (measured using the pulse reflectometer) into equation 6.8 and solving [Press et al 1988], yields a value for the hole radius.

The success of equation 6.8 in predicting the radius of the hole will clearly depend on a suitable choice of frequency. A small hole has little effect on the impedance of the air column if it is in the vicinity of a pressure node; thus, at frequencies which correspond to this condition the prediction of hole size from impedance measurement can be expected to break down. Likewise, the hole has little effect on the acoustical properties of the pipe at frequencies much above the cutoff frequency, which is determined by the radii of the main cylinder and the hole. This sets an upper limit on the usable frequency range.

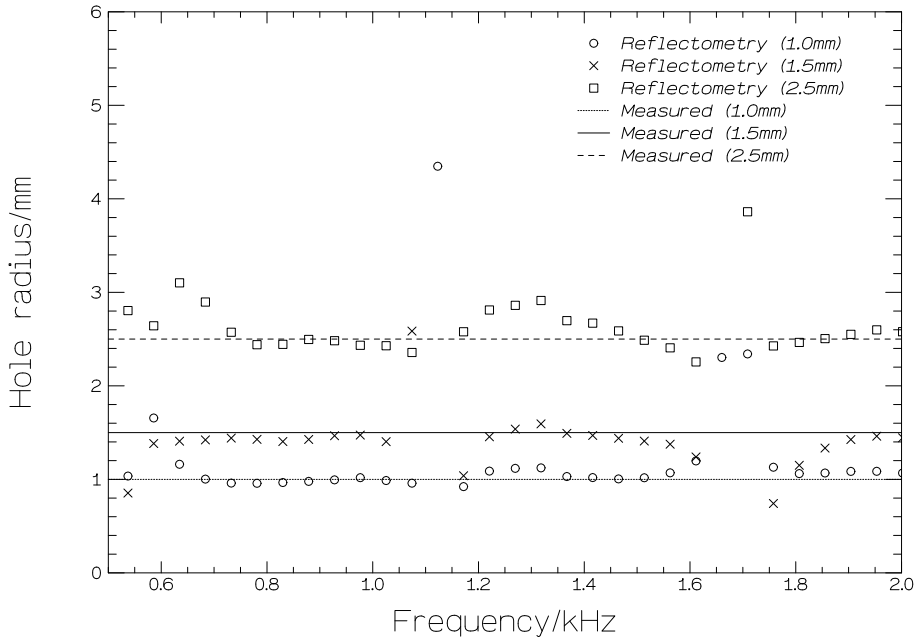

Figure 6.10: *Comparison of predicted and measured hole radii at various frequencies.*

6.4.3 Practical predictions of hole radii.

The complex input impedance of a leaking cylindrical pipe of length 501mm, internal radius 5.0mm and wall thickness 1.0mm was calculated for three different size of hole, from measurements made at $\tau = 17.0^{o}$C using the original pulse reflectometer. The input impulse response was not zero-padded before the input impedance calculation. In each case, the hole was 200mm from the start of the pipe. Figure 6.10 illustrates the predictions for the three different hole sizes, calculated from the three complex input impedance measurements using equations 6.2, 6.3, 6.4 and 6.8. Graphing the predicted radius as a function of frequency in this way is useful, since the divergences of the prediction allow an immediate identification of the frequencies for which pressure nodes coincide with the hole. In the present case, the divergences occur at integer multiples of 570Hz, as expected from the known geometry. The predictions also start to break down, although less dramatically, above approximately 2000Hz, which is in the vicinity of the cutoff frequency for the largest hole.

As a prescription for selecting valid data for predicting the effective radius of a leak

in an open ended pipe measured in this way, it is suggested that the frequency F of the first divergence is estimated by inspection of the graph. Values of r_h in the frequency ranges 1.25F-1.75F and 2.25F-2.75F are then averaged to give a best prediction of the effective radius of the hole.

Measured radius	Predicted radius (averaged over 712.5-997.5Hz)	Predicted radius (averaged over 1282.5-1567.5Hz)
1.0±0.05mm	0.979±0.02mm	1.044±0.04mm
1.5±0.05mm	1.440±0.02mm	1.463±0.07mm
2.5±0.05mm	2.479±0.05mm	2.627±0.16mm

Table 6.1: *Comparison of predicted hole radii (averaged over specific frequency ranges) and measured hole radii.*

Predictions drawn in this way from figure 6.10 are compared in table 6.1 with direct measurements using calipers. Six predicted values are averaged in each frequency range; the uncertainties quoted are single standard deviations. For each of the holes, the predicted radius agrees with that measured directly within experimental error.

Chapter 7

Measurement of longer objects

7.1 Introduction

In chapter 3, a method for measuring the input impulse response of a tubular object was described. The method involved injecting a sound pulse into the object, via a source tube, and then measuring the resultant reflections. However, after the object reflections passed the microphone embedded in the source tube wall, they were further reflected by the loudspeaker. These source reflections returned to the microphone, restricting the time over which the object reflections could be accurately sampled to $2l_1/c$ (the time for the first of the object reflections to travel the distance from the microphone to the loudspeaker and back). This in turn put a constraint on the length of object that could be measured. The input impulse response of the object was obtained by deconvolving the object reflections with the input pulse shape (measured by reflection from a rigid termination). From the input impulse response, the object's bore profile and input impedance could be calculated.

Various methods for measuring the input impulse responses of longer objects have been proposed. The simplest method was described in section 4.5 where a longer source tube section l_1 was used, allowing the object reflections to be accurately sampled over a longer time period. Unfortunately, lengthening the source tube only pro-

vides a partial solution to the problem. There is a limit to how much l_1 can be increased before the attenuation of the input pulse as it travels from loudspeaker to object becomes too great for accurate results to be obtained.

An alternative method was proposed by Marshall [1992a]. A sound pulse is again injected into the object but in this case the microphone signal is sampled until all the object and source reflections decay to zero. The signal is described as a geometric series. The first term in the series represents the pulse passing the microphone on its way down the source tube, the second term represents the object reflections travelling back up the source tube (the convolution of the input pulse with the input impulse response of the object), the third term represents the source reflections travelling down the source tube (the convolution of the object reflections with the input impulse response of the loudspeaker), the fourth term represents the source reflections being further reflected by the object and travelling back up the source tube (the convolution of the source reflections with the input impulse response of the object), and so on. Given the input impulse response of the loudspeaker, the input impulse response of the object can be calculated from the sum of the geometric series. Although reasonable reconstructions were made using this method, the measurements were not as accurate as those made on a longer source tube reflectometer. Similar work carried out by Amir et al [1995b] suggested that the method is very sensitive to mains noise.

A more established method involves the introduction of a second microphone into the source tube wall. First mentioned by Schroeder [1967], the concepts are described in greater detail by Louis et al [1993]. As before, a sound pulse is injected into the object via the source tube. From the pressures measured by the two microphones, the waves travelling down the source tube into the object (the input pulse and any source reflections) are separated from the waves travelling back up the source tube (the reflection of the input pulse and source reflections by the object). The input impulse response of the object is found by deconvolving the signal returning from the object

with the signal probing the object. Although good results were reported using this method, a correction procedure had to be employed to prevent reconstructions from becoming unstable with increasing axial distance. Even with the correction procedure, accurate and stable reconstructions were only achieved when the pressure of the input pulse was of the order of 100 times greater than the noise level.

Amir [1995a] put forward another idea for measuring the input impulse response of a longer object. He suggested that the formation of source reflections could be prevented by building a reflectometer with the loudspeaker positioned in the source tube wall (see figure 7.1). A sound pulse produced by the loudspeaker would travel two

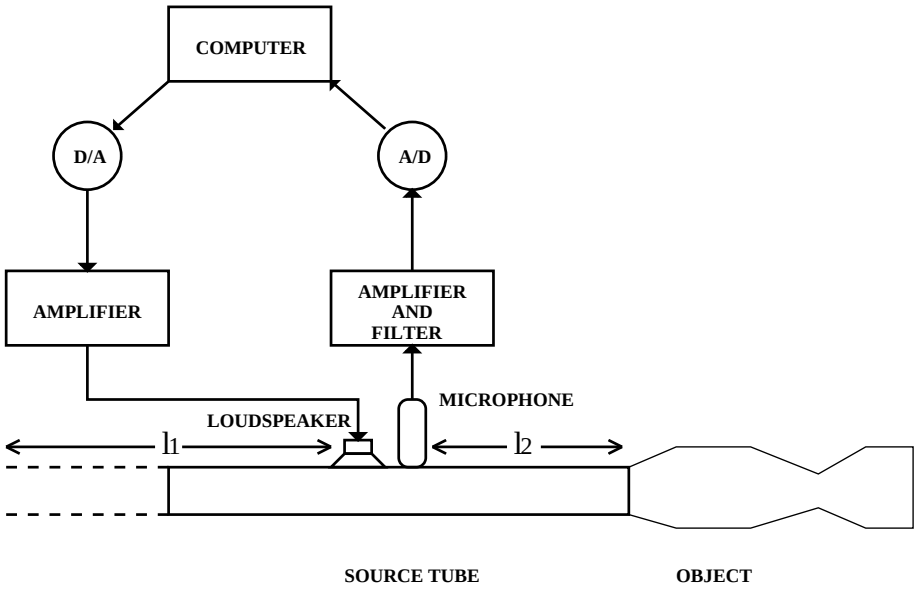

Figure 7.1: *Pulse reflectometer with loudspeaker in source tube wall.*

ways. Some of the pulse would travel along the semi-infinite source tube section l_1 and undergo no further reflection. The rest of the pulse would travel past the microphone and along the short section of source tube l_2 into the object, creating object reflections as before. The object reflections would travel back along the source tube section l_2, be recorded by the microphone and then continue along the semi-infinite section l_1 without further reflection. In this way, the object reflections could be recorded over an indefinite period of time. Again, the input impulse response would be found by

deconvolving the object reflections with the input pulse shape. This idea has never been implemented. To behave as a semi-infinite section of tubing, l_1 would need to be sufficiently long for the object reflections to decay to zero whilst travelling along it. This would make the system very bulky. Another potential difficulty would be matching the acoustic impedance of the loudspeaker to that of the source tube wall. Any difference in impedance would still produce source reflections.

In this chapter, another method which involves preventing the formation of source reflections is proposed and investigated. This method uses the reflectometer of section 3.2 but forces the loudspeaker to behave as an absorbing termination so that after the object reflections pass the microphone they are absorbed by the loudspeaker, instead of being reflected. To achieve this, at the same time as the incoming object reflections are reflected by the loudspeaker diaphragm, the loudspeaker is driven so that it emits a signal of opposite phase. Hence, the source reflections are cancelled out, removing the restriction on the length of time over which the object reflections can be sampled.

7.2 Description of absorbing termination method

The original reflectometer (described in section 3.2) is used in the investigation of the absorbing termination method. Previously object reflections measured using this reflectometer were restricted to a maximum sample length of 1024 points when a sampling frequency of 50kHz was used. Sampling for longer than this resulted in the recording of spurious source reflections. Here the object reflections are sampled at 50kHz and stored in an array of 2048 points. The second half of the sample will be contaminated by source reflections unless they are perfectly cancelled at the loudspeaker. The extent of the cancellation is investigated.

As before, an electrical pulse is produced, amplified and used to drive the loudspeaker. In this case, however, the 5V electrical pulse forms the first 80μs of a wave-

form of 4096 points with the remainder of the waveform set to 0V. This waveform is played out at 50kHz producing a sound pulse which is injected into the object via the source tube. As the waveform is played out, the microphone signal is simultaneously sampled (also at 50kHz) forming a second array of 4096 points. This array of 4096 points contains the input pulse, the object reflections and the source reflections. This procedure is repeated 1000 times and the samples are averaged.

The microphone output is analysed and the time at which the first of the object reflections passes the microphone is determined. (The first object reflection passes the microphone $(l_1 + 2l_2)/c$ seconds after the input pulse is produced by the loudspeaker). For the next $2l_1/c$ seconds, the microphone signal consists solely of reflections from the object. For $l_1 = 3.10$m, this means that for approximately 18ms only object reflections are recorded. At 50kHz, 512 samples last for 10.24ms, therefore the first 512 samples after the start of the object reflections are guaranteed to be source reflection free. These 512 samples are put into an array and convolved with an all-pole model filter representing the losses associated with source tube section l_1 (see section 2.3.3). This gives the shape of the first 10.24ms of object reflections just prior to their further reflection by the loudspeaker.

In order to find the first part of the cancellation signal, it is necessary to find the shape of the first 10.24ms of source reflections at the exit of the loudspeaker. Hence, the 512 point array must be further convolved with a filter representing the relationship between the pressure signal before and after reflection by the loudspeaker; i.e. the input impulse response of the loudspeaker. (The determination of the input impulse response of the loudspeaker is described in section 7.2.1). This gives the shape of the first 10.24ms of source reflections which, when inverted, provides the required cancellation signal.

To find the electrical signal necessary to produce this cancellation signal, the 512 point array is deconvolved with a filter representing the relationship between the elec-

trical input to the loudspeaker and the resultant pressure output. This gives the electrical signal as a rate of change of voltage from which, by integration, the required voltage signal can be calculated. The integration is carried out by performing a 'running sum'; i.e. sample 0 remains the same, sample 1 becomes sample 0 + sample 1, sample 2 becomes sample 0 + sample 1 + sample 2 etc. (The determination of the filter and the reason for using the rate of change of voltage as the input to the filter are discussed in section 7.2.2).

The time taken for a signal to travel down the source tube and back is $2(l_1 + l_2)/c$. Hence, the first object reflections reach the loudspeaker a time $2(l_1 + l_2)/c$ after the production of the input pulse. Thus, the original 4096 point waveform is altered so that the calculated voltage signal is played out $2(l_1 + l_2)/c$ seconds after the 5V electrical pulse.

When the new waveform is played out, a sound pulse is produced which is passed down the source tube and reflected by the object. The object reflections return to the loudspeaker at which point in time the first part of the cancellation signal is emitted, resulting in the absorption of the first 10.24ms of the object reflections; i.e. the cancellation of the first 10.24ms of source reflections. Again, the microphone output is sampled simultaneously and stored in a 4096 point array. The playing out of the modified waveform and simultaneous sampling of the microphone output is repeated 1000 times and the samples are averaged.

The new microphone output is analysed and the second 512 samples after the start of the object reflections are isolated. These are now also guaranteed to be source reflection free. They are processed in the same way as the first 512 samples of the object reflections, resulting in the calculation of a second voltage signal. The 4096 point waveform is further altered so that the second voltage signal is played out straight after the first.

This procedure is continued iteratively until the calculated voltage signal fills the

4096 point output waveform. When this final waveform is played out, a sound pulse is produced which is passed down the source tube and reflected by the object. The object reflections return to the loudspeaker at which point in time the full cancellation signal is emitted, resulting in the absorption of all the object reflections; i.e. the cancellation of all the source reflections. Again, the microphone output is simultaneously sampled and stored in a 4096 point array. The playing out of the final waveform and simultaneous sampling of the microphone output is repeated 1000 times and the samples are averaged.

The microphone output is analysed and the time at which the first of the object reflections passes the microphone is determined. The object reflections are put in an array of 2048 points. Now that the source reflections have been cancelled, the only limit to the size of the array of object reflections comes from the hardware which only allows waveforms of up to 4096 points to be simultaneously played out and sampled. The object reflections are deconvolved with the input pulse shape to give the input impulse response. As before, the input pulse shape is measured by reflection from a rigid termination but in this case the described cancellation technique is used to ensure that no source reflections are recorded.

7.2.1 Determination of the input impulse response of the loudspeaker

To determine the input impulse response of the loudspeaker, the reflectometer source tube was terminated by a perspex plate. A 5V electrical pulse lasting 80μs was produced and used to drive the loudspeaker. Meanwhile, the microphone output was simultaneously sampled at 50kHz, and stored in an 4096 point array. The reflection of the input pulse from the rigid termination and the subsequent source reflection were located, isolated and stored in two 512 point arrays. The reflection from the termination was then convolved with an all-pole model filter representing the losses associated

with source tube section l_1. The source reflection was deconvolved with the same filter. This gave the pressure signals immediately before and immediately after reflection by the loudspeaker. Deconvolving the post-reflection signal by the pre-reflection signal yielded the required loudspeaker input impulse response.

7.2.2 Determination of the filter relating the loudspeaker's electrical input and the resultant pressure output

The electrical input to the loudspeaker and the resultant pressure output are clearly related. To investigate this relationship a simple model of the loudspeaker is used [Nelson and Elliott 1993].

In the model, the loudspeaker diaphragm is represented as a piston of mass M_m which is supported by a suspension with stiffness K_m and damping R_m. Passing a current through a coil which is attached to the diaphragm and held in the magnetic field of a permanent magnet forces the diaphragm to move backwards and forwards in an oscillatory motion. To set the diaphragm in motion the force due to the current must overcome the inertia, stiffness and damping of the suspension. In addition it encounters another force due to the acoustic pressure fluctuations which load the surface.

From this model, an equation of motion for the loudspeaker diaphragm can be written:

$$j\omega M_m u + R_m u + \frac{K_m u}{j\omega} = BIl - pS \tag{7.1}$$

where harmonic motion has been assumed and u is the diaphragm velocity, p is the acoustic pressure which acts over the area S of the diaphragm, I is the current flowing in the coil and Bl is the product of the magnetic flux density and the length of the coil.

For a coil with electrical resistance R and inductance L, when a voltage V is applied:

$$j\omega LI + RI = V - Blu \tag{7.2}$$

where Blu is the back emf produced in the coil by its motion in the magnetic field.

Rearranging equation 7.2 gives an expression for the current,

$$I = \frac{V - Blu}{j\omega L + R} \tag{7.3}$$

which when substituted in to equation 7.1 yields:

$$j\omega M_m u + R_m u + \frac{K_m u}{j\omega} = \frac{BlV}{j\omega L + R} - \frac{(Bl)^2 u}{j\omega L + R} - pS \tag{7.4}$$

Rearranging equation 7.4 gives an expression for the pressure:

$$p = \left[\frac{Bl}{S(j\omega L + R)}\right] V - \left[\frac{j\omega M_m}{S} + \frac{R_m}{S} + \frac{K_m}{j\omega S} + \frac{(Bl)^2}{S(j\omega L + R)}\right] u \tag{7.5}$$

Assuming plane wave propagation, the particle velocity (which is equal to the diaphragm velocity) is related to the pressure by the specific acoustic impedance z, which is the product of the air density and the speed of sound in air.

$$u = \frac{p}{z} = \frac{p}{\rho c} \tag{7.6}$$

Substituting equation 7.6 into equation 7.5 gives:

$$p\left[1 + \frac{j\omega M_m}{\rho c S} + \frac{R_m}{\rho c S} + \frac{K_m}{j\omega \rho c S} + \frac{(Bl)^2}{\rho c S(j\omega L + R)}\right] = \left[\frac{Bl}{S(j\omega L + R)}\right] V \tag{7.7}$$

which can be rearranged to give:

$$p = \left[\frac{Bl\rho c}{(j\omega L + R)\left(\rho c S + j\omega M_m + R_m + \frac{K_m}{j\omega} + \frac{(Bl)^2}{(j\omega L + R)}\right)}\right] V \tag{7.8}$$

The Fane Professional MD2050 compression driver loudspeaker used in the reflectometer has a resistance $R = 6.2\Omega$, an inductance $L = 21\mu$H, a magnetic flux density $B = 1.74$T, a voice coil of length $l = 3.5$m, a diaphragm mass $M_m = 2.8$g, a damping

constant $R_m = 9.7$Ns/m and a suspension stiffness $K_m = 3.8 \times 10^6$N/m (the large stiffness value is a result of inserting a thin layer of cotton wool between the diaphragm and the magnet assembly to reduce 'ringing' - see section 3.2). The standard techniques used to measure these characteristics are described by Hall [1987] and are summarised in appendix A.

Using these values, equation 7.8 can be greatly simplified. The ratio between the resistance and the inductance of the loudspeaker is $R/L = 294762$ rad/s, implying that the loudspeaker behaves mainly resistively below ω=294762 rad/s; i.e. below f=46.9kHz. The frequency spectrum of the electrical input to the loudspeaker contains no significant energy at or above this frequency so the loudspeaker can be treated as purely resistive. Equation 7.8 therefore reduces to:

$$p = \left[\frac{Bl\rho c}{R\left(\rho cS + j\omega M_m + R_m + \frac{K_m}{j\omega} + \frac{(Bl)^2}{R} \right)} \right] V \tag{7.9}$$

At all frequencies $\omega M_m + K_m/\omega >> \rho cS + R_m + (Bl)^2/R$, so equation 7.9 can be further simplified to:

$$\begin{aligned} p &= \left[\frac{Bl\rho c}{R\left(\frac{K_m}{j\omega} + j\omega M_m \right)} \right] V = \left[\frac{Bl\rho c}{R\left(\frac{K_m}{j\omega} - \frac{\omega^2 M_m}{j\omega} \right)} \right] V \\ &= \left[\frac{Bl\rho c}{R(K_m - \omega^2 M_m)} \right] j\omega V = \left[\frac{Bl\rho c}{R(K_m - \omega^2 M_m)} \right] \frac{dV}{dt} \end{aligned} \tag{7.10}$$

So at all frequencies below f=46.9kHz, the pressure is dependent on the rate of change of voltage rather than on the voltage itself. Hence, when determining the filter relating the electrical input to the loudspeaker to the pressure output, it is most appropriate to use the rate of change of voltage as the input to the filter.

To determine the filter, the reflectometer source tube was again terminated by a perspex plate. This time a 0.25V electrical pulse lasting 80μs was produced and used to

drive the loudspeaker. (A 0.25V electrical pulse was chosen because 0.25V is comparable to the voltages required to cancel the source reflections. At these lower voltages the loudspeaker behaves linearly; i.e. it obeys the derived relationship between the rate of change of voltage and the output pressure. When a 5V electrical pulse was used, the loudspeaker behaved non-linearly resulting in a filter which described the relationship between electrical input and pressure output incorrectly at the low voltages required for source reflection cancellation). The microphone output was again simultaneously sampled at 50kHz and stored in a 4096 point array. The reflection of the pulse from the rigid termination was located, isolated and stored in a 512 point array. The array was then deconvolved with an all-pole model filter representing the losses associated with a length $l_1 + 2l_2 = 9.28$m of source tube to find the shape of the pulse immediately after production by the loudspeaker. The rate of change of voltage of the electrical pulse was calculated by differentiating the voltage of the pulse. To carry out the differentiation, the electrical pulse was placed in a 512 point array and then transformed so that sample 0 remained the same, sample 1 became sample 1 - sample 0, sample 2 became sample 2 - sample 1 etc. Deconvolving the pressure signal by the rate of change of voltage of the electrical pulse gave the required filter.

7.3 Cancellation of source reflections

To investigate the viability of the absorbing termination method as a means of measuring the input impulse responses of longer objects, it was first necessary to see how effectively source reflections were cancelled using the method. All the measurements were made at a temperature of $\tau = 21.5^{o}$C.

7.3.1 Perspex plate

The first object tested was a perspex plate, chosen because of the simplicity of the object reflections (i.e. the reflection of the input pulse) and because they completely passed the microphone before the source reflections reached it. Thus, the object and source reflections were separated in time enabling the cancellation of the source reflections to be clearly observed.

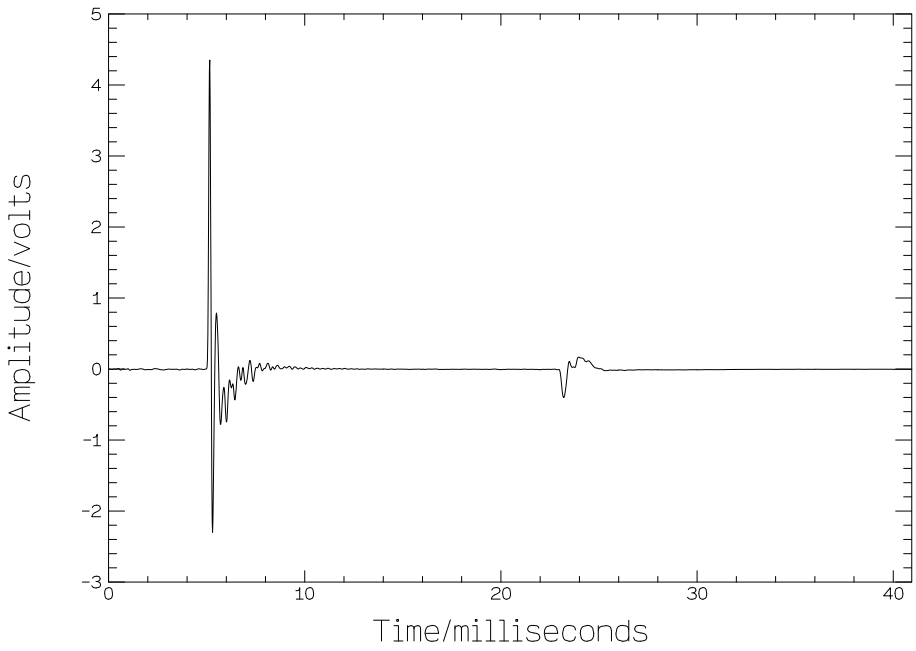

Figure 7.2: *Reflections from perspex plate. No cancellation to remove the source reflections.*

Figure 7.2 shows the signal recorded by the microphone before the loudspeaker was forced to act as an absorbing termination. The graph starts 5ms before the passage of the perspex plate reflections. The source reflections pass the microphone 18ms after the perspex plate reflections.

Figure 7.3 shows the signal recorded by the microphone after the loudspeaker was forced to act as an absorbing termination. The source reflections are now reduced in amplitude due to cancellation at the loudspeaker.

The reduction in amplitude of the source reflections can be seen more clearly in

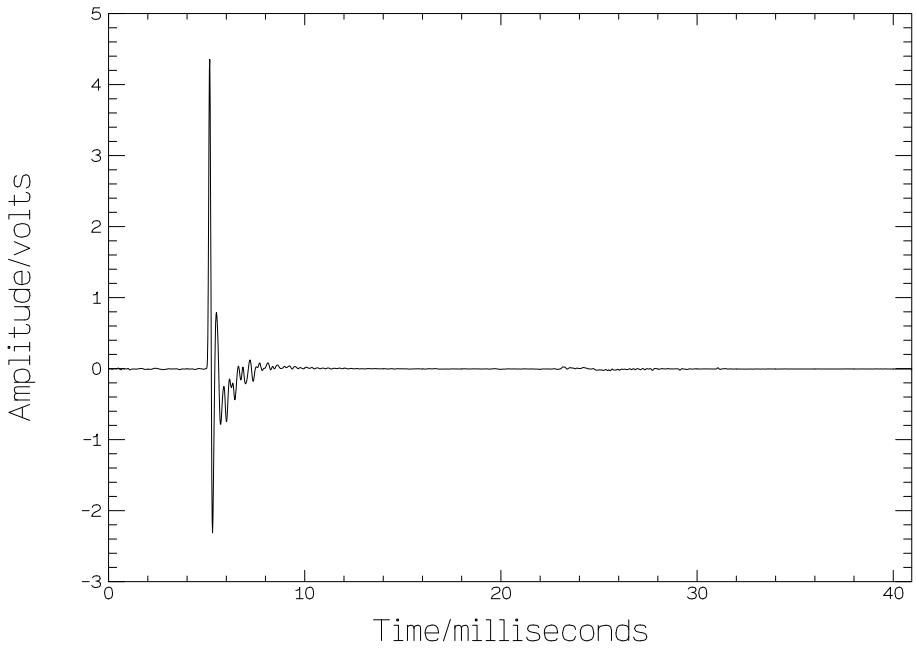

Figure 7.3: *Reflections from perspex plate. Cancellation to remove the source reflections.*

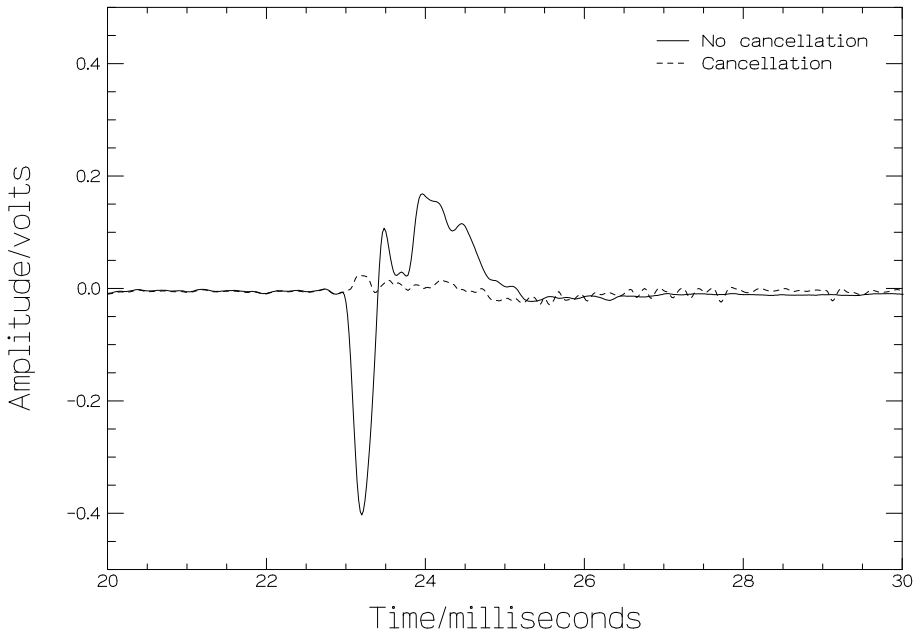

Figure 7.4: *Comparison of source reflections before and after cancellation.*

figure 7.4. The absorbing termination method has succeeded in reducing the source reflections to approximately 10% of their original peak-to-peak amplitude.

7.3.2 Stepped tube

The next object tested consisted of the stepped tube and the coupler described in section 3.5 (with an additional 403mm long cylindrical connector inserted between source tube and coupler for evaluation of DC offset). The stepped tube was chosen because the object reflections, although more complicated than those from the perspex plate, still completely passed the microphone before the first source reflections reached it. Hence, the cancellation of the source reflections could again be clearly observed.

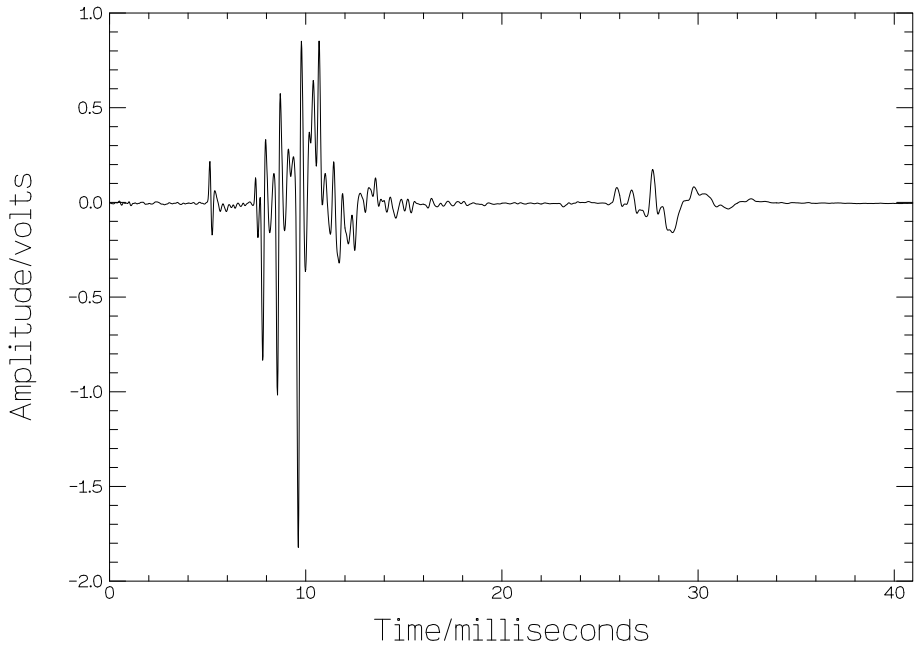

Figure 7.5: *Reflections from stepped tube and coupler with cylindrical connector. No cancellation to remove the source reflections.*

Figure 7.5 shows the signal recorded by the microphone before the loudspeaker was forced to act as an absorbing termination. Again, the graph starts 5ms before the passage of the stepped tube reflections. The source reflections are recorded by the microphone 18ms after the first of the stepped tube reflections passes the microphone.

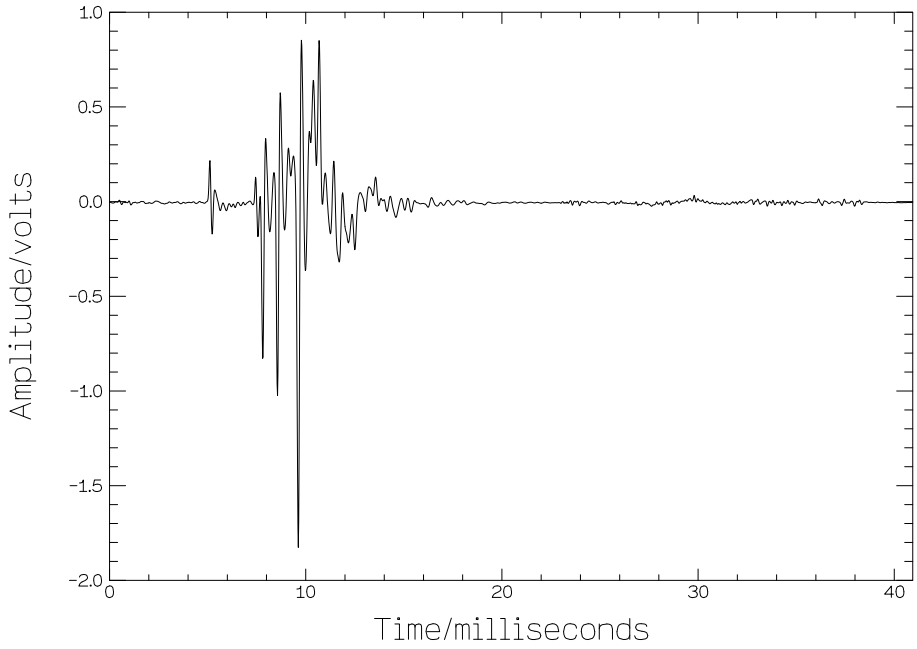

Figure 7.6: *Reflections from stepped tube and coupler with cylindrical connector. Cancellation to remove the source reflections.*

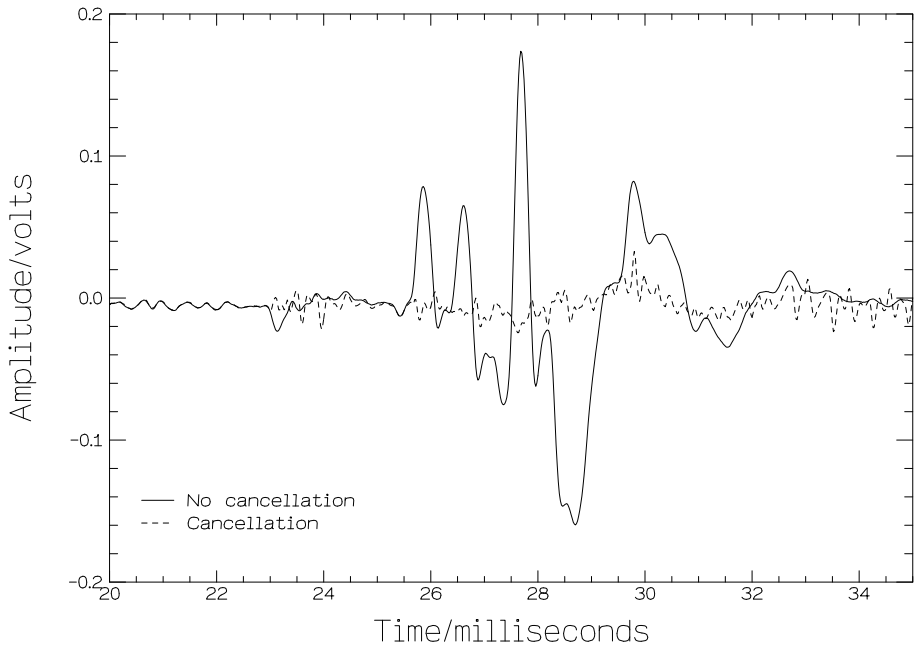

Figure 7.7: *Comparison of source reflections before and after cancellation.*

Figure 7.6 shows the signal recorded by the microphone after the loudspeaker was forced to act as an absorbing termination. The source reflections are now reduced in amplitude due to cancellation at the loudspeaker.

The extent of the cancellation of the source reflections can be seen in figure 7.7. Despite the more complicated shape of the source reflections, the absorbing termination method has again succeeded in reducing them to approximately 10% of their original peak-to-peak amplitude.

7.3.3 4.19m long stepped tube

The final object tested consisted of a 4.19m long stepped tube and the coupler shown in figure 3.5 (with an additional 403mm long cylindrical connector inserted between source tube and coupler for evaluation of DC offset). This stepped tube was made up of four cylindrical sections; a 646mm long section of 6.2mm radius, a 3290mm section of 5mm radius, an 80mm section of 6.2mm radius and a 177mm section of 9.25mm radius. These values were measured using calipers and a ruler, leading to a reading error of ± 0.05mm on each quoted radius and a reading error of ± 0.5mm on each quoted length.

The 4.19m long stepped tube was chosen because the object and source reflections were not separated in time, with the first source reflection reaching the microphone before the final primary reflection from the object.

Figure 7.8 shows the signal recorded by the microphone before the loudspeaker was forced to act as an absorbing termination. The graph starts 5ms before the passage of the first reflections from the 4.19m stepped tube. The first source reflections reach the microphone 18ms after the first of the stepped tube reflections and 13ms before the final primary stepped tube reflection.

Figure 7.9 shows the signal recorded by the microphone after the loudspeaker was forced to act as an absorbing termination. The source reflections have been cancelled

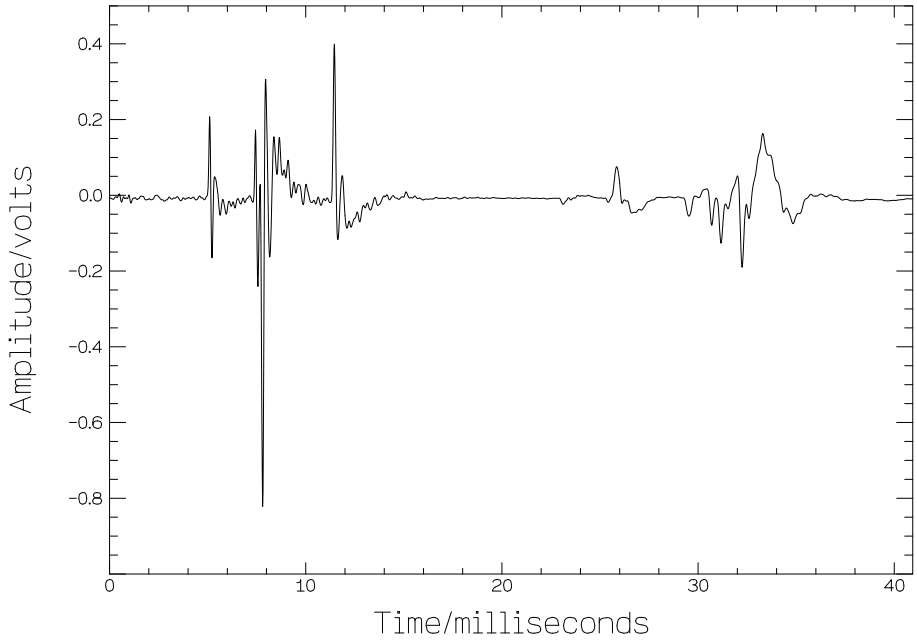

Figure 7.8: *Reflections from 4.19m stepped tube and coupler with cylindrical connector. No cancellation to remove the source reflections.*

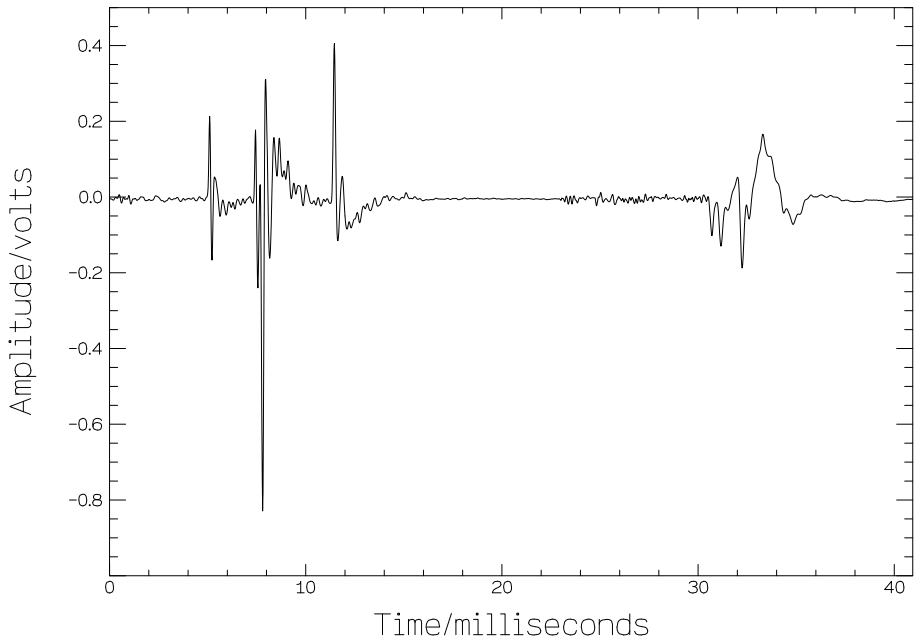

Figure 7.9: *Reflections from 4.19m stepped tube and coupler with cylindrical connector. Cancellation to remove the source reflections.*

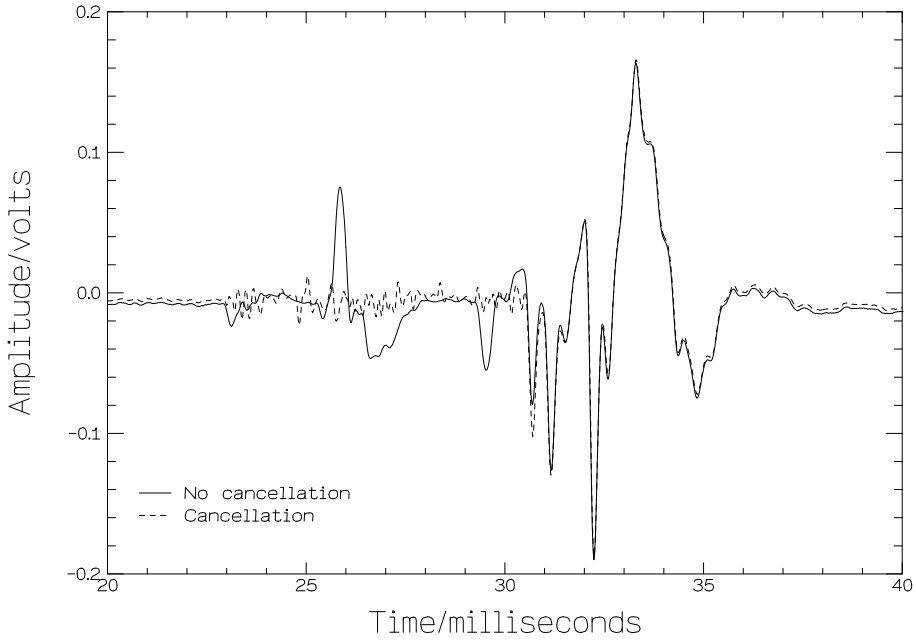

Figure 7.10: *Comparison of 4.19m stepped tube reflections before and after cancellation.*

but the object reflections have been left unaltered.

The cancellation of the source reflections and non-cancellation of the object reflections can be seen more clearly in figure 7.10.

7.4 Input impulse response and resultant bore reconstruction of 4.19m long stepped tube

The motivation for the development of the absorbing termination method was the measurement of the input impulse responses of longer objects (from which bore reconstructions could be evaluated). In the previous section, it was shown that the method could successfully reduce source reflections to approximately 10% of their original peak-to-peak amplitude. In this section, the effect of the reduction in source reflection amplitude on the input impulse response and bore reconstruction of the 4.19m long stepped tube is examined.

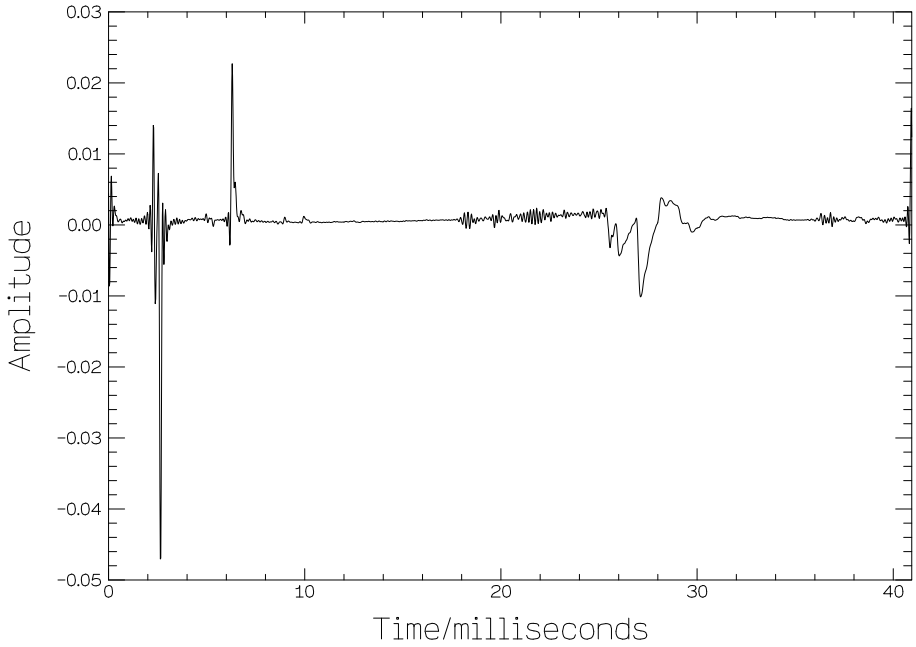

Figure 7.11: *Input impulse response of 4.19m stepped tube and coupler with cylindrical connector after cancellation.*

Figure 7.11 shows the input impulse response of the 4.19m long stepped tube and coupler (with cylindrical connector in place) obtained by deconvolving the stepped tube reflections of figure 7.9 with the reflected input pulse of figure 7.3. Due to the imperfect nature of the source reflection cancellation, the input impulse response becomes a little noisier at 18ms when the cancellation signal is played out.

Figure 7.12 shows the 4.19m long stepped tube bore reconstruction resulting from the application of the lossy layer-peeling algorithm to the input impulse response of figure 7.11. Before the algorithm was applied the DC offset in the response was evaluated by averaging over the 1ms-2ms range. The offset was removed and the first millisecond of the input impulse response was replaced by zero to improve the accuracy of the reconstruction (see section 4.2.2).

The reconstructed profile is in good agreement with the measured profile over the first 1.1m of the reconstruction (i.e. over the cylindrical connector used for DC offset evaluation and over the first section of the stepped tube). However, after this the

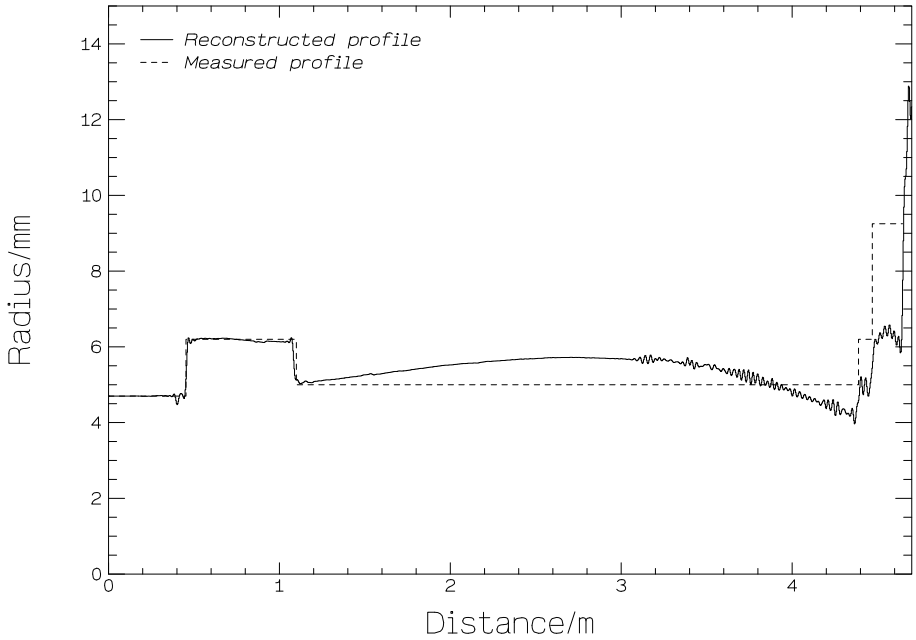

Figure 7.12: *Comparison of reconstructed and measured profiles of 4.19m stepped tube.*

reconstructed profile departs quite dramatically from the expected shape. The second cylindrical section, from 1.1m to 4.39m, has a measured radius of 5mm but the reconstruction expands to a radius of 5.9mm and then contracts to a radius of 4mm. The final two cylindrical sections, from 4.39m to 4.65m, are also reconstructed poorly with both radii underpredicted.

The expansion and contraction in the reconstruction of the second cylindrical section suggests that, after the DC offset had been removed, the input impulse response still contained a low amplitude component which varied slowly with time. Close examination of figure 7.11 over the 7ms-25ms region reveals a small variation in the amplitude of the input impulse response. The likely source of this time-varying offset is the hardware. Previously, an input pulse was played out and then no further signal was sent to the loudspeaker. The resultant reflections were recorded some time later. Now, once the input pulse has been played out, the D/A converter continues to play out a signal to the loudspeaker. It appears that this introduces a component into the

sampled microphone signal which becomes magnified upon deconvolution to find the input impulse response.

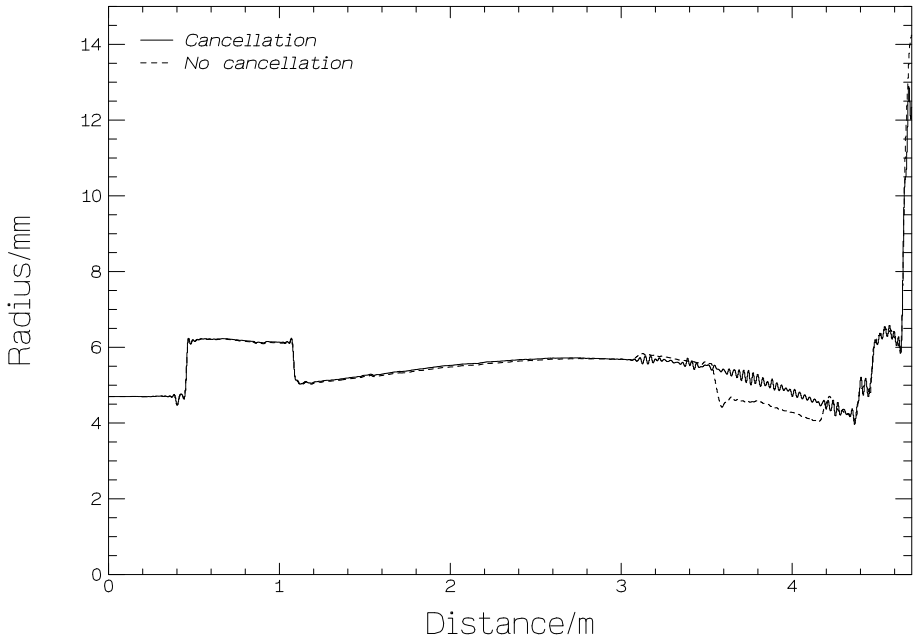

Figure 7.13: *Comparison of reconstructed profiles of 4.19m stepped tube before and after cancellation.*

Despite the poor reconstruction, the potential benefits of the absorbing termination method can be seen in figure 7.13. Here the reconstruction of figure 7.12 is compared with the reconstruction calculated from the input impulse response obtained when the uncancelled reflections of figure 7.8 are deconvolved with the input pulse of figure 7.3. It is clear that the source reflections cause both the spurious narrowing of the reconstruction at 3.5m and the spurious expansion of the reconstruction at 4.2m.

7.5 Discussion

The results presented show the promise of the absorbing termination method of measuring longer instruments. Cancellation of source reflections of varying shapes proved very successful but the input impulse response and bore profile measurements were

less impressive. A small undesirable time-varying component introduced into the input impulse response of the 4.19m long stepped tube caused the reconstructed profile to differ from the measured profile. A hardware upgrade appears necessary to attempt to eliminate this component from the sampled signal. If the removal of the component can be achieved, the absorbing termination method should prove successful in measuring the input impulse responses and bore profiles of longer instruments. If not, the source tube wall loudspeaker technique and the two microphone technique appear to be the best alternatives for the accurate measurement of longer objects (with the two microphone technique having the advantage of only requiring a short source tube).

Chapter 8

Summary and conclusions

8.1 Achievement of aims

The initial research aims set out in section 1.2 have all been achieved. In this chapter, each aim is restated and the extent to which the aim was fulfilled is discussed.

8.1.1 Aim 1

The first aim was to produce a working reflectometer to accurately measure the input impulse response, bore profile and input impedance of a musical wind instrument or other tubular object without prior knowledge of its dimensions.

A reflectometer was built which enabled the input impulse response of a tubular object to be experimentally determined. However, input impulse responses measured using the reflectometer were all found to contain both ripple and a DC offset. The introduction of ripple was a consequence of the finite sampling frequency and could not be prevented or removed. The introduction of the DC offset also could not be prevented but it could be removed computationally.

For accurate bore reconstruction, the removal of the DC offset in the input impulse response was shown to be vital. Application of a reconstruction algorithm to an input

impulse response whose DC offset had not been removed resulted in a bore profile which expanded or contracted spuriously. In the past, the DC offset was determined by repeatedly applying the reconstruction algorithm and adjusting the DC value removed until the bore profile coincided with a directly measured radius at an arbitrary point towards the end of the instrument. A new method was introduced which involved inserting a cylindrical connector between the reflectometer and object. The DC offset was then evaluated directly from the input impulse response. This method was much quicker and did not require prior knowledge of the object's dimensions.

The input impulse response of a stepped tube was measured and its DC offset removed using the new method. Various reconstruction algorithms were applied and the resultant bore profiles were compared. The lossy layer-peeling algorithm was found to be the most accurate, giving a bore profile which agreed with directly measured radii to within 0.05mm. Reconstructions of various brass instruments were found to have a similar accuracy.

The input impulse response of the stepped tube was also used to calculate its input impedance. It was shown that the resolution of the impedance curve could be improved by zero-padding the input impulse response. The peak frequencies of the stepped tube impedance curve were found to be approximately 0.7% lower than those of a theoretically calculated curve. This agreement was better than the agreement between the theoretical curve and a stepped tube impedance curve measured using a standard frequency domain technique. The peak frequencies of the impedance curve measured using the frequency domain technique were 2% lower than those of the theoretical curve. The impedance curves of various brass instruments (calculated from their input impulse responses) were presented and information about their playing characteristics was deduced.

8.1.2 Aim 2

The second aim was to apply acoustic pulse reflectometry to the problem of detecting leaks in musical wind instruments and in tubular objects in general.

It was found that when a leaking object was measured using the reflectometer, the reconstruction was only correct as far as the position of the leak (after which it expanded to an extent dependent on the size of the leak). This led to a method of detecting the presence of leaks in a tubular object. If the bore reconstruction of the object passed through a directly measured radius at a known point towards the end of the object, the object could be considered airtight. If, however, the reconstruction had a larger radius than the directly measured radius, the object contained a leak. The method was successfully demonstrated on two leaking brass instruments.

In certain cases, it also proved possible to locate the leak position simply by turning the object around and making a second measurement using the reflectometer. Comparing the two reconstructions allowed the position of the leak to be identified. In this way, two measurements made on a flute enabled a leaking pad to be located.

For the special case of a leaking cylinder, it was shown that the size of the hole could be found using reflectometry measurements. Given the cylinder dimensions and the leak position, the application of a theoretically derived expression to the input impedance of the cylinder yielded a prediction of the hole size. For three different hole sizes, the predicted radii were found to agree with directly measured radii to within experimental error (which depended on the hole size).

8.1.3 Aim 3

The third aim was to investigate possible methods of measuring longer tubular objects using acoustic pulse reflectometry.

The constraint on the length of object which could be measured using a practical reflectometer was discussed and various methods for measuring longer objects were

described. A new method was presented which involved driving the loudspeaker in such a way as to absorb the incoming reflections from the object rather than reflecting them back to the microphone. This method stopped the formation of source reflections, allowing the object reflections to be accurately recorded over an indefinite period of time. An investigation revealed that the method reduced source reflections to 10% of their original amplitude but left object reflections unchanged. This should have enabled longer objects to be more accurately reconstructed. However, when a longer object was measured using this method, the introduction of a slowly varying low amplitude component into the recorded signal caused the bore reconstruction to first expand spuriously and then contract spuriously.

8.2 Further work

Further work is necessary to produce a reflectometer capable of accurately measuring long objects. An improvement in the electronics might enable the absorbing termination method to be used. If not, positioning the loudspeaker in the source tube wall appears worthy of investigation. Like the absorbing termination method, this method would prevent the formation of source reflections but it has the disadvantage that the reflectometer required would be very bulky. The use of two microphones in the source tube wall may be a more promising approach as it has the benefit of allowing a more portable reflectometer to be constructed.

As computational power increases, it might prove possible to produce a reflectometer which accurately calculates bore reconstructions and input impedance curves in real-time. This would allow, for example, the effect of depressing a trumpet valve on the trumpet's bore profile and input impedance to be observed. At present, a pseudo real-time reflectometer could be implemented by storing sets of reflections (sampled without averaging) and applying the deconvolution, reconstruction and input impedance algorithms to them at a later time.

Input impulse response measurements made on musical instruments should prove useful in the field of instrument modelling. At present, lip models are used in conjunction with theoretically calculated input impulse responses to model the characteristics of cup mouthpiece instruments. Although a great deal of work has gone into improving the theoretical input impulse response calculations [Agullo et al 1995], it would be interesting to examine the effect on instrument models of using experimentally measured input impulse responses. Recent papers by Valimaki et al [1995] and Keefe [1996] report early work in this area.

Appendix A

Measuring the loudspeaker characteristics

The electrical and mechanical characteristics of the Fane Professional MD2050 compression driver loudspeaker were measured using standard techniques which are described by Hall [1987].

A.1 Theory

To understand these techniques, it is necessary to re-examine the theoretical expressions deduced from the loudspeaker model (section 7.2.2).

The first theoretical expression, equation 7.1, can be rewritten by replacing the pressure by the radiation impedance (Z_{rad}) multiplied by the volume velocity ($U = Su$). In fact, the radiation reactance X_{rad} dominates over most of the frequency range of interest, so:

$$j\omega M_m u + R_m u + \frac{K_m u}{j\omega} = BIl - X_{rad} S^2 u \tag{A.1}$$

The second theoretical expression, equation 7.2, can also be rewritten giving an expression for the effective electrical impedance of the loudspeaker. Dividing through

by I and rearranging gives:

$$Z_e = R + j\omega L + Blu/I \tag{A.2}$$

The impedance is made up of three terms; the loudspeaker's electrical resistance, the loudspeaker's electrical inductance and a term resulting from the back emf. This third term (often referred to as the motional impedance) represents the load imposed upon the circuit by the mechanical system that is being driven.

A.2 Electrical and mechanical characteristics

A.2.1 Electrical resistance of loudspeaker R

The electrical resistance of the loudspeaker ($R = 6.2\Omega$) was found by applying an ohmmeter across the terminals of the loudspeaker and making a DC measurement. The DC measurement ensured that both $j\omega L$ and Blu/I were zero and the measured impedance was simply the electrical resistance of the loudspeaker (see equation A.2).

A.2.2 Electrical inductance of loudspeaker L

To find the electrical inductance of the loudspeaker, a 1Ω resistor was connected in series with the loudspeaker and an alternating voltage was applied across both. The AC voltages across each individual component were measured, enabling the electrical impedance of the loudspeaker to be calculated. The electrical impedance was simply V_{sp}/V_{res}, where V_{sp} was the voltage across the loudspeaker and V_{res} was the voltage across the resistor (the 1Ω resistor ensured that the voltage across the resistor was equal to the current flowing through it). The magnitude of the impedance was measured over a range of frequencies, resulting in a curve which changed from a value of R at low frequencies to a value of ωL at high frequencies (in accordance with equation A.2). The electrical inductance of the loudspeaker ($L = 21\mu$H) was found by applying a high

frequency AC voltage and dividing the measured impedance by the angular frequency; i.e. $L = V_{sp}/(V_{res} \times \omega)$.

A.2.3 Magnetic flux density B, length of voice coil l, damping constant R_m

The electrical impedance curve measured in the previous section did not change smoothly from the low frequency value of R to the high frequency value of ωL but passed through a peak. This peak, which occured at the mechanical resonance frequency f_0, was caused by the motional impedance. At mechanical resonance, the diaphragm of the loudspeaker moves at a greater velocity and the motional impedance becomes large.

At mechanical resonance, the mechanical reactance vanishes and equation A.1 reduces to $R_m u = BIl$. Since f_0 is generally low enough for L to be unimportant, equation A.2 can be rewritten:

$$Z_e = R + (Bl)^2/R_m \tag{A.3}$$

This electrical impedance is purely resistive. Hence, the resonance frequency f_0 can be accurately determined by adjusting the frequency until V_{sp} and V_{res} have a phase difference of zero. Given the loudspeaker resistance R, measuring the electrical impedance at this frequency enables calculation of $(Bl)^2/R_m$. The loudspeaker was found to have a resonance frequency of $f_0 = 5.88$kHz at which the electrical impedance was 10.0Ω. Thus, $(Bl)^2/R_m$ was found to equal 3.81Ω.

The magnetic flux density ($B = 1.74$T) was given in the literature produced by Fane and the length of the voice coil ($l = 3.5$m) was found by multiplying the circumference of the coil ($\pi\times$ diameter of coil = $\pi \times 0.045$m = 0.14m) by the number of turns counted in the coil (25). Using the value of 3.81Ω for $(Bl)^2/R_m$, the damping constant $R_m = 9.7$ Ns/m was calculated.

A.2.4 Suspension stiffness K_m and diaphragm mass M_m

The resonance frequency $f_0 = 5.88$kHz of the loudspeaker enabled the ratio of the suspension stiffness K_m to the diaphragm mass M_m to be calculated:

$$\frac{K_m}{M_m} = (2\pi f_0)^2 = 1.37 \times 10^9 \text{s}^{-2} \tag{A.4}$$

The large ratio is due to the increased stiffness of the loudspeaker caused by inserting a thin layer of cotton wool between the diaphragm and the magnet assembly.

To find the mass M_m of the diaphragm, the cotton wool was removed. The resonance frequency of the loudspeaker without cotton wool inserted was measured as being $f_0' = (1/2\pi)\sqrt{K_m'/M_m} = 1.15$kHz, where K_m' is the suspension stiffness with no cotton wool inserted. Adding a mass m=0.1g to the diaphragm of the speaker lowered the resonance frequency to $f_1' = (1/2\pi)\sqrt{K_m'/(M_m+m)} = 1.13$kHz. Substituting $K_m' = M_m(2\pi f_0')^2$ and rearranging gave:

$$M_m = m\frac{(f_1')^2}{(f_0')^2 - (f_1')^2} = 0.1\text{g} \times \frac{1130^2}{1150^2 - 1130^2} = 2.8\text{g} \tag{A.5}$$

The stiffness of the loudspeaker with cotton wool inserted ($K_m = 3.8 \times 10^6$N/m) was then calculated by substituting the value of M_m=2.8g into equation A.4 and rearranging.

Appendix B

Computer programs

The programs listed were all written using Borland Turbo C.

B.1 Data acquisition program

```
/* PULSE REFLECTOMETRY DATA ACQUISITION PROGRAM */
/* (produces pulses, samples reflections and averages) */
/* The D/A converter sends out voltages in the -5V to +5V range */
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* The A/D converter samples voltages in the -5V to +5V range.*/
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* Include library files */
#include <stdio.h>
#include <c:\daqbook\dos\bc\daqbook.h>
/* Declare functions */
void main(void);
/* Main program */
void main()
{
int i, j, x, gainset, del, numofav, sampsize;
unsigned int buf[8192];
float avbuf[8192], sampfreq;
char filename[80];
FILE *fout;
/* Initialise DaqBoard/100 */
daqInit(PORT_0300,10);
daqAdcSetTag(0);
printf("\n\nPULSE REFLECTOMETRY DATA ACQUISITION PROGRAM\n");
printf("(produces pulses, samples reflections and averages)\n\n");
printf("How many averages do you wish to have?\n");
scanf("%d", &numofav);
printf("What is the sample size?\n");
scanf("%d", &sampsize);
printf("What is the sampling frequency (in kHz)?\n");
scanf("%f", &sampfreq);
printf("Enter the gain setting you require:\n");
printf("(1) DgainX1 (i.e. 0-5V range)\n");
printf("(2) DgainX2 (i.e. 0-2.5V range)\n");
printf("(4) DgainX4 (i.e. 0-1.25V range)\n");
printf("(8) DgainX8 (i.e. 0-0.625V range)\n");
scanf("%d", &gainset);
/* Defaults to DgainX1 if invalid option is selcted */
if (gainset!=1 && gainset!=2 && gainset!=4 && gainset!=8){
printf("This was not an option - gain has been set to DgainX1\n");
gainset=1;
}
printf("How long after the pulse is sent out do you wish to\n");
printf("start sampling (recommended delay for shorter source\n");
printf("tube reflectometer is 24ms and recommended delay for longer\n");
printf("source tube reflectometer is 38ms)?\n");
scanf("%d", &del);
printf("What do you wish to call the output filename?\n");
scanf("%s", filename);
fout=fopen(filename,"w");
/* Set all values of array avbuf equal to zero */
for(x=0;x<sampsize;x++){
```

```
avbuf[x]=0.0;
}
/* Set initial output voltage to zero */
daqDacWt(0,2048);
/* Include delay before sending pulse to allow signal to die away */
delay(1000);
/* Send requested number of pulses and sample after each one */
for(i=0;i<numofav;i++){
/* Switch off computer interrupts */
asm{CLI}
/* Send out 5V voltage pulse */
for (j=0; j<20; j++){
daqDacWt(0,4095);
}
daqDacWt(0,2048);
/* Delay before starting sampling */
delay(del);
/* Sample microphone signal at correct gain */
if (gainset==1){
daqAdcRdN(0,buf,sampsize,DtsPacerClock,0,0,sampfreq,DgainX1);
}
else if (gainset==2){
daqAdcRdN(0,buf,sampsize,DtsPacerClock,0,0,sampfreq,DgainX2);
}
else if (gainset==4){
daqAdcRdN(0,buf,sampsize,DtsPacerClock,0,0,sampfreq,DgainX4);
}
else {
daqAdcRdN(0,buf,sampsize,DtsPacerClock,0,0,sampfreq,DgainX8);
}
/* Add array of samples to array avbuf */
for(x=0;x<sampsize;x++){
avbuf[x]=avbuf[x]+buf[x];
}
/* Switch computer interrupts back on */
asm{STI}
/* Delay to ensure that all signal dies away before next pulse */
/* is sent out */
delay(180);
}
/* Divide summed array of samples by number of averages */
/* and print to file the voltage (converted from samples) */
/* and the time at which it was recorded in milliseconds */
for(x=0;x<sampsize;x++){
avbuf[x]=avbuf[x]/numofav;
fprintf(fout, "%f %f\n",
x*1000.0/sampfreq, -(avbuf[x]-2048.0)*10.0/(4096.0*gainset));
}
daqClose();
}
```

B.2 Data analysis program

```
/* PULSE REFLECTOMETRY DATA ANALYSIS PROGRAM */
/* Include library files */
#include <stdio.h>
#include <math.h>
#include <alloc.h>
#include <conio.h>
#include <stdlib.h>
#define SWAP(a,b) tempr=(a); (a)=(b); (b)=tempr
#define NAMESIZE 80
/* Declare functions */
int print_menu (void);
void main (void), four_tran (void), deconvolve (void), finddc (void),
removedc (void), repl1ms (void), ware_aki (void), noam (int menu_num),
zeropad (void), input_imped (void), farewell (void);
void dfour1(double huge *data, unsigned long nn, int isign);
void myfft(double huge *real, double huge *imag,
unsigned long samp_num, int isign);
/* Main program */
void main (void)
{
int menu_num;
do {
menu_num=print_menu();
```

```
switch (menu_num)
{
case 1 : {four_tran(); break;}
case 2 : {deconvolve(); break;}
case 3 : {finddc(); break;}
case 4 : {removedc(); break;}
case 5 : {repl1ms(); break;}
case 6 : {ware_aki(); break;}
case 7 : {noam(menu_num); break;}
case 8 : {noam(menu_num); break;}
case 9 : {noam(menu_num); break;}
case 10 : {zeropad(); break;}
case 11 : {input_imped(); break;}
case 12 : {farewell(); break;}
default: {printf("This is not a valid choice\n");}
}
} while (menu_num!=12);
}
/****************************** Functions *************************/
/************************* Print menu function ******************/
int print_menu(void)
{
int menu_num;
printf("\nPULSE REFLECTOMETRY PROGRAMS\n"
"(1) Fourier transform data\n"
"(2) Deconvolve two data sets\n"
"(3) Find DC offset\n"
"(4) Remove DC offset\n"
"(5) Replace first millisecond of iir\n"
"(6) Ware-Aki reconstruction algorithm\n"
"(7) Noam Amir lossless reconstruction algorithm\n"
"(8) Noam Amir lossy reconstruction algorithm (rotating phase)\n"
"(9) Noam Amir lossy reconstruction algorithm (all pole model)\n"
"(10) Zero pad iir\n"
"(11) Input Impedance\n"
"(12) Quit\n");
printf("Enter the number you require\n");
scanf("%d", &menu_num);
return(menu_num);
}
/*********************************************************************/
/************************* four_tran ****************************/
void four_tran (void)
{
FILE *fin, *fout1, *fout2;
char inputdata[NAMESIZE], fft_comp[NAMESIZE], fft_mag[NAMESIZE];
int samp_num=0, i;
double time, samp_space, max_mag=0, mag;
/* Set up arrays using pointers */
double huge *real, *imag;
if ((real=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((imag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
printf("\nFOURIER TRANSFORM PROGRAM\n");
/* Read in filename and open file containing data to be FFTed */
printf("What is the name of the input file which is to be FFTed?\n");
scanf("%s", inputdata);
if ((fin=fopen(inputdata, "r"))==0){
printf("Unable to open\n");
return;
}
/* Read real values from input file into an array, */
/* make imaginary values=0 and find sample size (samp_num) */
while (fscanf(fin, "%lf %le", &time, &real[samp_num])==2){
imag[samp_num]=0;
if (samp_num==1){
samp_space=time;
}
samp_num++;
}
/* Compute transform */
myfft(real, imag, samp_num, -1);
/* Read in filenames and open files for */
```

```
/* FFT complex and magnitude data */
printf("What do you wish to call the output file containing"
"\nthe fft complex data?\n");
scanf("%s", fft_comp);
printf("What do you wish to call the output file containing"
"\nthe fft magnitude data?\n");
scanf("%s", fft_mag);
fout1=fopen(fft_comp, "w");
fout2=fopen(fft_mag, "w");
/* Calculate maximum magnitude */
for (i=1; i<samp_num; i++){
if (sqrt(pow(real[i],2)+pow(imag[i],2))>max_mag){
max_mag=sqrt(pow(real[i],2)+pow(imag[i],2));
}
}
/* Write fft data to output files */
for (i=0; i<samp_num; i++){
/* Complex file */
fprintf(fout1, " % 17.14le  % 17.14le\n", real[i], imag[i]);
/* Magnitude file */
/* Ensure not taking log10 of zero */
if (sqrt(pow(real[i],2)+pow(imag[i],2))==0){
real[i]=real[i]+0.000001;
}
mag=20*log10(sqrt(pow(real[i],2)+pow(imag[i],2))/max_mag);
fprintf(fout2, " %13.6lf % 17.14lf\n",
(1000.0*i)/(samp_num*samp_space), mag);
}
/* Free pointers */
farfree(real);
farfree(imag);
/* Close files */
fclose(fin);
fclose(fout1);
fclose(fout2);
}
/*********************************************************************/
/*************************** deconvolve **************************/
void deconvolve (void)
{
FILE *fin1, *fin2, *fout;
int samp_num=0, i;
char comp1[NAMESIZE], comp2[NAMESIZE], deconvdata[NAMESIZE];
double real1, imag1, real2, imag2, samp_space, constrain_factor;
/* Set up arrays using pointers */
double huge *real, *imag;
if ((real=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((imag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
printf("\nDECONVOLUTION PROGRAM\n");
/* Read in filenames and open files containing */
/* data to be deconvolved */
printf("What is the file containing your FFTed,"
"\ncomplex input pulse called?\n");
scanf("%s", comp1);
printf("What is the file containing your FFTed,"
"\ncomplex reflections called?\n");
scanf("%s", comp2);
if (((fin1=fopen(comp1, "r"))==0)||((fin2=fopen(comp2,"r"))==0)){
printf("Unable to open one or both files\n");
return;
}
/* Read in constraining factor and sample spacing */
printf("What is the value of the constraining factor?\n");
scanf("%le", &constrain_factor);
printf("What is the sample spacing in milliseconds?\n");
printf("(for 50kHz samp freq the sample spacing is 0.02ms)\n");
scanf("%le", &samp_space);
samp_num=0;
/* Read values from input files into arrays */
while (fscanf(fin1, "%le %le", &real1, &imag1)==2
&& fscanf(fin2, "%le %le", &real2, &imag2)==2){
/* Perform complex division */
```

```
real[samp_num]=real1*real2+imag1*imag2;
real[samp_num]=real[samp_num]/
(pow(real1,2)+pow(imag1,2)+constrain_factor);
imag[samp_num]=real1*imag2-imag1*real2;
imag[samp_num]=imag[samp_num]/
(pow(real1,2)+pow(imag1,2)+constrain_factor);
samp_num++;
}
/* Compute inverse transform */
myfft(real, imag, samp_num, 1);
/* Read in filename and open file for deconvolved data */
printf("What do you wish to call the output file containing"
"\nthe deconvolved data?\n");
scanf("%s", deconvdata);
fout=fopen(deconvdata, "w");
/* Write deconvolved data to file and correct FFT scaling factor */
for (i=0;i<samp_num;i++){
fprintf(fout, "%13.6lf % 17.14le\n",
samp_space*i, real[i]/sqrt(samp_num));
}
/* Free pointers */
farfree(real);
farfree(imag);
/* Close files */
fclose(fin1);
fclose(fin2);
fclose(fout);
}
/*********************************************************************/
/***************************** finddc ****************************/
void finddc (void)
{
FILE *fin;
char deconvdata[NAMESIZE];
double time, signal, total=0, count=0;
/* Read in filename and open file containing deconvolved data */
printf("Enter the name of the deconvolved data file\n");
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata,"r"))==0){
printf("Unable to open\n");
return;
}
/* Average over 1ms-2ms range to find dc value */
while (fscanf(fin, "%lf %lf", &time, &signal)==2){
if ((time>1)&&(time<2)){
total=total+signal;
count=count+1;
}
}
printf("Dc offset=%le", total/count);
fclose(fin);
}
/*********************************************************************/
/***************************** removedc **************************/
void removedc (void)
{
FILE *fin, *fout;
char deconvdata[NAMESIZE], output[NAMESIZE];
double time, signal, dc;
/* Read in filename and open file containing deconvolved data */
printf("Enter the name of the deconvolved data file\n");
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata,"r"))==0){
printf("Unable to open\n");
return;
}
/* Read in filename and open file for dc-less */
/* input impulse response */
printf("Enter name for the output file containing deconvolved\n");
printf("data with dc offset removed (input impulse response)\n");
scanf("%s", output);
fout=fopen(output, "w");
printf("Enter the dc value\n");
scanf("%lf", &dc);
/* Remove dc offset and print iir to file */
while (fscanf(fin, "%lf %lf", &time, &signal)==2){
fprintf(fout, "%lf %lf\n", time, signal-dc);
}
```

```
fclose(fin);
fclose(fout);
}
/*********************************************************************/
/***************************** repl1ms ***************************/
void repl1ms (void)
{
FILE *fin, *fout;
char input[NAMESIZE], output[NAMESIZE];
double time, signal;
/* Read in filename and open file containing iir */
printf("Enter the name of the input impulse response file\n");
scanf("%s", input);
if ((fin=fopen(input,"r"))==0){
printf("Unable to open\n");
return;
}
/* Read in filename and open file for iir with first 1ms blanked */
printf("Enter name for the output file containing the input\n");
printf("impulse response with first millisecond blanked\n");
scanf("%s", output);
fout=fopen(output, "w");
while (fscanf(fin, "%lf %lf", &time, &signal)==2){
if (time<1.0){
fprintf(fout, "%lf %lf\n", time, 0.0);
}
else{
fprintf(fout, "%lf %lf\n", time, signal);
}
}
fclose(fin);
fclose(fout);
}
/*********************************************************************/
/***************************** ware_aki***************************/
void ware_aki (void)
{
FILE *fin, *fout;
int samp_num=0, i, j;
char deconvdata[NAMESIZE], ware[NAMESIZE];
double samp_space, radius, area, product=1, numerator, time,
speed, length, finlen;
/* Set up arrays using pointers */
double huge *reflec, *F, *G, *r;
if ((reflec=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((F=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((G=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((r=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
printf("\nWARE-AKI BORE RECONSTRUCTION\n");
/* Read in filename and open file containing iir */
printf("What is the name of the file containing the input\n"
"impulse response?\n");
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata, "r"))==0){
printf("Unable to open file\n");
return;
}
/* Read in filename and open file for Ware-Aki */
/* reconstruction data */
printf("What do you wish to call the output file containing the\n"
"Ware-Aki algorithm reconstruction data?\n");
scanf("%s", ware);
fout=fopen(ware, "w");
while (fscanf(fin, "%lf %le", &time, &reflec[samp_num])==2){
if (samp_num==1){
samp_space=time;
```

```
}
samp_num++;
}
/* Ware-Aki algorithm */
printf("What is the starting radius in metres?\n");
scanf("%le", &radius);
printf("What length do you wish to reconstruct up to?\n");
scanf("%le", &finlen);
printf("What is the speed of sound (in m/s)?\n");
scanf("%le", &speed);
area=3.14159*pow(radius,2);
length=samp_space*speed/2000;
/* Set polynomials=0 initially */
for (i=0; i<samp_num; i++){
F[i]=0;
G[i]=0;
}
F[0]=1;
for(i=0; i<samp_num; i++){
fprintf(fout, "% 17.14le  % 17.14le\n", i*length, radius);
if(i>0){
product=product*(1-r[i-1])*(1+r[i-1]);
}
numerator=0;
for(j=0; j<=i; j++){
numerator=numerator+(F[j]*reflec[i-j]);
}
r[i]=numerator/product;
area=area*(1-r[i])/(1+r[i]);
if (area<0){
return;
}
radius=sqrt(area/3.14159);
fprintf(fout, "% 17.14le  % 17.14le\n", i*length, radius);
if (finlen<(i*length)){
return;
}
j=i;
while(j>=1){
G[j]=r[i]*F[j]+G[j-1];
F[j]=F[j]+r[i]*G[j-1];
j--;
}
F[0]=1;
G[0]=r[i];
}
/* Free pointers */
farfree(F);
farfree(G);
farfree(reflec);
farfree(r);
/* Close file */
fclose(fout);
}
/*********************************************************************/
/******************************** noam ***************************/
void noam (int menu_num)
{
FILE *fin, *fout;
int samp_num=0, i, j, l;
char deconvdata[NAMESIZE], amir[NAMESIZE];
double samp_space, temp_value, length, samp_freq, rho=1.21,
visc=0.0000181, cond=0.024, cp=1400, ratio=1.4, speed,
v, A, B, C, D, E, rvnndiv2, vpnndiv2, omega, rv, alpha,
vprecip, radius, area, r, finlen, time, sum, G;
static double a[41][41];
/* Set up arrays using pointers */
double huge *reflec_real, *reflec_imag, *input_real, *input_imag,
*fre, *fim, *rr, *e, *k, *Areal, *Aimag, *Hreal, *Himag;
if ((reflec_real=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((reflec_imag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((input_real=farmalloc(2048*sizeof(double)))==0){
```

```
perror("Malloc failed\n");
exit(0);
}
if ((input_imag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((fre=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((fim=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((rr=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((e=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((k=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((Areal=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((Aimag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((Hreal=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((Himag=farmalloc(2048*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
printf("\nNOAM AMIR'S ");
if (menu_num==7){
printf("LOSSLESS ");
}
if (menu_num==8){
printf("LOSSY (rotating phase) ");
}
if (menu_num==9){
printf("LOSSY (all pole model) ");
}
printf("BORE RECONSTRUCTION\n");
/* Read in filename and open file containing iir */
printf("What is the name of the file containing the input\n"
"impulse response?\n");
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata, "r"))==0){
printf("Unable to open file\n");
return;
}
/* Read in filename and open file for Noam algorithm */
/* reconstruction data */
printf("What do you wish to call the output file containing the\n"
"Noam algorithm reconstruction data?\n");
scanf("%s", amir);
fout=fopen(amir, "w");
while (fscanf(fin, "%lf %le", &time, &reflec_real[samp_num])==2){
if (samp_num==1){
samp_space=time;
}
samp_num++;
}
/* Noam's algorithm */
printf("What is the starting radius in metres?\n");
scanf("%le", &radius);
printf("What length do you wish to reconstruct up to?\n");
```

```
scanf("%le", &finlen);
printf("What is the speed of sound (in m/s)?\n");
scanf("%le", &speed);
area=3.14159*pow(radius,2);
length=samp_space*speed/2000;
/* Construct input pulse */
input_real[0]=1;
for (i=1; i<samp_num; i++){
input_real[i]=0;
}
/* Main Loop */
for (i=0; i<samp_num; i++){
fprintf(fout, "% 17.14le  % 17.14le\n", i*length,
radius);
r=reflec_real[0]/input_real[0];
area=area*(1-r)/(1+r);
if (area<0){
return;
}
radius=sqrt(area/3.14159);
fprintf(fout, "% 17.14le  % 17.14le\n", i*length,
radius);
if (finlen<(i*length)){
return;
}
/* Scattering junction */
for (j=0; j<samp_num; j++){
temp_value=input_real[j];
input_real[j]=(input_real[j]-r*reflec_real[j])/(1-r);
reflec_real[j]=(reflec_real[j]-r*temp_value)/(1-r);
}
/* Delay */
for (j=0;j<(samp_num-1); j++){
reflec_real[j]=reflec_real[j+1];
}
reflec_real[samp_num-1]=0;
/* Lossy part of algorithm */
if (menu_num==8){
/* Rotating phase */
/* Calculate continuous frequency response of segment */
samp_freq=1000/samp_space;
v=sqrt(visc*cp/cond);
A=(1/sqrt(2))*(1+(ratio-1)/v);
B=1+(ratio-1)/v-0.5*(ratio-1)/pow(v,2)-0.5*pow((ratio-1)/v,2);
C=7/8+(ratio-1)/v-0.5*(ratio-1)/pow(v,2)-1/8*(ratio-1)/pow(v,3);
C=C-0.5*pow((ratio-1)/v,2)+0.5*pow((ratio-1),2)/pow(v,3);
C=(1/sqrt(2))*(C+0.5*pow((ratio-1)/v,3));
D=A;
E=C;
rvnndiv2=radius*sqrt(rho*2*3.14159*samp_freq/(2*visc));
vpnndiv2=(1/speed)*(1+D/rvnndiv2+E/pow(rvnndiv2,3));
fre[0]=1;
fim[0]=0;
for (j=1; j<=(samp_num/2); j++){
omega=2*3.14159*j*samp_freq/samp_num;
rv=radius*sqrt(rho*omega/visc);
alpha=(omega/speed)*(A/rv+B/pow(rv,2)+C/pow(rv,3));
vprecip=(1/speed)*(1+D/rv+E/pow(rv,3));
fre[j]=exp(-alpha*length);
fre[j]=fre[j]*cos(omega*length*(vpnndiv2-vprecip));
fim[j]=exp(-alpha*length);
fim[j]=fim[j]*sin(omega*length*(vpnndiv2-vprecip));
}
for (j=(samp_num/2); j<samp_num; j++){
fre[j]=fre[samp_num-j];
fim[j]=-fim[samp_num-j];
}
for (j=0; j<samp_num; j++){
input_imag[j]=0;
reflec_imag[j]=0;
}
/* Convolve with input pulse */
myfft(input_real,input_imag, samp_num, -1);
for (j=0; j<samp_num; j++){
temp_value=input_real[j];
input_real[j]=input_real[j]*fre[j]-input_imag[j]*fim[j];
input_imag[j]=temp_value*fim[j]+input_imag[j]*fre[j];
}
```

```
myfft(input_real, input_imag, samp_num, 1);
/* Deconvolve with output pulses */
myfft(reflec_real, reflec_imag, samp_num, -1);
for (j=0; j<samp_num; j++){
temp_value=reflec_real[j];
reflec_real[j]=(reflec_real[j]*fre[j]+reflec_imag[j]*fim[j]);
reflec_real[j]=reflec_real[j]/(pow(fre[j],2)+pow(fim[j],2));
reflec_imag[j]=reflec_imag[j]*fre[j]-temp_value*fim[j];
reflec_imag[j]=reflec_imag[j]/(pow(fre[j],2)+pow(fim[j],2));
}
myfft(reflec_real, reflec_imag, samp_num, 1);
}
else if (menu_num==9){
/* All pole model */
/* Calculate continuous frequency response of segment */
samp_freq=1000/samp_space;
v=sqrt(visc*cp/cond);
A=(1/sqrt(2))*(1+(ratio-1)/v);
B=1+(ratio-1)/v-0.5*(ratio-1)/pow(v,2)-0.5*pow((ratio-1)/v,2);
C=7/8+(ratio-1)/v-0.5*(ratio-1)/pow(v,2)-1/8*(ratio-1)/pow(v,3);
C=C-0.5*pow((ratio-1)/v,2)+0.5*pow((ratio-1),2)/pow(v,3);
C=(1/sqrt(2))*(C+0.5*pow((ratio-1)/v,3));
D=A;
E=C;
fre[0]=1;
fim[0]=0;
for (j=1; j<=(samp_num/2); j++){
omega=2*3.14159*j*samp_freq/samp_num;
rv=radius*sqrt(rho*omega/visc);
alpha=(omega/speed)*(A/rv+B/pow(rv,2)+C/pow(rv,3));
vprecip=(1/speed)*(1+D/rv+E/pow(rv,3));
fre[j]=exp(-alpha*length);
fre[j]=fre[j]*cos(-omega*length*vprecip);
fim[j]=exp(-alpha*length);
fim[j]=fim[j]*sin(-omega*length*vprecip);
}
for (j=(samp_num/2); j<samp_num; j++){
fre[j]=fre[samp_num-j];
fim[j]=-fim[samp_num-j];
}
fre[samp_num/2]=sqrt(pow(fre[samp_num/2],2.0)
+pow(fim[samp_num/2],2.0));
fim[samp_num/2]=0.0;
/* Inverse fft continuous complex expression */
myfft(fre, fim, samp_num, 1);
/* Correct FFT scaling factor */
for (j=0; j<samp_num; j++){
fre[j]=fre[j]/sqrt(samp_num);
fim[j]=fim[j]/sqrt(samp_num);
}
/* Calculate autocorrelation function */
for (j=0; j<samp_num; j++){
rr[j]=0.0;
for (l=0; l<samp_num; l++){
if (l<(samp_num-j)){
rr[j]=rr[j]+fre[l]*fre[l+j];
}
else{
rr[j]=rr[j]+fre[l]*fre[l+j-samp_num];
}
}
}
/* Calculate a values */
e[0]=rr[0];
k[1]=-rr[1]/e[0];
a[1][1]=k[1];
e[1]=(1.0-pow(k[1],2.0))*e[0];
for (j=2; j<41; j++){
sum=0.0;
for (l=1; l<j; l++){
sum=sum+a[l][j-1]*rr[j-l];
}
k[j]=-(rr[j]+sum)/e[j-1];
a[j][j]=k[j];
for (l=1; l<j; l++){
a[l][j]=a[l][j-1]+k[j]*a[j-l][j-1];
}
e[j]=(1.0-pow(k[j],2.0))*e[j-1];
```

```
}
/* Calculate gain G */
G=rr[0];
for (j=1; j<41; j++){
G=G+a[j][40]*rr[j];
}
G=sqrt(G);
/* Calculate A(z) */
Areal[0]=1.0;
Aimag[0]=0.0;
for (j=1; j<41; j++){
Areal[j]=a[j][40];
Aimag[j]=0.0;
}
for (j=41; j<samp_num; j++){
Areal[j]=0.0;
Aimag[j]=0.0;
}
/* Fourier transform to calculate A(z) */
myfft(Areal, Aimag, samp_num, -1);
/* Correct FFT scaling factor and calculate H(z) */
for (j=0; j<samp_num; j++){
Areal[j]=Areal[j]*sqrt(samp_num);
Aimag[j]=Aimag[j]*sqrt(samp_num);
Hreal[j]=G*Areal[j]/(pow(Areal[j],2.0)+pow(Aimag[j],2.0));
Himag[j]=-G*Aimag[j]/(pow(Areal[j],2.0)+pow(Aimag[j],2.0));
}
for (j=0; j<samp_num; j++){
input_imag[j]=0;
reflec_imag[j]=0;
}
/* Convolve with input pulse */
myfft(input_real, input_imag, samp_num, -1);
for (j=0; j<samp_num; j++){
temp_value=input_real[j];
input_real[j]=input_real[j]*Hreal[j]-input_imag[j]*Himag[j];
input_imag[j]=temp_value*Himag[j]+input_imag[j]*Hreal[j];
}
myfft(input_real, input_imag, samp_num, 1);
/* Deconvolve with output pulses */
myfft(reflec_real, reflec_imag, samp_num, -1);
for (j=0; j<samp_num; j++){
temp_value=reflec_real[j];
reflec_real[j]=(reflec_real[j]*Hreal[j]+reflec_imag[j]*Himag[j]);
reflec_real[j]=reflec_real[j]/(pow(Hreal[j],2)+pow(Himag[j],2));
reflec_imag[j]=reflec_imag[j]*Hreal[j]-temp_value*Himag[j];
reflec_imag[j]=reflec_imag[j]/(pow(Hreal[j],2)+pow(Himag[j],2));
}
myfft(reflec_real, reflec_imag, samp_num, 1);
}
}
/* Free pointers */
farfree(input_real);
farfree(input_imag);
farfree(reflec_real);
farfree(reflec_imag);
farfree(fre);
farfree(fim);
farfree(rr);
farfree(e);
farfree(k);
farfree(Areal);
farfree(Aimag);
farfree(Hreal);
farfree(Himag);
/* Close files */
fclose(fout);
}
/*********************************************************************/
/*************************   zeropad   ***************************/
void zeropad (void)
{
FILE *fin, *fout;
char deconvdata[NAMESIZE], zeropadded[NAMESIZE];
long padsize, i, j;
double time, signal, samp_space;
/* Read in filename and open file containing iir */
printf("Enter the name of the input impulse response file\n");
```

```
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata,"r"))==0){
printf("Unable to open\n");
return;
}
/* Read in filename and open file for padded iir data */
printf("Enter the name of the padded file\n");
scanf("%s", zeropadded);
fout=fopen(zeropadded, "w");
printf("What size would you like iir to be padded to?\n");
printf("(maximum size=8192)\n");
scanf("%ld", &padsize);
if (padsize>8192){
printf("This is too big\n");
return;
}
/* Print zeropadded iir to file */
i=0;
while (fscanf(fin, "%lf %lf", &time, &signal)==2){
fprintf(fout, "%lf %lf\n", time, signal);
if (i==1){
samp_space=time;
}
i++;
}
for (j=i; j<padsize; j++){
fprintf(fout, "%lf %lf\n", j*samp_space, 0.0);
}
fclose(fin);
fclose(fout);
}
/*********************************************************************/
/************************** input_imped **************************/
void input_imped (void)
{
FILE *fin, *fout, *fout1;
long samp_num=0, i;
char deconvdata[NAMESIZE], imped[NAMESIZE], impedcom[NAMESIZE];
double samp_space, radius, imp_real, imp_imag, speed, time;
/* Set up arrays using pointers */
double huge *reflec_real, *reflec_imag;
if ((reflec_real=farmalloc(8192L*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
if ((reflec_imag=farmalloc(8192L*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
printf("\nINPUT IMPEDANCE PROGRAM\n");
/* Read in input filename and open file containing */
/* iir data (padded or not) */
printf("What is the name of the file containing the input\n"
"impulse response?\n");
scanf("%s", deconvdata);
if ((fin=fopen(deconvdata, "r"))==0){
printf("Unable to open file\n");
return;
}
while (fscanf(fin, "%lf %le", &time, &reflec_real[samp_num])==2){
if (samp_num==1){
samp_space=time;
}
samp_num++;
}
printf("What is the source tube radius?\n");
scanf("%lf", &radius);
printf("What is the speed of sound?\n");
scanf("%lf", &speed);
/* Create imaginary part=0 and correct FFT scaling factor */
for (i=0; i<samp_num; i++){
reflec_real[i]=reflec_real[i]*sqrt(samp_num);
reflec_imag[i]=0.0;
}
/* Fourier transform */
myfft(reflec_real, reflec_imag, samp_num, -1);
/* Read in filename and open file for magnitude impedance data */
printf("What do you wish to call the output file containing the\n"
```

```
"magnitude impedance data?\n");
scanf("%s", imped);
fout=fopen(imped, "w");
/* Read in filename and open file for complex impedance data */
printf("What do you wish to call the output file containing the\n"
"complex impedance data?\n");
scanf("%s", impedcom);
fout1=fopen(impedcom,"w");
/* Impedance algorithm */
for (i=0; i<samp_num; i++){
imp_real=(1-pow(reflec_real[i],2)-pow(reflec_imag[i],2))/
(1-2*reflec_real[i]+pow(reflec_real[i],2)+
pow(reflec_imag[i],2));
imp_imag=(2*reflec_imag[i])/(1.0-2*reflec_real[i]+
pow(reflec_real[i],2)+pow(reflec_imag[i],2));
imp_real=imp_real*((1.21*speed)/(3.14159*pow(radius,2)));
imp_imag=imp_imag*((1.21*speed)/(3.14159*pow(radius,2)));
fprintf(fout, "% 17.14le  % 17.14le\n",
i*1.0/(samp_space*samp_num),
sqrt(pow(imp_real,2)+pow(imp_imag,2)));
fprintf(fout1, "% 17.14le  % 17.14le\n", imp_real, imp_imag);
}
/* Free pointers */
farfree(reflec_real);
farfree(reflec_imag);
/* Close file */
fclose(fout);
fclose(fout1);
}
/*********************************************************************/
/************************** farewell *****************************/
void farewell (void)
{
printf("Goodbye\n");
}
/*********************************************************************/
/*************************** myfft *******************************/
void myfft(double huge *real, double huge *imag,
unsigned long samp_num, int isign)
{
int i=0;
double huge *data;
if ((data=farmalloc(17000L*sizeof(double)))==0){
perror("Malloc failed\n");
exit(0);
}
for (i=1; i<(2*samp_num+1); i=i+2){
data[i]=real[(i-1)/2];
data[i+1]=imag[(i-1)/2];
}
dfour1(data, samp_num, isign);
for (i=1; i<(2*samp_num+1); i=i+2){
real[(i-1)/2]=data[i]/sqrt(samp_num);
imag[(i-1)/2]=data[i+1]/sqrt(samp_num);
}
farfree(data);
}
/*********************************************************************/
/******************* dfour1 (Numerical recipes) ******************/
void dfour1(double huge *data, unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
double tempr,tempi;
n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {
if (j > i) {
SWAP(data[j],data[i]);
SWAP(data[j+1],data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
j -= m;
m >>= 1;
}
j += m;
}
```

```
mmax=2;
while (n > mmax) {
istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);
wtemp=sin(0.5*theta);
wpr = -2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for (m=1;m<mmax;m+=2) {
for (i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*data[j]-wi*data[j+1];
tempi=wr*data[j+1]+wi*data[j];
data[j]=data[i]-tempr;
data[j+1]=data[i+1]-tempi;
data[i] += tempr;
data[i+1] += tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}
/*************************************************************************/
```

B.3 Program to find the theoretical impedance curve of a stepped cylinder

```
/*     PROGRAM TO FIND THEORETICAL IMPEDANCE CURVE OF A STEPPED */
/*     CYLINDER DRIVEN AT ONE END AND OPEN AT THE OTHER */
/* Include library files */
#include <stdio.h>
#include <math.h>
/* Declare functions */
void main (void);
void compmult (double aa, double bb, double cc, double dd,
double *ee, double *ff);
void compdiv (double aa, double bb, double cc, double dd,
double *ee, double *ff);
void comptan (double aa, double bb, double *ee, double *ff);
/* Main program */
void main (void)
{
FILE   *fout1, *fout2;
double rho, c, gamma, cp, n, kappa, r1, r2, r3, l1, l2, l3,
term1, term2, om, k, alphacy1, alphacy2, alphacy3, numreal,
numimag, denreal, denimag, zcreal, zcimag, zbreal, zbimag,
zareal, zaimag, tempreal, tempimag, s1, s2, s3;
int   i;
char   impedcom[80], impedmag[80];
printf("THEORETICAL IMPEDANCE CURVE FOR STEPPED TUBE\n\n");
printf("What is your value for the air density (try 1.21)?\n");
scanf("%lf", &rho);
printf("What is your value for the speed of sound in air?\n");
scanf("%lf", &c);
printf("What is your value for the ratio of specific\n");
printf("heats of air (try1.4)?\n");
scanf("%lf", &gamma);
printf("What is your value for the specific heat of air at\n");
printf("constant pressure (try 1400)?\n");
scanf("%lf", &cp);
printf("What is your value for the coefficient of shear\n");
printf("viscosity of air (try 0.0000181)?\n");
scanf("%lf", &n);
printf("What is your value for the thermal conductivity of\n");
printf("air (try 0.024)?\n");
scanf("%lf", &kappa);
printf("What is your value for r1?\n");
scanf("%lf", &r1);
printf("What is your value for l1?\n");
scanf("%lf", &l1);
printf("What is your value for r2?\n");
scanf("%lf", &r2);
```

```
printf("What is your value for l2?\n");
scanf("%lf", &l2);
printf("What is your value for r3?\n");
scanf("%lf", &r3);
printf("What is your value for l3?\n");
scanf("%lf", &l3);
s1=M_PI*r1*r1;
s2=M_PI*r2*r2;
s3=M_PI*r3*r3;
printf("Enter the name of the file for the complex impedance\n");
scanf("%s", impedcom);
fout1=fopen(impedcom, "w");
printf("Enter the name of the file for the impedance magnitude\n");
scanf("%s", impedmag);
fout2=fopen(impedmag, "w");
/* Calculations loop  */
for (i=1; i<5000; i++){
/* Calculate alpha */
om=2.0*M_PI*i;
k=om/c;
term1=sqrt((n*om)/(2.0*rho));
term2=(gamma-1.0)*sqrt((kappa*om)/(2.0*rho*cp));
alphacy1=(term1+term2)/(r1*c);
alphacy2=(term1+term2)/(r2*c);
alphacy3=(term1+term2)/(r3*c);
/* Calculation of zcreal and zcimag */
/* Calculate numerator */
compmult(k, -alphacy3, l3, 0.0, &numreal, &numimag);
comptan(numreal, numimag, &numreal, &numimag);
compmult(0.6*r3, 0.0, k, -alphacy3, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
compmult(0.0, 1.0, numreal, numimag, &numreal, &numimag);
compmult(0.25*r3*r3, 0.0, k, -alphacy3, &tempreal, &tempimag);
compmult(tempreal, tempimag, k, -alphacy3, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
/* Calculate denominator */
compmult(k, -alphacy3, l3, 0.0, &denreal, &denimag);
comptan(denreal, denimag, &denreal, &denimag);
compmult(0.0, 0.25*r3*r3, denreal, denimag, &denreal, &denimag);
compmult(denreal, denimag, k, -alphacy3, &denreal, &denimag);
compmult(denreal, denimag, k, -alphacy3, &denreal, &denimag);
compmult(k, -alphacy3, l3, 0.0, &tempreal, &tempimag);
comptan(tempreal, tempimag, &tempreal, &tempimag);
compmult(-0.6*r3, 0.0, tempreal, tempimag, &tempreal, &tempimag);
compmult(tempreal, tempimag, k, -alphacy3, &tempreal, &tempimag);
denreal=denreal+tempreal+1.0;
denimag=denimag+tempimag;
/* Calculate whole expression */
compdiv(numreal, numimag, denreal, denimag, &zcreal, &zcimag);
compmult(zcreal, zcimag, rho*om/s3, 0.0, &zcreal, &zcimag);
compdiv(zcreal, zcimag, k, -alphacy3, &zcreal, &zcimag);
/* Calculation of zbreal and zbimag */
/* Calculate numerator */
compmult(l2, 0.0, k, -alphacy2, &numreal, &numimag);
comptan(numreal, numimag, &numreal, &numimag);
compmult(0.0, 1.0, numreal, numimag, &numreal, &numimag);
compmult(s2/(rho*om), 0.0, k, -alphacy2, &tempreal, &tempimag);
compmult(tempreal, tempimag, zcreal, zcimag, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
/* Calculate denominator */
compmult(l2, 0.0, k, -alphacy2, &denreal, &denimag);
comptan(denreal, denimag, &denreal, &denimag);
compmult(0.0, s2/(rho*om), denreal, denimag, &denreal, &denimag);
compmult(k, -alphacy2, denreal, denimag, &denreal, &denimag);
compmult(zcreal, zcimag, denreal, denimag, &denreal, &denimag);
denreal=denreal+1.0;
/* Calculate whole expression */
compdiv(numreal, numimag, denreal, denimag, &zbreal, &zbimag);
compmult(zbreal, zbimag, rho*om/s2, 0.0, &zbreal, &zbimag);
compdiv(zbreal, zbimag, k, -alphacy2, &zbreal, &zbimag);
/* Calculation of zareal and zaimag */
/* Calculate numerator */
compmult(l1, 0.0, k, -alphacy1, &numreal, &numimag);
comptan(numreal, numimag, &numreal, &numimag);
compmult(0.0, 1.0, numreal, numimag, &numreal, &numimag);
```

```
compmult(s1/(rho*om), 0.0, k, -alphacy1, &tempreal, &tempimag);
compmult(tempreal, tempimag, zbreal, zbimag, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
/* Calculate denominator */
compmult(l1, 0.0, k, -alphacy1, &denreal, &denimag);
comptan(denreal, denimag, &denreal, &denimag);
compmult(0.0, s1/(rho*om), denreal, denimag, &denreal, &denimag);
compmult(k, -alphacy1, denreal, denimag, &denreal, &denimag);
compmult(zbreal, zbimag, denreal, denimag, &denreal, &denimag);
denreal=denreal+1.0;
/* Calculate whole expression */
compdiv(numreal, numimag, denreal, denimag, &zareal, &zaimag);
compmult(zareal, zaimag, rho*om/s1, 0.0, &zareal, &zaimag);
compdiv(zareal, zaimag, k, -alphacy1, &zareal, &zaimag);
fprintf(fout1, "%le %le\n", zareal, zaimag);
fprintf(fout2, "%lf %le\n",
i/1000.0, sqrt(pow(zareal,2.0)+pow(zaimag,2.0)));
}
}
/***************************** Functions ************************/
/***************************** compmult *************************/
void compmult (double aa, double bb, double cc, double dd,
double *ee, double *ff)
{
double real, imag;
real=aa*cc-bb*dd;
imag=bb*cc+aa*dd;
*ee=real;
*ff=imag;
}
/************************************************************/
/***************************** compdiv **************************/
void compdiv (double aa, double bb, double cc, double dd,
double *ee, double *ff)
{
double real, imag;
real=(aa*cc+bb*dd)/(cc*cc+dd*dd);
imag=(bb*cc-aa*dd)/(cc*cc+dd*dd);
*ee=real;
*ff=imag;
}
/************************************************************/
/***************************** comptan **************************/
void comptan (double aa, double bb, double *ee, double *ff)
{
double real, imag;
real=2*cos(aa)*sin(aa)/(cosh(2*bb)+pow(cos(aa),2)-pow(sin(aa),2));
imag=sinh(2*bb)/(cosh(2*bb)+pow(cos(aa),2)-pow(sin(aa),2));
*ee=real;
*ff=imag;
}
/************************************************************/
```

B.4 Program to predict the size of a leak in a leaking cylinder

```
/* PROGRAM TO PREDICT THE SIZE OF A LEAK IN A LEAKING CYLINDER */
/* (OPEN AT ONE END) */
/* Include library files */
#include <stdio.h>
#include <math.h>
/* Declare functions */
void main (void);
void compmult (double aa, double bb, double cc, double dd,
double *ee, double *ff);
void compdiv (double aa, double bb, double cc, double dd,
double *ee, double *ff);
void comptan (double aa, double bb, double *ee, double *ff);
/* Main program */
void main (void)
{
FILE *fin, *fout;
char impedcom[80], holedata[80];
double l1, l2, r, S, lh, rh, om, rho, c, zbreal, zbimag,
```

```
zcreal, zcimag, zhreal, zhimag, k, zareal, zaimag,
freq, numreal, numimag, denreal, denimag, i=0, gamma,
cp, n, kappa, term1, term2, acubic, bcubic, ccubic,
alphacy=0, alphahrh=0, tempreal, tempimag, R, Q,
theta, rh1, rh2, rh3;
printf(" PROGRAM TO PREDICT THE SIZE OF A LEAK IN A LEAKING");
printf("CYLINDER (open at one end)\n\n");
printf("What is the length of cylinder section 1? (in m)\n");
scanf("%le", &l1);
printf("What is the length of cylinder section 2? (in m)\n");
scanf("%le", &l2);
printf("What is the cylinder radius? (in m)\n");
scanf("%le", &r);
S=M_PI*pow(r,2.0);
printf("What is the depth of the leak (in m)?\n");
printf("i.e. the cylinder wall thickness\n");
scanf("%le", &lh);
printf("What is the air density (try 1.21)?\n");
scanf("%le", &rho);
printf("What is the speed of sound (in m/s)?\n");
scanf("%le", &c);
printf("What is the value of the ratio of specific heats of ");
printf("air (gamma) ? \n(try 1.4)\n");
scanf("%le", &gamma);
printf("What is the value of the specific heat of air at constant");
printf(" pressure (cp) ? \n(try 1400)\n");
scanf("%le", &cp);
printf("What is the value of the coefficient of shear viscosity");
printf(" of air (n) ? \n (try 0.0000181)\n");
scanf("%le", &n);
printf("What is the value of the thermal conductivity of ");
printf("air (kappa) ? \n(try 0.024)\n");
scanf("%le", &kappa);
printf("What is the frequency spacing (in Hz)\n");
printf("i.e. resolution of impedance curve?\n");
scanf("%le", &freq);
/* Read in filename and open file containing leaking cylinder */
/* complex impedance data */
printf("Enter the name of the complex impedance file\n");
scanf("%s", impedcom);
if ((fin=fopen(impedcom,"r"))==0){
printf("Unable to open\n");
return;
}
/* Read in filename and open file for hole radius predictions */
printf("Enter the name of the file for the hole radius\n");
printf("predictions\n");
scanf("%s", holedata);
fout=fopen(holedata, "w");
/* Main loop */
/* Read in complex impedance */
while ((fscanf(fin, "%le %le", &zareal, &zaimag)==2)&&(i<5050)){
if(i==0){
fscanf(fin, "%le %le", &zareal, &zaimag);
i=i+1.0*freq;
}
om=2.0*M_PI*i;
k=om/c;
term1=sqrt((n*om)/(2.0*rho));
term2=(gamma-1.0)*sqrt((kappa*om)/(2.0*rho*cp));
alphacy=(term1+term2)/(r*c);
alphahrh=(term1+term2)/c;
/* Calculation of zbreal and zbimag */
/* Calculate numerator */
compmult(k, -alphacy, l1, 0.0, &numreal, &numimag);
comptan(numreal, numimag, &numreal, &numimag);
compmult(numreal, numimag, 0.0, -rho*om/S, &numreal, &numimag);
compdiv(numreal, numimag, k, -alphacy, &numreal, &numimag);
numreal=numreal+zareal;
numimag=numimag+zaimag;
/* Calculate denominator */
compmult(k, -alphacy, l1, 0.0, &denreal, &denimag);
comptan(denreal, denimag, &denreal, &denimag);
compmult(denreal, denimag, 0.0, -S/(rho*om), &denreal, &denimag);
compmult(denreal, denimag, zareal, zaimag, &denreal, &denimag);
compmult(denreal, denimag, k, -alphacy, &denreal, &denimag);
denreal=denreal+1.0;
/* Calculate whole expression */
```

```
compdiv(numreal, numimag, denreal, denimag, &zbreal, &zbimag);
/* Calculation of zcreal and zcimag */
/* Calculate numerator */
compmult(k, -alphacy, l2, 0.0, &numreal, &numimag);
comptan(numreal, numimag, &numreal, &numimag);
compmult(k, -alphacy, 0.6*r, 0.0, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
compmult(0.0, 1.0, numreal, numimag, &numreal, &numimag);
compmult(k, -alphacy, k, -alphacy, &tempreal, &tempimag);
compmult(0.25*r*r, 0.0, tempreal, tempimag, &tempreal, &tempimag);
numreal=numreal+tempreal;
numimag=numimag+tempimag;
/* Calculate denominator */
compmult(k, -alphacy, l2, 0.0, &denreal, &denimag);
comptan(denreal, denimag, &denreal, &denimag);
compmult(0.25*r*r, 0.0, denreal, denimag, &denreal, &denimag);
compmult(k, -alphacy, k, -alphacy, &tempreal, &tempimag);
compmult(denreal, denimag, tempreal, tempimag, &denreal, &denimag);
compmult(0.0, 1.0, denreal, denimag, &denreal, &denimag);
compmult(k, -alphacy, l2, 0.0, &tempreal, &tempimag);
comptan(tempreal, tempimag, &tempreal, &tempimag);
compmult(-0.6*r, 0.0, tempreal, tempimag, &tempreal, &tempimag);
compmult(k, -alphacy, tempreal, tempimag, &tempreal, &tempimag);
denreal=denreal+tempreal;
denimag=denimag+tempimag;
denreal=denreal+1.0;
/* Calculate whole expression */
compdiv(numreal, numimag, denreal, denimag, &zcreal, &zcimag);
compmult(zcreal, zcimag, rho*om/S, 0.0, &zcreal, &zcimag);
compdiv(zcreal, zcimag, k, -alphacy, &zcreal, &zcimag);
/* Calculate zh from zb and zc */
compmult(zcreal, zcimag, zbreal, zbimag, &numreal, &numimag);
denreal=zcreal-zbreal;
denimag=zcimag-zbimag;
compdiv(numreal, numimag, denreal, denimag, &zhreal, &zhimag);
/* Solve cubic equation */
/* Calculate acubic, bcubic and ccubic */
acubic=(zhimag*M_PI*r*r*1.724)/(rho*om);
bcubic=(0.431*r*r*alphahrh)-(2.75*r*r);
ccubic=-1.724*lh*r*r;
/* Calculate Q and R */
Q=(acubic*acubic-3.0*bcubic)/9.0;
R=(2.0*pow(acubic,3.0)-9.0*acubic*bcubic+27.0*ccubic)/54.0;
/* Find cube roots using Q and R */
if ((pow(R,2.0))<(pow(Q,3.0))){
theta=acos(R/(pow(Q,1.5)));
rh1=-2.0*pow(Q,0.5)*cos(theta/3.0)-acubic/3.0;
rh2=-2.0*pow(Q,0.5)*cos((theta+2.0*M_PI)/3.0)-acubic/3.0;
rh3=-2.0*pow(Q,0.5)*cos((theta-2.0*M_PI)/3.0)-acubic/3.0;
}
else{
printf("At %lfHz the cube root can't be found this way\n", i);
}
/* Print out to file the required cube root (the predicted */
/* hole radius) and the frequency at which it was calculated */
fprintf(fout, "%le %le\n", i, rh2);
i=i+freq;
}
}
/***************************** Functions ***************************/
/***************************** compmult ****************************/
void compmult (double aa, double bb, double cc, double dd,
double *ee, double *ff)
{
double real, imag;
real=aa*cc-bb*dd;
imag=bb*cc+aa*dd;
*ee=real;
*ff=imag;
}
/*******************************************************************/
/***************************** compdiv *****************************/
void compdiv (double aa, double bb, double cc, double dd,
double *ee, double *ff)
{
double real, imag;
real=(aa*cc+bb*dd)/(cc*cc+dd*dd);
```

```
imag=(bb*cc-aa*dd)/(cc*cc+dd*dd);
*ee=real;
*ff=imag;
}
/*********************************************************************/
/***************************** comptan ******************************/
void comptan (double aa, double bb, double *ee, double *ff)
{
double real, imag;
real=2*cos(aa)*sin(aa)/(cosh(2*bb)+pow(cos(aa),2)-pow(sin(aa),2));
imag=sinh(2*bb)/(cosh(2*bb)+pow(cos(aa),2)-pow(sin(aa),2));
*ee=real;
*ff=imag;
}
/*********************************************************************/
```

B.5 Program to find the input impulse response of the loudspeaker

```
/* PROGRAM TO FIND THE INPUT IMPULSE RESPONSE OF THE LOUDSPEAKER */
/* The D/A converter sends out voltages in the -5V to +5V range. */
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* The A/D converter samples voltages in the -5V to +5V range.*/
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* Include library files */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "c:\daqbook\dos\bc\daqbook.h"
#include <alloc.h>
#define SWAP(a,b) tempr=(a); (a)=(b); (b)=tempr
/* Declare functions */
void dfour1(double data[], unsigned long nn, int isign);
void myfft(double real[], double imag[],
unsigned long samp_num, int isign);
void _far _pascal myhandler(int error_code);
/* Main program */
void main(void)
{
FILE *fout1, *fout2, *fout3, *fout4, *fout5, *fout6;
int i, y, ipstart, iobstart, isfstart;
double c, temp, del, obstart, sfstart, om, rv, v, A, B, C, D, E,
stlength, stradius, alpha, vpinv, sum;
static char filenm1[80], filenm2[80];
static char filenm3[80], filenm4[80], filenm5[80], filenm6[80];
static double a[41][41], G, tempval;
/* Set up arrays using pointers */
float huge *sample;
unsigned int huge *waveBuf;
unsigned int huge *buf;
float huge *bufav;
double huge *orreal;
double huge *orimag;
double huge *apreal;
double huge *apimag;
double huge *autocor;
double huge *e;
double huge *k;
double huge *srreal;
double huge *srimag;
double huge *sp2real;
double huge *sp2imag;
double huge *lsrreal;
double huge *lsrimag;
printf("PROGRAM TO FIND THE LOUDSPEAKER INPUT IMPULSE RESPONSE\n\n");
/* Set error handler and initialize DaqBoard/100 */
daqSetErrHandler(myhandler);
daqInit(PORT_0300, DMA5+INTERRUPT10);
daqBrdSetDmaMode(DmaRead);
/* Read in filenames and open files */
printf("What do you wish to call the name of the ");
printf("file containing the full microphone output?\n");
scanf("%s", filenm1);
fout1=fopen(filenm1, "w");
```

```
printf("What do you wish to call the file containing the\n");
printf("reflection of the input pulse by the perspex)?\n");
scanf("%s", filenm2);
fout2=fopen(filenm2, "w");
printf("What do you wish to call the file containing the\n");
printf("reflected input pulse just prior to further reflection\n");
printf("by the speaker?\n");
scanf("%s", filenm3);
fout3=fopen(filenm3, "w");
printf("What do you wish to call the file containing the source ");
printf("reflections?\n");
scanf("%s", filenm4);
fout4=fopen(filenm4, "w");
printf("What do you wish to call the file containing the source\n");
printf("reflections just at the exit of the loudspeaker?\n");
scanf("%s", filenm5);
fout5=fopen(filenm5, "w");
printf("What do you wish to call the file containing the\n");
printf("loudspeaker input impulse response?\n");
scanf("%s", filenm6);
fout6=fopen(filenm6, "w");
/* SAMPLING SECTION */
if( (waveBuf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (buf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (bufav=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
/* Setup waveforms for 20 uS update rate on DAC channel 0 only */
daqBrdDacSetMode(20, DacFIFOChan0, 0);
/* Build a user wave - i.e. a 5V input pulse of 80uS long */
for (i=0; i<4; i++) {
waveBuf[i] = 4095;
}
for (i=4; i<4096; i++) {
waveBuf[i] = 2048;
]
/* Specify the user built waveform for DAC channel 0 */
daqBrdDacUserWave(0, (unsigned int far *)waveBuf, 4096);
for (y=0; y<4096; y++){
bufav[y]=0.0;
}
/* Sends out pulse and samples result 1000 times */
/* (samples then averaged) */
for (i=0; i<1000; i++){
/* Set output voltage to 0V */
daqDacWt(0,2048);
/* Delay before playing pulse to ensure all signal died away */
delay(1000);
/* Set up the background sampling */
/* a) Reads from channel 0 at DgainX1 */
daqAdcSetMux(0,0,DgainX1);
/* b) No tagged data */
daqAdcSetTag(0);
/* c) Set sampling frequency = 50kHz */
daqAdcSetFreq(50000);
/* d) Set to trigger from a software command */
daqAdcSetTrig(DtsSoftware,0,0,0,0);
/* Start background sampling. The sampled data is read into */
/* an array called buf (DMA is used to send data to computer) */
daqAdcRdNBack((unsigned int far *) buf, 4096, 0, DusDma);
daqAdcSoftTrig();
/* Start output waveform */
daqBrdDacStart();
/* Introduce a delay to ensure all output and input has */
/* stopped before the daqBoard is stopped */
delay(100);
/* Stop output waveform */
daqBrdDacStop();
for (y=0; y<4096; y++){
bufav[y]=bufav[y]+buf[y]/1000.0;
}
```

```
}
/* Close and exit */
daqClose();
farfree(waveBuf);
/* PROCESSING DATA SECTION */
/* Print averaged data to a file */
if ( (sample=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (y=0; y<4096; y++){
/* Use 2048-buf[], in order to invert signal (after */
/* inversion by amplifier) and to position around y=0 line. */
/* Need to divide by 16.0 because DMA is 16bit */
sample[y]=2048-bufav[y]/16.0;
}
for (y=0; y<4096; y++){
fprintf(fout1, "%d %f\n", y, sample[y]);
}
farfree(buf);
farfree(bufav);
/* Find where input sound pulse passes microphone (define */
/* starting point of input pulse as two samples before the */
/* level rises above 100 - out of a possible 2048) */
for (y=0; y<4096; y++){
if (sample[y]>100){
ipstart=y-2;
break;
}
}
printf("Input sound pulse first apparent at %d\n", ipstart);
/* Calculate where object and source reflections should start */
/* a) Calculate speed of sound from temperature */
printf("What is the temperature in degrees C?\n");
scanf("%lf", &temp);
c=331.6*(sqrt((temp+273.0)/273.0));
/* b) Calculate the delay before object reflections (reflection */
/* of input pulse by perspex) arrive */
/* Length of tube = 2*l2 = 6.18m */
obstart=ipstart+6.18/(0.00002*c);
/* c) Round up or down to nearest integer */
iobstart=obstart;
if ((obstart-iobstart)>=0.5){
iobstart++;
}
printf("Object reflections first apparent at %d\n", iobstart);
/* d) Calculate time taken for signal to travel */
/* length 2*(l1+l2) = 6.192m) */
del=2.0*(6.192/(c*0.00002));
printf("Delay=%lf\n", del);
/* e) Calculate where the source reflections should start */
sfstart=ipstart+del;
/* f) Round up or down to nearest integer */
isfstart=sfstart;
if ((sfstart-isfstart)>=0.5){
isfstart++;
}
printf("Source reflections first apparent at %d\n", isfstart);
/* CALCULATE LOUDSPEAKER REFLECTION RESPONSE */
/* Isolate object reflections - put into a 512 array */
/* ready for FFTing */
if ( (orreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (orimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
orreal[i]=sample[iobstart+i];
orimag[i]=0;
}
for (i=0; i<512; i++){
fprintf(fout2, "%d %lf\n", i, orreal[i]);
}
/* FFT the array of object reflections */
myfft((double far *) orreal, (double far *) orimag, 512, -1);
```

```
/* Correct scaling */
for (i=0; i<512; i++){
orreal[i]=orreal[i]*sqrt(512.0);
orimag[i]=orimag[i]*sqrt(512.0);
}
/* Isolate the source reflections - put into a 512 array */
if ( (srreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (srimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
srreal[i]=sample[i+isfstart];
srimag[i]=0.0;
}
farfree(sample);
for (i=0; i<512; i++){
fprintf(fout4, "%d  %lf\n", i, srreal[i]);
}
/* FFT source reflections */
myfft((double far *) srreal, (double far *) srimag, 512, -1);
/* Correct scaling */
for (i=0; i<512; i++){
srreal[i]=srreal[i]*sqrt(512.0);
srimag[i]=srimag[i]*sqrt(512.0);
}
/* Create all-pole model filter for tube length l1 */
/* a) Make first point = 1+0j */
if ( (apreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (apimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
apreal[0]=1.0;
apimag[0]=0.0;
stlength=3.102;
stradius=0.0048;
/* b) Calculate continous expression over a vector of 257 points */
/*    (point 0=0Hz, point 256=25000Hz */
for (i=1; i<257; i++){
om=2.0*M_PI*i*25000.0/256.0;
rv=stradius*sqrt(1.21*om/0.0000181);
v=sqrt(0.0000181*1400.0/0.024);
A=(1.0+(0.4/v))/sqrt(2.0);
B=1.0+(0.4/v)-0.5*(0.4/(v*v))-0.5*pow((0.4/v),2.0);
C=7.0/8.0+(0.4/v)-0.5*(0.4/(v*v));
C=C-(1.0/8.0)*(0.4/(pow(v,3.0)));
C=C-0.5*pow((0.4/v),2.0)+0.5*(pow(0.4,2.0)/pow(v,3.0));
C=C+0.5*pow((0.4/v),3.0);
C=C/sqrt(2.0);
D=A;
E=C;
alpha=(om/c)*((A/rv)+(B/pow(rv,2.0))+(C/pow(rv,3.0)));
vpinv=(1.0/c)*(1.0+(D/rv)+(E/pow(rv,3.0)));
apreal[i]=exp(-alpha*stlength)*cos(-om*vpinv*stlength);
apimag[i]=exp(-alpha*stlength)*sin(-om*vpinv*stlength);
}
/* c) Make point 256 entirely real by putting real part=magnitude */
apreal[256]=sqrt(pow(apreal[256],2.0)+pow(apimag[256],2.0));
apimag[256]=0.0;
/* d) Concatenate vector */
for (i=257; i<512; i++){
apreal[i]=apreal[512-i];
apimag[i]=-apimag[512-i];
}
/* e) Inverse FFT the continuous expression */
myfft((double far *) apreal, (double far *) apimag, 512, 1);
/* f) Correct scaling of FFT */
for (i=0; i<512; i++){
apreal[i]=apreal[i]/sqrt(512.0);
apimag[i]=apimag[i]/sqrt(512.0);
}
```

```
/* g) Calculate autocorrelation function */
if ( (autocor=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
autocor[i]=0.0;
for (y=0; y<512; y++){
if (y<(512-i)){
autocor[i]=autocor[i]+apreal[y]*apreal[y+i];
}
else{
autocor[i]=autocor[i]+apreal[y]*apreal[y+i-512];
}
}
}
/* h) Calculate a values */
if ( (e=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (k=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
e[0]=autocor[0];
k[1]=-autocor[1]/e[0];
a[1][1]=k[1];
e[1]=(1.0-pow(k[1],2.0))*e[0];
for (i=2; i<41; i++){
sum=0.0;
for (y=1; y<i; y++){
sum=sum+a[y][i-1]*autocor[i-y];
}
k[i]=-(autocor[i]+sum)/e[i-1];
a[i][i]=k[i];
for (y=1; y<i; y++){
a[y][i]=a[y][i-1]+k[i]*a[i-y][i-1];
}
e[i]=(1.0-pow(k[i],2.0))*e[i-1];
}
farfree(e);
farfree(k);
/* i) Calculate gain G */
G=autocor[0];
for (i=1; i<41; i++){
G=G+a[i][40]*autocor[i];
}
G=sqrt(G);
farfree(autocor);
/* j) Calculate A(z) (put in apreal, apimag) */
apreal[0]=1.0;
apimag[0]=0.0;
for(i=1; i<41; i++){
apreal[i]=a[i][40];
apimag[i]=0.0;
}
for(i=41; i<512; i++){
apreal[i]=0.0;
apimag[i]=0.0;
}
/* k) Fourier transform to calculate A(z) */
myfft((double far *) apreal, (double far *) apimag, 512, -1);
/* l) Correct scaling and calculate H(z) */
for (i=0; i<512; i++){
apreal[i]=apreal[i]*sqrt(512.0);
apimag[i]=apimag[i]*sqrt(512.0);
tempval=apreal[i];
apreal[i]=G*apreal[i]/(pow(apreal[i],2.0)+pow(apimag[i],2.0));
apimag[i]=-G*apimag[i]/(pow(tempval,2.0)+pow(apimag[i],2.0));
}
/* Need to deconvolve source reflections with all pole filter for */
/* tube length l1, to find what look like at exit of loudspeaker. */
if ( (sp2real=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (sp2imag=farmalloc(512*sizeof(double)) )==0){
```

```
perror("Malloc failed\n");
exit(0);
}
/* Complex division of source reflections and all pole filter */
for (i=0; i<512; i++){
sp2real[i]=srreal[i]*apreal[i]+srimag[i]*apimag[i];
sp2real[i]=sp2real[i]/(pow(apreal[i],2.0)+pow(apimag[i],2.0)+0.01);
sp2imag[i]=srimag[i]*apreal[i]-srreal[i]*apimag[i];
sp2imag[i]=sp2imag[i]/(pow(apreal[i],2.0)+pow(apimag[i],2.0)+0.01);
}
farfree(srreal);
farfree(srimag);
/* Inverse fft for plotting purposes */
myfft( (double far *) sp2real, (double far *) sp2imag, 512, 1);
for (i=0; i<512; i++){
fprintf(fout5, "%d  %lf\n", i, sp2real[i]/sqrt(512.0));
}
myfft( (double far *) sp2real, (double far *) sp2imag, 512, -1);
/* Need to convolve the object reflections with all-pole filter */
/* for tube length l1, to find out what look like at input of */
/* loudspeaker */
/* Multiply allpole filter with the isolated object reflections */
for (i=0; i<512; i++){
tempval=apreal[i];
apreal[i]=apreal[i]*orreal[i]-apimag[i]*orimag[i];
apimag[i]=apimag[i]*orreal[i]+tempval*orimag[i];
}
farfree(orreal);
farfree(orimag);
myfft((double far *) apreal, (double far *) apimag, 512, 1);
for (i=0; i<512; i++){
fprintf(fout3, "%d  %lf\n", i, apreal[i]/sqrt(512.0));
}
myfft((double far *) apreal, (double far *) apimag, 512, -1);
/* Deconvolve source reflections at exit of speaker with object */
/* reflections at input to speaker, to find the loudspeaker input */
/* impulse response */
if ( (lsrreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (lsrimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
/* Complex division */
for (i=0; i<512; i++){
lsrreal[i]=sp2real[i]*apreal[i]+sp2imag[i]*apimag[i];
lsrreal[i]=lsrreal[i]/(pow(apreal[i],2.0)+
pow(apimag[i],2.0)+100000.0);
lsrimag[i]=sp2imag[i]*apreal[i]-sp2real[i]*apimag[i];
lsrimag[i]=lsrimag[i]/(pow(apreal[i],2.0)+
pow(apimag[i],2.0)+100000.0);
}
farfree(apreal);
farfree(apimag);
for (i=0; i<512; i++){
fprintf(fout6, "%lf  %lf\n",lsrreal[i],lsrimag[i]);
}
farfree(lsrreal);
farfree(lsrimag);
farfree(sp2real);
farfree(sp2imag);
}
/************************* myhandler *****************************/
void _far _pascal
myhandler(int error_code)
{
printf("\nError! Aborted\nDaqBook/100 Error: 0x%x\n",error_code);
daqClose();
 exit(1);
}
/*****************************************************************/
/************************** myfft ********************************/
void myfft(double real[], double imag[],
unsigned long samp_num, int isign)
{
int i=0;
```

```
static double data[2*512+1];
for (i=1; i<(2*samp_num+1); i=i+2){
data[i]=real[(i-1)/2];
data[i+1]=imag[(i-1)/2];
}
dfour1(data, samp_num, isign);
for (i=1; i<(2*samp_num+1); i=i+2){
real[(i-1)/2]=data[i]/sqrt(samp_num);
imag[(i-1)/2]=data[i+1]/sqrt(samp_num);
}
}
/*************************************************************************/
/********************** dfour1 (Numerical Recipes) ****************/
void dfour1(double data[], unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
double tempr,tempi;
n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {
if (j > i) {
SWAP(data[j],data[i]);
SWAP(data[j+1],data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
j -= m;
m >>= 1;
}
j += m;
}
mmax=2;
while (n > mmax) {
istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);
wtemp=sin(0.5*theta);
wpr = -2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for (m=1;m<mmax;m+=2) {
for (i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*data[j]-wi*data[j+1];
tempi=wr*data[j+1]+wi*data[j];
data[j]=data[i]-tempr;
data[j+1]=data[i+1]-tempi;
data[i] += tempr;
data[i+1] += tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}
/*************************************************************************/
```

B.6 Program to find the relationship between the electrical input and the pressure output for the loudspeaker

```
/* PROGRAM TO FIND RELATIONSHIP BETWEEN ELECTRICAL INPUT AND PRESSURE */
/* OUTPUT FOR LOUDSPEAKER */
/* The D/A converter sends out voltages in the -5V to +5V range. */
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* The A/D converter samples voltages in the -5V to +5V range.*/
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/*Include library files */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
```

```
#include "c:\daqbook\dos\bc\daqbook.h"
#include <alloc.h>
#define SWAP(a,b) tempr=(a); (a)=(b); (b)=tempr
/* Declare functions */
void dfour1(double data[], unsigned long nn, int isign);
void myfft(double real[], double imag[],
unsigned long samp_num, int isign);
void _far _pascal myhandler(int error_code);
/* Main program */
void main(void)
{
FILE *fout1, *fout2, *fout3, *fout4;
int i, y, ipstart, iobstart;
double c, temp, obstart, om, rv, v, A, B, C, D, E, stlength,
stradius, alpha, vpinv, sum;
static char filenm1[80], filenm2[80], filenm3[80], filenm4[80];
static double a[41][41], G, tempval;
/* Set up arrays using pointers */
float huge *sample;
unsigned int huge *waveBuf;
unsigned int huge *buf;
float huge *bufav;
double huge *orreal;
double huge *orimag;
double huge *apreal;
double huge *apimag;
double huge *autocor;
double huge *e;
double huge *k;
double huge *sp1real;
double huge *sp1imag;
double huge *epreal;
double huge *epimag;
double huge *lspreal;
double huge *lspimag;
printf("PROGRAM TO FIND RELATIONSHIP BETWEEN ELECTRICAL INPUT\n");
printf("AND PRESSURE OUTPUT FOR LOUDSPEAKER \n");
/* Set error handler and initialize DaqBoard/100 */
daqSetErrHandler(myhandler);
daqInit(PORT_0300, DMA5+INTERRUPT10);
daqBrdSetDmaMode(DmaRead);
/* Read in filenames and open files */
printf("What do you wish to call the name of the ");
printf("file containing the full microphone output?\n");
scanf("%s", filenm1);
fout1=fopen(filenm1, "w");
printf("What do you wish to call the file containing the\n");
printf("reflection of the input pulse by the perspex?\n");
scanf("%s", filenm2);
fout2=fopen(filenm2, "w");
printf("What do you wish to call the file containing the\n");
printf("input pulse just after production by the loudpeaker?\n");
scanf("%s", filenm3);
fout3=fopen(filenm3, "w");
printf("What do you wish to call the file containing the\n");
printf("loudspeaker electrical production response?\n");
scanf("%s", filenm4);
fout4=fopen(filenm4, "w");
/* SAMPLING SECTION */
if( (waveBuf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (buf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (bufav=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
/* Setup waveforms for 20 uS update rate on DAC channel 0 only */
daqBrdDacSetMode(20, DacFIFOChan0, 0);
/* Build a user wave - i.e. a 80uS small voltage input pulse */
for (i=0; i<4; i++) {
waveBuf[i] = 2148;
}
for (i=4; i<4096; i++) {
```

```
waveBuf[i] = 2048;
}
/* Specify the user built waveform for DAC channel 0 */
daqBrdDacUserWave(0, (unsigned int far *)waveBuf, 4096);
for (y=0; y<4096; y++){
bufav[y]=0.0;
}
/* Sends pulse and samples result 1000 times */
/* (samples are then averaged) */
for (i=0; i<1000; i++){
/* Set output voltage equal to 0V */
daqDacWt(0,2048);
/* Delay before playing pulse to ensure signal died away */
delay(1000);
/* Set up the background sampling */
/* a) Set scan sequence - reads from channel at */
/*    DgainX1 */
daqAdcSetMux(0,0,DgainX1);
/* b) No tagged data */
daqAdcSetTag(0);
/* c) Set sampling frequency = 50kHz */
daqAdcSetFreq(50000);
/* d) Set to trigger from a software command */
daqAdcSetTrig(DtsSoftware,0,0,0,0);
/* Start background sampling. Sampled data read into an */
/* array called buf using DMA to send data to computer */
daqAdcRdNBack((unsigned int far *) buf, 4096, 0, DusDma);
daqAdcSoftTrig();
/* Start output waveform */
daqBrdDacStart();
/* Introduce a delay to ensure all output and input has */
/* stopped before the daqBoard is stopped */
delay(100);
/* Stop output waveform */
daqBrdDacStop();
for (y=0; y<4096; y++){
bufav[y]=bufav[y]+buf[y]/1000.0;
}
}
/* Close and exit */
daqClose();
farfree(waveBuf);
/* PROCESSING DATA SECTION */
/* Print averaged data to a file */
if ( (sample=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (y=0; y<4096; y++){
/* We use 2048-buf[], in order to invert signal (after */
/* inversion by amplifier) and to position around y=0 line. */
/* Need to divide by 16.0 because DMA is 16bit */
sample[y]=2048-bufav[y]/16.0;
}
for (y=0; y<4096; y++){
fprintf(fout1, "%d %f\n", y, sample[y]);
}
farfree(buf);
farfree(bufav);
/* Find where input sound pulse passes microphone (define */
/* starting point of input pulse as two samples before the level */
/* rises above 100 - out of a possible 2048) */
for (y=0; y<4096; y++){
if (sample[y]>100){
ipstart=y-2;
break;
}
}
printf("Input sound pulse first apparent at %d\n", ipstart);
/* Calculate where object reflections (i.e. reflection of input */
/* pulse by perspex) should start */
/* a) Calculate speed of sound from temperature */
printf("What is the temperature in degrees C?\n");
scanf("%lf", &temp);
c=331.6*(sqrt((temp+273.0)/273.0));
/* b) Calculate the delay before object reflections arrive */
/*    Length of tube = 2*l2 = 6.18m */
obstart=ipstart+6.18/(0.00002*c);
```

```
/* c) Round up or down to nearest integer */
iobstart=obstart;
if ((obstart-iobstart)>=0.5){
iobstart++;
}
printf("Object reflections first apparent at %d\n", iobstart);
/* CALCULATE LOUDSPEAKER PRODUCTION RESPONSE */
/* Isolate object reflections - put into a 512 array */
/* ready for FFTing */
if ( (orreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (orimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
orreal[i]=sample[iobstart+i];
orimag[i]=0;
}
for (i=0; i<512; i++){
fprintf(fout2, "%d %lf\n", i, orreal[i]);
}
/* FFT the array of object reflections */
myfft((double far *) orreal, (double far *) orimag, 512, -1);
for (i=0; i<512; i++){
orreal[i]=orreal[i]*sqrt(512.0);
orimag[i]=orimag[i]*sqrt(512.0);
}
/* Create all-pole model filter for tube length l1+2*l2 */
/* a) Make first point = 1+0j */
if ( (apreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (apimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
apreal[0]=1.0;
apimag[0]=0.0;
stlength=3.102+2.0*3.09;
stradius=0.0048;
/* b) Calculate continous expression on a vector of 257 points */
/*    (point 0=0Hz, point 256=25000Hz */
for (i=1; i<257; i++){
om=2.0*M_PI*i*25000.0/256.0;
rv=stradius*sqrt(1.21*om/0.0000181);
v=sqrt(0.0000181*1400.0/0.024);
A=(1.0+(0.4/v))/sqrt(2.0);
B=1.0+(0.4/v)-0.5*(0.4/(v*v))-0.5*pow((0.4/v),2.0);
C=7.0/8.0+(0.4/v)-0.5*(0.4/(v*v));
C=C-(1.0/8.0)*(0.4/(pow(v,3.0)));
C=C-0.5*pow((0.4/v),2.0)+0.5*(pow(0.4,2.0)/pow(v,3.0));
C=C+0.5*pow((0.4/v),3.0);
C=C/sqrt(2.0);
D=A;
E=C;
alpha=(om/c)*((A/rv)+(B/pow(rv,2.0))+(C/pow(rv,3.0)));
vpinv=(1.0/c)*(1.0+(D/rv)+(E/pow(rv,3.0)));
apreal[i]=exp(-alpha*stlength)*cos(-om*vpinv*stlength);
apimag[i]=exp(-alpha*stlength)*sin(-om*vpinv*stlength);
}
/* c) Make point 256 real by putting real part=magnitude */
apreal[256]=sqrt(pow(apreal[256],2.0)+pow(apimag[256],2.0));
apimag[256]=0.0;
/* d) Concatenate vector */
for (i=257; i<512; i++){
apreal[i]=apreal[512-i];
apimag[i]=-apimag[512-i];
}
/* e) Inverse FFT the continuous expression */
myfft((double far *) apreal, (double far *) apimag, 512, 1);
/* f) Correct scaling of FFT */
for (i=0; i<512; i++){
apreal[i]=apreal[i]/sqrt(512.0);
apimag[i]=apimag[i]/sqrt(512.0);
```

```
}
/* g) Calculate autocorrelation function */
if ( (autocor=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
autocor[i]=0.0;
for (y=0; y<512; y++){
if (y<(512-i)){
autocor[i]=autocor[i]+apreal[y]*apreal[y+i];
}
else{
autocor[i]=autocor[i]+apreal[y]*apreal[y+i-512];
}
}
}
/* h) Calculate a values */
if ( (e=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (k=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
e[0]=autocor[0];
k[1]=-autocor[1]/e[0];
a[1][1]=k[1];
e[1]=(1.0-pow(k[1],2.0))*e[0];
for (i=2; i<41; i++){
sum=0.0;
for (y=1; y<i; y++){
sum=sum+a[y][i-1]*autocor[i-y];
}
k[i]=-(autocor[i]+sum)/e[i-1];
a[i][i]=k[i];
for (y=1; y<i; y++){
a[y][i]=a[y][i-1]+k[i]*a[i-y][i-1];
}
e[i]=(1.0-pow(k[i],2.0))*e[i-1];
}
farfree(e);
farfree(k);
/* i) Calculate gain G */
G=autocor[0];
for (i=1; i<41; i++){
G=G+a[i][40]*autocor[i];
}
G=sqrt(G);
farfree(autocor);
/* j) Calculate A(z) (put in apreal, apimag) */
apreal[0]=1.0;
apimag[0]=0.0;
for(i=1; i<41; i++){
apreal[i]=a[i][40];
apimag[i]=0.0;
}
for(i=41; i<512; i++){
apreal[i]=0.0;
apimag[i]=0.0;
}
/* k) Fourier transform to calculate A(z) */
myfft((double far *) apreal, (double far *) apimag, 512, -1);
/* l) Correct scaling and calculate H(z) */
for (i=0; i<512; i++){
apreal[i]=apreal[i]*sqrt(512.0);
apimag[i]=apimag[i]*sqrt(512.0);
tempval=apreal[i];
apreal[i]=G*apreal[i]/(pow(apreal[i],2.0)+pow(apimag[i],2.0));
apimag[i]=-G*apimag[i]/(pow(tempval,2.0)+pow(apimag[i],2.0));
}
/* Deconvolve the FFTed object reflections with the calculated */
/* all pole filter */
if ( (sp1real=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed here\n");
exit(0);
}
```

```
if ( (sp1imag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed or here\n");
exit(0);
}
/* Complex division */
for (i=0; i<512; i++){
sp1real[i]=orreal[i]*apreal[i]+orimag[i]*apimag[i];
sp1real[i]=sp1real[i]/(pow(apreal[i],2.0)+
pow(apimag[i],2.0)+0.00001);
sp1imag[i]=orimag[i]*apreal[i]-orreal[i]*apimag[i];
sp1imag[i]=sp1imag[i]/(pow(apreal[i],2.0)+
pow(apimag[i],2.0)+0.00001);
}
farfree(apreal);
farfree(apimag);
/* Inverse transform for plotting purposes*/
myfft( (double far *)sp1real, (double far *)sp1imag, 512, 1);
for (i=0; i<512; i++){
fprintf(fout3, "%d  %lf\n", i, sp1real[i]/sqrt(512.0));
}
/* Forward transform to return to position before plotting */
myfft( (double far *) sp1real, (double far *)sp1imag, 512, -1);
/* Create electrical pulse array (rate of change of voltage) */
if ( (epreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (epimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
epreal[i]=0.0;
epimag[i]=0.0;
}
epreal[0]=100.0;
epreal[4]=-100.0;
/* FFT electrical pulse */
myfft( (double far *)epreal, (double far *)epimag, 512, -1);
/* Correct scaling */
for (i=0; i<512; i++){
epreal[i]=epreal[i]*sqrt(512.0);
epimag[i]=epimag[i]*sqrt(512.0);
}
/* Deconvolve sound pulse at loudspeaker with electrical pulse */
if ( (lspreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (lspimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
/*Complex division*/
for (i=0; i<512; i++){
lspreal[i]=sp1real[i]*epreal[i]+sp1imag[i]*epimag[i];
lspreal[i]=lspreal[i]/(pow(epreal[i],2.0)+
pow(epimag[i],2.0)+0.000000000001);
lspimag[i]=sp1imag[i]*epreal[i]-sp1real[i]*epimag[i];
lspimag[i]=lspimag[i]/(pow(epreal[i],2.0)+
pow(epimag[i],2.0)+0.000000000001);
}
farfree(sp1real);
farfree(sp1imag);
farfree(epreal);
farfree(epimag);
for (i=0; i<512; i++){
fprintf(fout4, "%lf  %lf\n",lspreal[i],lspimag[i]);
}
farfree(lspreal);
farfree(lspimag);
}
/************************* myhandler *****************************/
void _far _pascal
myhandler(int error_code)
{
printf("\nError! Aborted\nDaqBook/100 Error: 0x%x\n",error_code);
daqClose();
```

```
exit(1);
}
/*********************************************************************/
/************************** myfft ********************************/
void myfft(double real[], double imag[],
unsigned long samp_num, int isign)
{
int i=0;
static double data[2*512+1];
for (i=1; i<(2*samp_num+1); i=i+2){
data[i]=real[(i-1)/2];
data[i+1]=imag[(i-1)/2];
}
dfour1(data, samp_num, isign);
for (i=1; i<(2*samp_num+1); i=i+2){
real[(i-1)/2]=data[i]/sqrt(samp_num);
imag[(i-1)/2]=data[i+1]/sqrt(samp_num);
}
}
/*********************************************************************/
/********************** dfour1 (Numerical Recipes ****************/
void dfour1(double data[], unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
double tempr,tempi;
n=nn << 1;
j=1;
for (i=1;i<n;i+=2) {
if (j > i) {
SWAP(data[j],data[i]);
SWAP(data[j+1],data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
j -= m;
m >>= 1;
}
j += m;
}
mmax=2;
while (n > mmax) {
istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);
wtemp=sin(0.5*theta);
wpr = -2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for (m=1;m<mmax;m+=2) {
for (i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*data[j]-wi*data[j+1];
tempi=wr*data[j+1]+wi*data[j];
data[j]=data[i]-tempr;
data[j+1]=data[i+1]-tempi;
data[i] += tempr;
data[i+1] += tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}
/*********************************************************************/
```

B.7 Data acquisition program (employing absorbing termination method)

```
/* DATA ACQUISITION PROGRAM EMPLOYING ABSORBING TERMINATION METHOD */
/* The D/A converter sends out voltages in the -5V to +5V range. */
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
/* The A/D converter samples voltages in the -5V to +5V range.*/
/* (thus 0=-5V, 2048=0V and 4095=+5V) */
```

```
/* Include library files */
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "c:\daqbook\dos\bc\daqbook.h"
#include <alloc.h>
#define SWAP(a,b) tempr=(a); (a)=(b); (b)=tempr
/* Declare functions */
void dfour1(double data[], unsigned long nn, int isign);
void myfft(double real[], double imag[],
unsigned long samp_num, int isign);
void _far _pascal myhandler(int error_code);
/* Main program */
void main(void)
{
FILE *fin1, *fin2, *fout1, *fout2, *fout3, *fout4, *fout5,
*fout6, *fout7;
int i, y, ipstart, iobstart, idel, loop;
double temp, c, obstart, del, stlength, stradius, om, rv, v, A, B,
C, D, E, alpha, vpinv, sum, G, tempval, fincurr;
static double a[41][41];
unsigned int chans[2];
unsigned char gains[2];
static char filenm1a[80], filenm2a[80], filenm1[80], filenm2[80],
filenm3[80], filenm4[80], filenm5[80], filenm6[80],
filenm7[80];
/* Set up arrays using pointers */
unsigned int huge *waveBuf;
unsigned int huge *buf;
float huge *sample;
float huge *bufav;
double huge *orreal;
double huge *orimag;
double huge *apreal;
double huge *apimag;
double huge *autocor;
double huge *e;
double huge *k;
double huge *lsrreal;
double huge *lsrimag;
double huge *lspreal;
double huge *lspimag;
double huge *orretemp;
printf("DATA ACQUISITION PROGRAM WHICH EMPLOYS ABSORBING\n");
printf("TERMINATION METHOD\n");
printf("(program calculates the noise cancellation signal\n");
printf("and plays it out)\n\n");
/* Set error handler and initialize DaqBoard/100 */
daqSetErrHandler(myhandler);
daqInit(PORT_0300, DMA5+INTERRUPT10);
daqBrdSetDmaMode(DmaRead);
/* Read in loudspeaker input impulse response */
printf("What is the name of the file containing the\n");
printf("loudspeaker input impulse response?\n");
scanf("%s", filenm2a);
if ( (fin2=fopen(filenm2a, "r"))==0){
printf("Unable to open\n");
return;
}
if ( (lsrreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (lsrimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
fscanf(fin2, "%lf  %lf", &lsrreal[i], &lsrimag[i]);
}
/* Read in loudspeaker electrical production response */
printf("What is the name of the file containing the\n");
printf("loudspeaker electrical production response?\n");
scanf("%s", filenm1a);
if ( (fin1=fopen(filenm1a, "r"))==0){
printf("Unable to open\n");
return;
```

```
}
if ( (lspreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (lspimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
fscanf(fin1, "%lf  %lf", &lspreal[i], &lspimag[i]);
}
/* Calculate speed of sound from temperature */
printf("What is the temperature in degrees C?\n");
scanf("%lf", &temp);
c=331.6*(sqrt((temp+273.0)/273.0));
/* Calculate the all-pole model filter for tube length l1 */
/* a) Make first point = 1+0j */
if ( (apreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (apimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
apreal[0]=1.0;
apimag[0]=0.0;
stlength=3.102;
stradius=0.0048;
/* b) Calculate continous expression over a vector of 257 points */
/*    (point 0=0Hz, point 256=25000Hz */
for (i=1; i<257; i++){
om=2.0*M_PI*i*25000.0/256.0;
rv=stradius*sqrt(1.21*om/0.0000181);
v=sqrt(0.0000181*1400.0/0.024);
A=(1.0+(0.4/v))/sqrt(2.0);
B=1.0+(0.4/v)-0.5*(0.4/(v*v))-0.5*pow((0.4/v),2.0);
C=7.0/8.0+(0.4/v)-0.5*(0.4/(v*v));
C=C-(1.0/8.0)*(0.4/(pow(v,3.0)));
C=C-0.5*pow((0.4/v),2.0)+0.5*(pow(0.4,2.0)/pow(v,3.0));
C=C+0.5*pow((0.4/v),3.0);
C=C/sqrt(2.0);
D=A;
E=C;
alpha=(om/c)*((A/rv)+(B/pow(rv,2.0))+(C/pow(rv,3.0)));
vpinv=(1.0/c)*(1.0+(D/rv)+(E/pow(rv,3.0)));
apreal[i]=exp(-alpha*stlength)*cos(-om*vpinv*stlength);
apimag[i]=exp(-alpha*stlength)*sin(-om*vpinv*stlength);
}
/* c) Make point 256 entirely real by putting real part=magnitude */
apreal[256]=sqrt(pow(apreal[256],2.0)+pow(apimag[256],2.0));
apimag[256]=0.0;
/* d) Concatenate vector */
for (i=257; i<512; i++){
apreal[i]=apreal[512-i];
apimag[i]=-apimag[512-i];
}
/* e) Inverse FFT the continuous expression */
myfft((double far *) apreal, (double far *) apimag, 512, 1);
/* f) Correct scaling of FFT */
for (i=0; i<512; i++){
apreal[i]=apreal[i]/sqrt(512.0);
apimag[i]=apimag[i]/sqrt(512.0);
}
/* g) Calculate autocorrelation function */
if ( (autocor=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
autocor[i]=0.0;
for (y=0; y<512; y++){
if (y<(512-i)){
autocor[i]=autocor[i]+apreal[y]*apreal[y+i];
}
else{
autocor[i]=autocor[i]+apreal[y]*apreal[y+i-512];
```

```
}
}
}
/* h) Calculate a values */
if ( (e=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (k=farmalloc(41*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
e[0]=autocor[0];
k[1]=-autocor[1]/e[0];
a[1][1]=k[1];
e[1]=(1.0-pow(k[1],2.0))*e[0];
for (i=2; i<41; i++){
sum=0.0;
for (y=1; y<i; y++){
sum=sum+a[y][i-1]*autocor[i-y];
}
k[i]=-(autocor[i]+sum)/e[i-1];
a[i][i]=k[i];
for (y=1; y<i; y++){
a[y][i]=a[y][i-1]+k[i]*a[i-y][i-1];
}
e[i]=(1.0-pow(k[i],2.0))*e[i-1];
}
farfree(e);
farfree(k);
/* i) Calculate gain G */
G=autocor[0];
for (i=1; i<41; i++){
G=G+a[i][40]*autocor[i];
}
G=sqrt(G);
farfree(autocor);
/* j) Calculate A(z) (put in apreal, apimag) */
apreal[0]=1.0;
apimag[0]=0.0;
for(i=1; i<41; i++){
apreal[i]=a[i][40];
apimag[i]=0.0;
}
for(i=41; i<512; i++){
apreal[i]=0.0;
apimag[i]=0.0;
}
/* k) Fourier transform to calculate A(z) */
myfft((double far *) apreal, (double far *) apimag, 512, -1);
/* l) Correct scaling and calculate H(z) */
for (i=0; i<512; i++){
apreal[i]=apreal[i]*sqrt(512.0);
apimag[i]=apimag[i]*sqrt(512.0);
tempval=apreal[i];
apreal[i]=G*apreal[i]/(pow(apreal[i],2.0)+pow(apimag[i],2.0));
apimag[i]=-G*apimag[i]/(pow(tempval,2.0)+pow(apimag[i],2.0));
}
/* Read in names of files to write to */
printf("What do you wish to call the file containing\n");
printf("the full microphone output with no cancellation?\n");
scanf("%s", filenm1);
fout1=fopen(filenm1, "w");
printf("What do you wish to call the file containing\n");
printf("the first revised sample?\n");
scanf("%s", filenm2);
fout2=fopen(filenm2, "w");
printf("What do you wish to call the file containing\n");
printf("the second revised sample?\n");
scanf("%s", filenm3);
fout3=fopen(filenm3, "w");
printf("What do you wish to call the file containing\n");
printf("the third revised sample?\n");
scanf("%s", filenm4);
fout4=fopen(filenm4, "w");
printf("What do you wish to call the file containing\n");
printf("the fourth revised sample?\n");
scanf("%s", filenm5);
```

```
fout5=fopen(filenm5, "w");
printf("What do you wish to call the file containing\n");
printf("the reflections after cancellation?\n");
scanf("%s", filenm6);
fout6=fopen(filenm6, "w");
printf("What do you wish to call the file containing\n");
printf("the uncancelled reflections?\n");
scanf("%s", filenm7);
fout7=fopen(filenm7, "w");
/* Play unmodified pulse and obtain unrevised sample */
if ( (waveBuf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (buf=farmalloc(4096*sizeof(int)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (bufav=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
/* Set up waveform for 20 uS update rate on DAC channel 0 only */
daqBrdDacSetMode(20,DacFIFOChan0, 0);
/* Build a user wave - i.e. a 5V input pulse of 80uS length */
for (i=0; i<4; i++){
waveBuf[i]=4095;
}
for (i=4; i<4096; i++){
waveBuf[i]=2048;
}
/* Specify the user built waveform for DAC channel 0 */
daqBrdDacUserWave(0, (unsigned int far *)waveBuf, 4096);
for (y=0; y<4096; y++){
bufav[y]=0.0;
}
/* Sends out pulse and samples result 1000 times */
/* (samples are then averaged) */
for (i=0; i<1000; i++){
/* Set output voltage equal to 0V */
daqDacWt(0,2048);
/* Delay before playing pulse to ensure signal died away */
delay(1000);
/* Set up background sampling */
/* a) Reads from channel 0 at DgainX1 */
daqAdcSetMux(0,0,DgainX1);
/* b) No tagged data */
daqAdcSetTag(0);
/* c) Set sampling frequency=50kHz */
daqAdcSetFreq(50000);
/* d)Software trigger */
daqAdcSetTrig(DtsSoftware,0,0,0,0);
/* Start background sampling.Reads into an array called */
/* buf (DMA used to send data to computer) */
daqAdcRdNBack((unsigned int far *) buf, 4096, 0 ,DusDma);
daqAdcSoftTrig();
/* Start output waveform */
daqBrdDacStart();
/* Introduce a delay to ensure all output and input has */
/* stopped before the daqBoard is stopped */
delay(100);
/* Stop output waveform */
daqBrdDacStop();
for(y=0; y<4096; y++){
bufav[y]=bufav[y]+buf[y]/1000.0;
}
}
/* Processing data section */
/* Print averaged data to file */
if ( (sample=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<4096; i++){
/* We use 2048-buf[], in order to invert signal (after */
/* inversion by amplifier) and to position around y=0 line. */
sample[i]=2048-bufav[i]/16.0;
}
```

```
farfree(bufav);
for (i=0; i<4096; i++){
fprintf(fout1, "%d %f\n", i, sample[i]);
}
/* Find where input sound pulse passes microphone (define */
/* starting point of input pulse as two sample before the level */
/*  rises above 100 out of a possible 2048) */
for (i=0; i<4096; i++){
if (sample[i]>100){
ipstart=i-2;
break;
}
}
printf("Input sound pulse first apparent at %d\n", ipstart);
/* a) Calculate the delay before object reflections arrive */
/*    (length of tube=2*l2=6.18m) */
obstart=ipstart+6.18/(0.00002*c);
/* b) Round up or down to nearest integer */
iobstart=obstart;
if ((obstart-iobstart)>=0.5){
iobstart++;
}
printf("Object reflections first apparent at %d\n", iobstart);
/* c) Calculate delay between elec input pulse and elec noise */
/*    canc (time taken to travel length 2*(l1+l2) = 6.192m) */
del=2.0*(6.192/(c*0.00002));
/* d) Round up or down to nearest integer */
idel=del;
if ((del-idel)>=0.5){
idel++;
}
printf("Delay=%d\n", idel);
for (i=0; i<2048; i++){
fprintf(fout7, "%lf  %lf\n", i*0.00002,
(5.0/2048)*sample[i+(iobstart-250)]);
}
fincurr=0;
idel=idel-9;
/* Build up cancellation signal in blocks */
for(loop=0; loop<4; loop++){
/* Isolate object reflections - put into a 512 array */
/* ready for FFTing */
if ( (orreal=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
if ( (orimag=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<512; i++){
orreal[i]=sample[iobstart+i+(loop*512)];
orimag[i]=0;
}
farfree(sample);
/* FFT the array of object reflections */
myfft((double far *) orreal, (double far *) orimag, 512, -1);
for (i=0; i<512; i++){
orreal[i]=orreal[i]*sqrt(512.0);
orimag[i]=orimag[i]*sqrt(512.0);
}
/* Convolve object reflections with allpole filter */
for (i=0; i<512; i++){
tempval=orreal[i];
orreal[i]=orreal[i]*apreal[i]-orimag[i]*apimag[i];
orimag[i]=orimag[i]*apreal[i]+tempval*apimag[i];
}
/* Convolve with loudspeaker input impulse response to find */
/* signal at exit of loudspeaker */
for (i=0; i<512; i++){
tempval=orreal[i];
orreal[i]=orreal[i]*lsrreal[i]-orimag[i]*lsrimag[i];
orimag[i]=orimag[i]*lsrreal[i]+tempval*lsrimag[i];
}
/* Deconvolve with loudspeaker electrical production response */
/* to find the elec signal required to reproduce waveform */
for (i=0; i<512; i++){
tempval=orreal[i];
```

```
orreal[i]=orreal[i]*lspreal[i]+orimag[i]*lspimag[i];
orreal[i]=orreal[i]/(pow(lspreal[i],2.0)
+pow(lspimag[i],2.0)+0.0000000000001);
orimag[i]=orimag[i]*lspreal[i]-tempval*lspimag[i];
orimag[i]=orimag[i]/(pow(lspreal[i],2.0)
+pow(lspimag[i],2.0)+0.0000000000001);
}
/* Inverse FFT and print out */
myfft( (double far *) orreal, (double far *) orimag, 512, 1);
for (i=0; i<512; i++){
orreal[i]=orreal[i]/sqrt(512.0);
}
/* Cancellation signal is 'wrapped around' need to shift */
/* last 7 samples of orreal back to the start */
if ( (orretemp=farmalloc(512*sizeof(double)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<7; i++){
orretemp[i]=orreal[505+i];
}
for (i=7; i<512; i++){
orretemp[i]=orreal[i-7];
}
for (i=0; i<512; i++){
orreal[i]=orretemp[i];
}
farfree(orretemp);
/* Integrate rate of change of voltage signal */
orreal[0]=fincurr;
for (i=1; i<512; i++){
orreal[i]=orreal[i]+orreal[i-1];
}
fincurr=orreal[511];
if ( (bufav=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
/* Set up waveform for 20 uS update rate on DAC channel 0 */
daqBrdDacSetMode(20,DacFIFOChan0, 0);
/* Build a user wave (add cancellation signal ) */
for (i=idel+(loop*512); i<(idel+(loop+1)*512); i++){
waveBuf[i]=(-(int)orreal[i-(idel+(loop*512))])+2048;
}
farfree(orreal);
farfree(orimag);
/* Specify the user built waveform for DAC channel 0 */
daqBrdDacUserWave(0, (unsigned int far *)waveBuf, 4096);
for (y=0; y<4096; y++){
bufav[y]=0.0;
}
for (i=0; i<1000; i++){
daqDacWt(0,2048);
delay(1000);
/* Set up background sampling */
/* a) Reads from channel 0 at DgainX1 */
daqAdcSetMux(0,0,DgainX1);
/* b) No tagged data */
daqAdcSetTag(0);
/* c) Set sampling frequency=50kHz */
daqAdcSetFreq(50000);
/* d)Software trigger */
daqAdcSetTrig(DtsSoftware,0,0,0,0);
/* Start background sampling. Reads into an array */
/* called buf (data transferred to computer using DMA) */
daqAdcRdNBack((unsigned int far *) buf, 4096, 0, DusDma);
daqAdcSoftTrig();
/* Start output waveform */
daqBrdDacStart();
/* Introduce a delay to ensure all output and input has */
/* stopped before the daqBoard is stopped */
delay(100);
/* Stop output waveform */
daqBrdDacStop();
for(y=0; y<4096; y++){
bufav[y]=bufav[y]+buf[y]/1000.0;
}
}
```

```
/* Processing data section */
/* Print channel 0 data to file */
if ( (sample=farmalloc(4096*sizeof(float)) )==0){
perror("Malloc failed\n");
exit(0);
}
for (i=0; i<4096; i++){
/* We use 2048-buf[], in order to invert signal */
/* (after inversion by amplifier) and to position */
/* around y=0 line. */
sample[i]=2048-bufav[i]/16.0;
}
farfree(bufav);
for(i=0; i<4096; i++){
if (loop==0){
fprintf(fout2, "%d  %lf\n", i, sample[i]);
}
if (loop==1){
fprintf(fout3, "%d  %lf\n", i, sample[i]);
}
if (loop==2){
fprintf(fout4, "%d  %lf\n", i, sample[i]);
}
if (loop==3){
fprintf(fout5, "%d  %lf\n", i, sample[i]);
}
}
}
farfree(apreal);
farfree(apimag);
farfree(lsrreal);
farfree(lsrimag);
farfree(lspreal);
farfree(lspimag);
farfree(waveBuf);
/* Print cancelled reflections data to file as voltage */
/* versus time in milliseconds */
for (i=0; i<2048; i++){
fprintf(fout6, "%lf  %lf\n", i*0.02,
(5.0/2048)*sample[i+(iobstart-250)]);
}
farfree(buf);
farfree(sample);
daqClose();
}
/************************* myhandler *****************************/
void _far _pascal
myhandler(int error_code)
{
printf("\nError! Aborted\nDaqBook/100 Error: 0x%x\n",error_code);
daqClose();
 exit(1);
}
/*****************************************************************/
/*************************** myfft *******************************/
void myfft(double real[], double imag[],
unsigned long samp_num, int isign)
{
int i=0;
static double data[2*512+1];
for (i=1; i<(2*samp_num+1); i=i+2){
data[i]=real[(i-1)/2];
data[i+1]=imag[(i-1)/2];
}
dfour1(data, samp_num, isign);
for (i=1; i<(2*samp_num+1); i=i+2){
real[(i-1)/2]=data[i]/sqrt(samp_num);
imag[(i-1)/2]=data[i+1]/sqrt(samp_num);
}
}
/*****************************************************************/
/********************** dfour1 (Numerical Recipes) ***************/
void dfour1(double data[], unsigned long nn, int isign)
{
unsigned long n,mmax,m,j,istep,i;
double wtemp,wr,wpr,wpi,wi,theta;
double tempr,tempi;
n=nn << 1;
```

```
j=1;
for (i=1;i<n;i+=2) {
if (j > i) {
SWAP(data[j],data[i]);
SWAP(data[j+1],data[i+1]);
}
m=n >> 1;
while (m >= 2 && j > m) {
j -= m;
m >>= 1;
}
j += m;
}
mmax=2;
while (n > mmax) {
istep=mmax << 1;
theta=isign*(6.28318530717959/mmax);
wtemp=sin(0.5*theta);
wpr = -2.0*wtemp*wtemp;
wpi=sin(theta);
wr=1.0;
wi=0.0;
for (m=1;m<mmax;m+=2) {
for (i=m;i<=n;i+=istep) {
j=i+mmax;
tempr=wr*data[j]-wi*data[j+1];
tempi=wr*data[j+1]+wi*data[j];
data[j]=data[i]-tempr;
data[j+1]=data[i+1]-tempi;
data[i] += tempr;
data[i+1] += tempi;
}
wr=(wtemp=wr)*wpr-wi*wpi+wr;
wi=wi*wpr+wtemp*wpi+wi;
}
mmax=istep;
}
}
/*************************************************************************/
```

Bibliography

[Agullo et al 1995] Agullo J, Cardona S and Keefe DH. *Time domain deconvolution to measure reflection functions for discontinuities in waveguides.* J.Acoust.Soc.Am. (1995), 97(3), 1950-1957.

[Amir 1995a] Amir N. *Personal correspondence.*

[Amir et al 1995b] Amir N, Rosenhouse G and Shimony U. *A discrete model for tubular acoustic systems with varying cross section - The direct and inverse problems. Parts 1 and 2: Theory and experiment.* Acustica(1995), 81(5), 450-474.

[Amir et al 1996] Amir N, Rosenhouse G and Shimony U. *Losses in tubular acoustic systems - Theory and experiment in the sampled time and frequency domains.* Acustica - Acta Acustica(1996), 82(1), 1-8.

[Ayers et al 1985a] Ayers RD, Eliason LJ and Salem MMB. *An acoustic pulse generator for wind instrument bores.* J.Acous.Soc.Am.(1985), 77, S89.

[Ayers et al 1985b] Ayers RD, Eliason LJ and Salem MMB. *Multiple reflections in simple bore shapes.* J.Acoust.Soc.Am. (1985), 77, S90.

[Backus 1974] Backus J. *Input impedance curves for the reed woodwind instruments.* J.Acoust.Soc.Am.(1974), 56(4), 1266-1279.

[Backus 1975] Backus J. *Acoustic impedance of an annular capillary.* J.Acoust.Soc.Am.(1975), 58(5), 1078-1081.

[Backus 1976] Backus J. *Input impedance curves for the brass instruments.* J.Acoust.Soc.Am.(1976), 60(2), 470-480.

[Benade and Murday 1967] Benade AH and Murday J. *Measured end corrections for woodwind toneholes.* J.Acoust.Soc.Am. (1967), 41, 1609-1610.

[Benade 1968] Benade AH. *On the propagation of sound waves in a cylindrical conduit.* J.Acoust.Soc.Am.(1968), 44(2), 616-623.

[Benade and Jansson 1974] Benade AH and Jansson EV. *On plane and spherical waves in horns with non-uniform flare. I.Theory of radiation, resonance frequencies and mode conversion.* Acustica(1974), 31, 79-98.

[Benade and Smith 1981] Benade AH and Smith JH. *Brass wind instrument impulse response measurements.* J.Acoust.Soc.Am. (1981) 70, S22.

[Brooks et al 1984] Brooks LJ, Castile RG, Glass GM, Griscom NT, Wohl MEB and Fredberg JJ. *Reproducibility and*

accuracy of airway area by acoustic reflection. J.Appl.Physiol.(1984), 57(3), 777-787.

[Bruckstein et al 1985] Bruckstein AM, Levy BC and Kailath T. *Differential methods in inverse scattering.* SIAM J.Appl.Math.(1985), 45(2), 312-335.

[Causse et al 1984] Causse R, Kergomard J and Lurton X. *Input impedance of brass musical instruments - Comparison between experiment and numerical models.* J.Acoust.Soc.Am.(1984), 75(1), 241-254.

[Deane 1986] Deane AM. *Time domain work on brass instruments.* PhD thesis(1986), University of Surrey.

[Duffield 1984] Duffield A. *Problems encountered when making simple impulse measurements.* Proceedings of the Institute Of Acoustics(1984), 21-28.

[Fredberg et al 1980] Fredberg JJ, Wohl MEB, Glass GM and Dorkin HL. *Airway area by acoustic reflections measured at the mouth.* J.Appl.Physiol.(1980), 48(5), 749-758.

[Gazengel et al 1995] Gazengel B, Gilbert J and Amir N. *Time domain simulation of single reed wind instrument. From the measured input impedance to the synthesis signal. Where are the traps?* Acta Acustica(1995), 3(5), 445-472.

[Goodwin 1981] Goodwin JC. *Relations between the geometry and acoustics of brass instruments.* PhD thesis(1981), University of Surrey.

[Hall 1987] Hall DE. *Basic acoustics.* 1st edition(1987), John Wiley and sons, New York.

[Hoffstein et al 1987] Hoffstein V, Castile RG, O'Donnell CR, Glass GM, Strieder DJ, Wohl MEB and Fredberg JJ. *In vivo estimation of tracheal distensibility and hysteresis in normal adults.* J.Appl.Physiol.(1987), 63(6), 2482-2489.

[Jackson et al 1977] Jackson AC, Butler JP, Millet EJ, Hoppin FG and Dawson SV. *Airway geometry by analysis of acoustic pulse response measurements.* J.Appl.Physiol.(1977), 43(3), 523-536.

[Jackson and Olsen 1980] Jackson AC and Olsen DE. *Comparison of direct and acoustical area measurements in physical models of human central airways.* J.Appl.Physiol. (1980), 48(5), 896-902.

[Jansson and Benade 1974] Jansson EV and Benade AH. *On plane and spherical waves in horns with non-uniform flare. II.Prediction and measurements of resonance frequencies and radiation losses.* Acustica(1974), 31, 185-202.

[Keefe 1982] Keefe DH. *Experiments on the single woodwind tonehole.* J.Acoust.Soc.Am.(1982), 72(3), 688-699.

[Keefe 1984] Keefe DH. *Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions.* J.Acoust.Soc.Am.(1984), 75(1), 58-62.

[Keefe 1996] Keefe DH. *Wind instrument reflection function measurements in the time domain.* J.Acoust.Soc. Am.(1995), 99(4), Pt.1, 2370-2381.

[Kinsler et al 1982] Kinsler LE, Frey AR, Coppens AB and Sanders JV. *Fundamentals of acoustics.* 3rd edition(1982), John Wiley and sons, New York.

[Louis et al 1993] Louis B, Glass G, Kresen B, Fredberg J. *Airway area by acoustic reflection: The two-microphone method.* J.Biomech.Eng.(1993), 115, 278-285.

[Makhoul 1975] Makhoul J. *Linear prediction: A tutorial review.* Proceedings of the IEEE(1975), 63(4), 561-580.

[Marshall 1990] Marshall I. *The production of acoustic impulses in air.* Meas.Sci.Technol.(1990), 1, 413-418.

[Marshall et al 1991] Marshall I, Rogers M and Drummond G. *Acoustic reflectometry for airway measurement. Principles, limitations and previous work.* Clin.Phys.Physiol. Meas.(1991), 12(2), 131-141.

[Marshall 1992a] Marshall I. *Acoustic reflectometry with an arbitrarily short source tube.* J.Acoust.Soc.Am.(1992), 91(6), 3558-3564.

[Marshall 1992b] Marshall I. *Acoustic reflectometry for airway measurement.* PhD thesis(1992), University of Edinburgh.

[McIntyre et al 1983] McIntyre ME, Schumacher RT and Woodhouse J. *On the oscillations*

of musical instruments. J.Acoust.Soc.Am.(1983), 74(5), 1325-1344.

[Merhaut 1989] Merhaut J. *Impulse measurement of acoustic impedance.* J.Audio.Eng.Soc.(1989), 37(5), 343-352.

[Mermelstein 1967] Mermelstein P. *Determination of the vocal-tract shape from measured formant frequencies.* J.Acoust.Soc.Am.(1967), 41(5), 1283-1294.

[Morgan and Crosse 1978] Morgan ES and Crosse PAE. *The acoustic ranger, a new instrument for tube and pipe inspection.* NDT International(1978), August issue, 179-183.

[Morgan 1981] Morgan ES. *Experience with the acoustic ranger - a sound method for tube inspection.* Materials Evaluation(1981), 39, 926-930.

[Nederveen 1969] Nederveen CJ. *Acoustical aspects of woodwind instruments.* 1st edition(1969), Frits Knuf, Amsterdam.

[Nelson and Elliott 1993] Nelson PA and Elliott SJ. *Active control of sound.* Paperback edition(1993), Academic Press, London.

[Papoulis 1984] Papoulis A. *Signal Analysis.* International student edition(1984), McGraw Hill Book Company.

[Polack et al 1987] Polack JD, Meynial X, Kergomard J, Cosnard C and Bruneau M. *Reflection function of a plane sound wave in a cylindrical tube.* Revue.Phys.Appl.(1987), 22, 331-337.

[Pratt et al 1977] Pratt RL, Elliott SJ and Bowsher JM. *The measurement of the acoustic impedance of brass instruments.* Acustica(1977), 38, 236-246.

[Press et al 1988] Press WH, Teukolsky SA, Vetterling WT and Flannery BP. *Numerical recipes in C.* First edition(1988), Cambridge University Press.

[Rubinstein et al 1987] Rubinstein I, McClean PA, Boucher R, Zamel N, Fredberg JJ and Hoffstein V. *Effect of mouthpiece, noseclips, and head position on airway area measured by acoustic reflections.* J.Appl.Physiol.(1987), 63(4), 1469-1474.

[Schroeder 1967] Schroeder MR. *Determination of the geometry of the human vocal tract by acoustic measurements.* J.Acoust.Soc.Am.(1967), 41(4), 1002-1010.

[Smith 1988] Smith RA. *It's all in the bore!* International Trumpet Guild Journal 12, 4.

[Sondhi and Gopinath 1971] Sondhi MM and Gopinath B. *Determination of vocal-tract shape from impulse response at the lips.* J.Acoust.Soc.Am.(1971), 49(6), 1867-1873.

[Sondhi and Resnick 1983] Sondhi MM and Resnick JR. *The inverse problem for the vocal tract: numerical methods, acoustical experiments, and speech synthesis.* J.Acoust.Soc.Am.(1983), 73(3), 985-1002.

[Valimaki et al 1995] Valimaki V, Karjalainen M and Huopaniemi J. *Measurement, estimation and modelling of wind instruments*

using DSP techniques. Proceedings of the 15th Int. Congress on Acoustics(1995), 481-484.

[Ware and Aki 1969] Ware JA and Aki K. *Continuous and discrete inverse scattering problems in a stratified elastic medium. I: Planes at normal incidence.* J.Acoust.Soc.Am.(1969), 45(4), 911-921.

[Watkinson et al 1982] Watkinson PS, Shepherd R and Bowsher JM. *Acoustic energy losses in brass instruments.* Acustica(1982), 51, 213-221.

[Watson and Bowsher 1987] Watson AP and Bowsher JM. *Recent progress in time domain work on brass instruments.* Proceedings of the IOA(1987), 9, 103-109.

[Watson and Bowsher 1988] Watson AP and Bowsher JM. *Impulse measurements on brass musical instruments.* Acustica(1988), 66(3), 170-174.

[Watson 1989] Watson AP. *Impulse measurements on tubular acoustic systems.* PhD thesis(1989), University of Surrey.

[Weston 1953] Weston DE. *The theory of the propagation of plane sound waves in tubes.* Proc.Phys.Soc.(1953), B66, 695-709.

Publications

Conference papers

Campbell DM, Parks R and Sharp DB. *Acoustic pulse reflectometry in musical wind instrument research.* Proceedings of the 3rd French Conference on Acoustics, Toulouse, France, 2-6 May 1994; Journal de Physique IV (C5), 657-660.

Sharp DB, Myers A, Parks R and Campbell DM. *Bore reconstruction by pulse reflectometry and its potential for the taxonomy of brass instruments.* Proceedings of the 15th International Congress on Acoustics, Trondheim, Norway, 26-30 June 1995; 481-484.

Sharp DB, Campbell DM and Myers A. *Input impedance measurements on wind instruments using pulse reflectometry.* Proceedings of the International Symposium on Musical Acoustics, Le Normont, Dourdan, France, 2-6 July 1995; 74-78.

Cullen J, Batty D, Campbell DM and Sharp DB. *Acoustical investigation of the cornetto.* Proceedings of Forum Acusticum - the 1st Convention of the European Acoustics Association, Antwerp, Belgium, 1-4 April 1996; Acustica - Acta Acustica, 82(1), supplement 1, 183.

Sharp D, Campbell DM and Myers A. *Pulse reflectometry as a method of leak detection in historical brass instruments.* Proceedings of Forum Acusticum - the 1st Convention of the European Acoustics Association, Antwerp, Belgium, 1-4 April 1996; Acustica - Acta Acustica, 82(1), supplement 1, 188.

Sharp DB, Myers AM and Campbell DM. *Improvements in the resolution of bore*

reconstruction of brass musical instruments by pulse reflectometry. Proceedings of the Institute of Physics Annual Congress, Telford, UK, 23-25 April 1996; 31.

Journal papers

Sharp DB and Campbell DM. *Leak detection in pipes using acoustic pulse reflectometry.* Acustica - Acta Acustica (in press).