Computational Impedance Generation and Bore Optimisation for Matlab

Craig Meek

MSc Acoustics & Music Technology
University of Edinburgh
s0791739
Summer 2012

Abstract

This project focuses on the combination of input impedance generation and optimisation techniques to construct code for Matlab that will produce an impedance curve for a pre-specified bore profile, or suggest a bore profile to fit a pre-specified impedance curve. These two separate but related problems will be referred to as 'impedance generation' and 'bore optimisation' respectively.

Impedance generation is achieved by modelling wave propagation using finite different schemes and devising a method for reconstructing brass instrument shapes. Bore optimisation is performed using the Rosenbrock method along with a suitably defined objective function to rate how closely designs match the target.

The code for both problems is subject to a series of tests to determine accuracy and performance. Following this, improvements are made and suggestions put forward for future work. Finally, applications of the system are outlined in each case.

Acknowledgements

This project would not have been possible without the assistance and willing input of many current and previous members of the University of Edinburgh's Acoustics department. Gratitude is therefore due to a number of people.

First and foremost, thanks go to Mike Newton for supervising this project, arranging regular meetings and generally helping me mull over the multitude of ideas and offshoots that came out of the basic idea of this project. Alistair Braden, the author of a comprehensive PhD project on a similar topic, willingly provided guidance and expertise on several occasions. Stefan Bilbao regularly provided valuable information on finite difference schemes, optimisation theory and lots of other general bits and pieces. Thanks also go to Murray Campbell for introducing me to the topic of impedance analysis and providing his Serpent for research, Arnold Myers for providing detailed bore measurements, and Jacek Gondzio for his excellent and engaging lectures on optimisation theory.

Finally, I thank my classmates for helping create an engaging and enjoyable atmosphere in which to study this masters course.

Contents

Introduction

Motivation

The study and computational modelling of the acoustical behaviour in an instrument bore has become a well-established undertaking in the previous two decades, with a notable motivating factor being to provide assistance in the construction of quality musical instruments. Studying the impedance from a pre-specified bore gives useful indications of the pitch, stability and timbre of sounded notes. The reverse problem - finding a bore profile to fit a pre-specified set of impedance values - allows the user to set a desired output and request a suitable bore reconstruction.

Combining these two methods can provide a useful tool for brass instrument design and construction. While crafting a top class instrument is regarded as an art in itself learnt over many years of practice, instrument modelling and bore reconstruction can still be a valuable supplement in the design process. Many different models can be tested and optimised computationally, saving the need for construction of time and money consuming prototypes.

Many authors have produced work at least partly driven by this motivation, including Braden ([1, 2]), Norland ([3, 4]),Kausel ([5]) and Amir ([6]). Consequently, the source codes are not always widely available, are written in different coding languages, or are focused on running for longer periods of time in order to achieve pinpoint accuracy.

One of the main motivations of this project is therefore not necessarily to create a meticulously accurate tool for manufacturers, but instead to build a relatively more accessible and user-friendly impedance analysis and optimisation package. It is intended to be geared towards students, in the mould of the 'Fourier' and 'Levels' Matlab programmes ([31]) developed at the University of Edinburgh to provide an easy system to analyse data and enhance learning. Further motivation comes from the author's study of both finite difference schemes and optimisation techniques throughout the year, accompanied by the desire to

apply the acquired knowledge on a larger and practical scale.

Project Aims

Following the previous discussion, it is necessary to provide a number of aims of this project:

- To develop Matlab code to find the impedance curve for a specified bore profile, and to optimise a bore profile for a specified impedance curve.
- To rigourously test each case and review accuracy and efficiency.
- To suggest improvements, further work and applications for each case.
- To have the final code accessible so that it may be built on and improved by anyone at a later date.

Summary

This project is split into four distinct sections, with the first half covering the forward problem and the second half covering the reverse problem. Chapter 1 will cover the theory of wave propagation in tubes, leading the way to the introduction of Webster's equation, finite difference schemes and the construction of an '*impedance generator*'. In Chapter 2, the computational model is put through a series of rigorous accuracy tests. The results are then analysed and suggestions are made for improvements. Following this, some practical uses of the system are outlined. Chapter 3 introduces optimisation techniques to deal with the 'reverse problem', and covers the theory of the specific algorithm used with the '*bore optimiser*'. In Chapter 4, the optimiser is put through a series of tests and the results discussed. Challenges in implementation are highlighted, improvements are suggested and any possible future work is detailed. The full code is included digitally, and a hard copy of the main scripts is presented in the appendix.

Chapter 1

Impedance Generation

As already mentioned, studying the input impedance enables us to go a long way in predicting the particular sound of an instrument in terms of intonation, stability and to a lesser extent, timbre. Constructing efficient and accurate code that returns the impedance from a pre-specified bore will also be key to later implementing the reverse problem in Chapter 3, where a constant stream of bore designs need to be tested and their impedance values rated against a target.

1.1 Initial Discussion and Theory

Before going any further, it is worth briefly refreshing the theory of acoustical behaviour in tubes. More specifically, the theory in this project is based on that of lip-driven brass instruments.

One main characteristic of brass instruments is their 'cup-shaped' mouthpieces, necessitating the use of a lip-reed mechanism. In a simplified sense, notes are sounded by interaction between lip vibrations and the resonances of the air column in the main instrument bore. The column may resonate at several different frequencies for any one fingering or valve configuration, and it is the spacing, width and magnitude that govern the overall properties of the emitted sound. These resonances correspond to the natural modes of vibration of the air column contained in the tube.[7, 9]

To produce a note, the player sets their lips to vibrate at a particular frequency and injects an oscillatory flow into the main bore, which induces vibrations into the air column. These vibrations build up, causing the lips to react to the increasing pressure fluctuations and 'lock' into a stable vibration with the air column, producing standing waves. It is the resonances feeding off the energy of the air flow and powering other modes that most influences what note will be

sounded.[1, 7] Given the correct frequency and phase relationships, several modes can feed off the components of the injected air flow to produce a strong resultant note.[8]

Essentially, resonances in the air column that form an approximate harmonic series can be used to set up a stable, cooperative regime of oscillation. The more harmonically related peaks that support a fundamental, the more stable and easy to play the sounded note will be. In reality, the resonant body of brass instruments causes these ideal modes of vibration to deviate, producing overtones and a degree of inharmonicity. This is counteracted by 'mode locking', which causes the overtones to lock precisely onto integer multiples of the fundamental pitch, and hence have the desired phase relationship despite being slightly different to the natural resonances points of the instrument.[9] Taking this all into account, it follows that in searching for patterns of harmonically related peaks, one can predict what notes a brass instrument may play.

1.2 Input Impedance

To build an impedance generator, it is of course necessary to first define input impedance. As the chapter title suggests, it is a key component in this part of the project. Input impedance, denoted as Z, is a frequency dependent value defined as follows:[7, 9]

$$Z = \frac{p}{vS} \tag{1.1}$$

where

p = Acoustic Pressure, Pa

v = Velocity, m/s

S = Surface Area, m^2

The quantity vS is commonly known as *Volume Velocity* or *Acoustic Volume Flow*. The units of Z are hence $\frac{Pa}{m^3/s} = \frac{Pa.s}{m^3}$, known as an *Acoustic Ohm*. Readings are taken at a point close the mouthpiece.[9]

Acoustic impedance measures the pressure level generated by some air vibration at a particular frequency.[10] Informally, it can be described as an 'acoustical fingerprint', as using the theory from section 1.1 allows much information about the behaviour of an instrument to be extracted from its impedance plot. Peaks

on the plots correspond to resonances, the larger the magnitude naturally implying a stronger response. As already highlighted, many large harmonically related peaks will sustain a cooperative regime of oscillation and result in a stable tone being sounded. On a side note, the excitation frequency of the lips need not correspond with a large impedance peak; it is the extent to which a particular frequency can combine with other strong resonances that determines the overall quality and ease of sounding a particular note. This is a common feature of brass instruments.[11]

1.3 Webster's Equation

As the goal of this section is to create a computational impedance generator, it will be essential to model wave propagation in a tube, from which the relevant quantities may be used to calculate impedance. In tubes, the length scale in one coordinate is significantly larger than that in the other coordinates, allowing the dynamics of the model to be reduced to one dimension. In other words, we are assuming that no transverse modes exist and hence only working along the x-axis.[12] The equation to be used is called Webster's equation. The derivation will not be covered in this project, but a thorough description can be found in [1], or from the original paper [14]. Given a discretized bore profile, Webster's equation is defined as follows:

$$S\Psi_{tt} = \gamma^2 (S\Psi_x)_x \tag{1.2}$$

where

Ψ = Velocity Potential

$S(x)$ = Cross sectional area of tube at x

γ = c/L (speed of sound in air/length of tube)

This is a scaled form of Webster's equation, where the x coordinates have been normalised to unit length and the cross sectional area scaled by a reference surface area, usually taken to be the left extremity of the tube. The pressure and volume velocity can be derived from Ψ as follows:

$$p = \Psi_t, u = -S\Psi_x \tag{1.3}$$

The equation as stated above assumes that there are no viscothermal losses. This will be addressed later on, but for the time being we will work with the

lossless version to maintain clarity in how it is modelled computationally. Other assumptions include that the instrument bore has no curvature, the cross section is exactly circular and the wave propagation is strictly planar.

1.3.1 Boundary Conditions

To ensure the computational model works effectively, well-defined boundary conditions are required. The left end of the tube is effectively treated as being closed and coupled to an excitation mechanism to be added later. A typical zero velocity Neumann condition is sufficient - it can be expressed as $\Psi_x(0,t) = 0$. Since the right end of the tube will be radiating sound, it is possible to account for these energy losses by using a Dirichlet type condition with loss terms. The form we will use is $\Psi_x(1,t) = -\alpha_1\Psi_t - \alpha_2\Psi(1,t)$.[12] The two α terms are dependent on the tube parameters. Suitable values are outlined in [12] and [15].

1.4 Modelling Impedance

Instead of modelling using the more common transmission line approach, this project uses finite difference methods. This is in contrast to much of the other literature that deals with computational impedance generation, possibly with the exception of [4] where a mix of the two methods is used. Finite difference methods offer potential advantages over digital waveguides in various areas. The main advantage is that they provide a clearer insight into the physical workings of a system. Waveguides can be more successful in producing synthesised sound, but since the overriding motivation is to study acoustical behaviour, this declines in importance. Historically, use of numerical methods has come at a computational cost, but the speed and quality of modern computers means that this is becoming much less of a concern.

Another advantage (or at least, difference) that has emerged during application of finite difference methods is that they negate the use of sectional analysis - that is, while the specified bore shape may be defined in terms of many sections, the final impedance calculations treat it as a whole. This is in contrast to waveguide methods, that calculate impedances for a series of transmission matricies corresponding to each individual section of the horn. It could be argued that once the fundamentals of finite differences are understood, it can become a more straightforward method of deriving impedance. If nothing else, taking this approach is simply a fresh way of doing things, and could throw up interesting new opportunities.

1.4.1 Finite Difference Schemes

A finite difference scheme is a numerical method that attempts to approximate the overall solution of partial differential equations by approximating the derivatives using finite differences. To elaborate, a solution space is discretized into a finite grid of spacial parts, with time discretized in a similar manner. Derivatives are approximated by taking differences between nearby grid points, and the overall numerical solution advanced by the use of recursive methods. The finite difference methods used in this project are based in the time domain.

The introduction of some basic notation will be helpful in keeping track of the workings of the scheme when it is presented. A more comprehensive glossary of terms is contained in Appendix II, to which any readers less familiar with these techniques are directed. Remembering that the solution space is now a discrete grid, we represent a solution to 1.2 at spacial point l and temporal point n as Ψ_l^n. Derivatives are calculated using a selection of templates, with δ representing a spacial or temporal operator. Most commonly used are forward, backward and central templates. Equation 1.4 below demonstrates the forward case, where h is a spacial step.[1]

$$\frac{\partial}{\partial x}\Psi_l^n = \delta x_+ \Psi_l^n = \frac{1}{h}(e_{x+} - 1)\Psi_l^n = \frac{\Psi_{l+1} - \Psi_l}{h} \tag{1.4}$$

The backward and central cases are derived similarly using $\delta x_- \Psi_l^n$ and $(\delta x_+ - \delta x_-)\Psi_l^n$ respectively. Double derivatives are approximated by combining the forward and backward templates. Temporal derivatives are dealt with in a similar fashion, using a time step k.[12]

1.4.2 Scheme for Webster's Equation

Recall equation 1.2:

$$S\Psi_{tt} = \gamma^2 (S\Psi_x)_x \tag{1.5}$$

A suggested finite difference scheme, set out in [12], is:

$$[S]\delta_{tt}\Psi = \gamma^2 \delta_{x+}((\mu_{x-}S)(\delta_{x-}\Psi)) \tag{1.6}$$

where μ_{x-} is an averaging operator and $[S] = \mu_{xx}S$ is a second order approximation to the continuous form of S. After operator expansion, some rearrange-

[1] To keep the notation simple, the part that is not being operated on will often be omitted. In 1.4, the temporal part of Ψ is omitted since there are only spacial operators active.

ment is required to enable the recursive relation. This involves expressing the temporal update point in terms of everything else. After some algebraic work, 1.6 becomes:

$$\Psi_l^{n+1} = \lambda^2 \frac{S_{l+1} + S_l}{2[S]_l} \Psi_{l+1}^n + \lambda^2 \frac{S_l + S_{l-1}}{2[S]_l} \Psi_{l-1}^n + 2\left(1 - \lambda^2 \frac{S_{l+1} + 2S_l + S_{l-1}}{2[S]_l}\right) \Psi_l^n - \Psi_l^{n-1} \tag{1.7}$$

where

$\lambda = \gamma k/h$, known as the *Courant* number.

Once the scheme has been run, the fourier transform of Ψ is taken in order to observe the results in the frequency domain.

Boundary Conditions

Clearly, the scheme will be compatible for interior points of the grid domain. However, when we try to use the boundary points $l = 0$ and $l = N$, problems emerge as 1.7 attempts to access grid points Ψ_{-1} and Ψ_{N+1} which of course do not exist. This problem was accounted for in the previous section, where boundary conditions were imposed for each end of the instrument. Assimilating these into the scheme is fairly straightforward, with the Neumann case providing a simple example:

$$\begin{aligned} \Psi_x(0,t) &= \delta_x \Psi_0 \\ &= \frac{\Psi_1 - \Psi_{-1}}{2h} \\ &= 0 \\ \therefore \Psi_{-1} &= \Psi_1 \end{aligned}$$

Similarly, S_{-1} is set to S_1 and due to various cancellations there is much simplification of the surface area quotients in 1.7. The grid point $l = 0$ then becomes:

$$\begin{aligned} \Psi_0^{n+1} &= \lambda^2 \Psi_1^n + \lambda^2 \Psi_{-1}^n + 2(1 - \lambda^2)\Psi_0^n - \Psi_0^{n-1} \\ &= 2\lambda^2 \Psi_1^n + 2(1 - \lambda^2)\Psi_0^n - \Psi_0^{n-1} \end{aligned}$$

which is an explicit result for behaviour at the left boundary. With a bit more effort, the Dirichlet boundary condition with radiation loss can be added to the scheme in a similar fashion. Note that in the accompanying source code the boundary condition expressions are broken into parts to maintain clarity.

Adding Viscothermal Losses

While radiation loss has already been taken care of, there is still scope to further improve the model by accommodating viscothermal losses. This incorporates loss terms accredited to a viscous boundary layer at the walls of a tube, and the effect of vibration and damping in the tube walls themselves.[12] Modelling viscothermal losses is relatively straightforward in the frequency domain, but can be more consuming in the time domain.[13] Nevertheless, a fairly simple model presented in chapter 9 of [12] takes care of many of these considerations. It extends Webster's equation and couples it to another functions w which incorporates damping parameters. A full statement of the extended system with associated parameter descriptions is presented in Appendix II.

Further Comments

Some more minor but still important points need to be mentioned. The excitation mechanism to be bolted on to the Neumann boundary condition will simply be a unit impulse. Another essential task is to ensure scheme stability. Through energy analysis[2] of 1.7, it is determined that the *stability condition* is $\lambda \leq 1$. Having λ as close to 1 as possible is advisable for these schemes. Of particular note is the special case where $\lambda = 1$, which in fact simplifies 1.7 to a digital waveguide.[12] Finally, since finite difference methods only approximate an overall solution, there will of course be some (small) error. Forward and backward schemes have error proportional to h (or k, if temporal operators are being used), while central schemes have error proportional to h^2 (or k^2).[16]

1.4.3 Defining the Bore Profile

Now that the theory of impedance computation has been covered, we move on to discussion about how to construct bore profiles to be used as the input. As the resulting programme is intended to be user-friendly, it is essential that the construction be relatively straightforward and easy to use while at the same time

[2]Energy analysis is a commonly used tool to determine stability of finite difference schemes. Although it is beyond the scope of this project, much theory and application of the methods is contained in various chapters of [12].

being flexible enough to define many instrument shapes and features sufficiently. The approach adopted here is to use a concatenation of cylinders, truncated cones and bessel horns. This allows effective and reasonably accurate construction of instrument shapes, but without over complicating things. It also enables the user to easily specify 'jump sections', which may occur if a particularly detailed design is being constructed.[3]

When working with multiple bore sections, the equivalent sample length must be calculated accurately for each section. Recall that a bore of length L will be discretized into a finite grid of spacial parts. More specifically, the instrument radius will be 'sampled' at N equally spaced points. If there are NS sections each with length $d_1, d2...d_{NS}$, then the number of points allocated to section i will be $N\frac{d_i}{L}$, rounded to the nearest integer. This ensures that the ratio of section length to overall length will always be maintained, even if section lengths are repeatedly changed. Each sample point requires calculation of a radius, which in turn is used to calculate the surface area values S for the finite difference scheme.

Cylindrical and conical sections are easy to define. The cylindrical case simply takes two variables - length and radius, with the latter remaining constant all the way along the section. Conical cases requires three variables - length, input radius and output radius. The cone profile then consists of N_i linearly spaced points between the two specified radii, where N_i is the number of sample points assigned to the section, calculated in the same way as above.

Constructing bessel horns is a slightly more complicated but still worthwhile undertaking as they offer a reasonable approximation of instrument bells.[2] A bessel horn can be defined as follows:

$$r(x) = b(x + x_0)^{-\gamma} \tag{1.8}$$

where r is the radius at point x along the instrument and γ is the flare coefficient. Furthermore;

$$b = \left(\frac{d}{r_0^{-\gamma} - r_1^{-\gamma}}\right)^{-\gamma} \tag{1.9}$$

$$x_0 = -\left(\frac{r_0}{b}\right)^{-1/\gamma} \tag{1.10}$$

with r_0 and r_1 being input/output radii respectively and d being the length of the section. Therefore, constructing the bessel case requires four variables -

[3] 'Jump section' refers to when the end radius and starting radius differ in two consecutive sections.

length, two radius values and a flare constant, which is generally between 0.5 and 0.8. Lower values of γ produce a more rapid flare.[9]

1.4.4 Implementation of Impedance Generator

The code included along with the project is written entirely in Matlab and is used for all the later experiments. Given a pre-specified instrument shape, it models the acoustical behaviour in the bore and returns four plots - two impedance plots (one linear, one logarithmic); a plot of the 2D cross-sectional area; and a 3D rendering of the instrument.

User input includes the total instrument length and the number of individual sections along with all the quantities to describe each. These quantities are presented in vector form. When entering the information for a particular section, the first entry in each 'section vector' is required to be an identification number indicating the type of section to be created. 1 represents a cylinder, 2 a cone and 3 a bessel horn. For example, a conical section vector is entered as [2, d, r_0, r_1]. This identification method allows the programme to select the relevant script to construct each bore section.

Chapter 2

Experiment and Analysis of Impedance Generator

With the conclusion of the previous chapter we now may explore the performance and applications of the impedance generation software. The main ethos of this chapter is to investigate the robustness of the program and give an insight into how it can be used. Preliminarily, experimental tools and analysis methods are summarised, and this is followed by accuracy tests and demonstration of practical use. Comments on strengths, weaknesses, improvements and miscellaneous intricacies are noted throughout.

2.1 Experimental and Analytical Tools

2.1.1 BIAS

Testing the performance of the impedance generator necessitates the need for accurate impedance curves of real instruments so comparisons can be made. BIAS, or Brass Instrument Analysis System, is now a widely used system renowned for producing reliable and accurate impedance data.[30] All physical impedance measurements in this project were carried out over a frequency range of 0-4096 Hz.

2.1.2 Equivalent Fundamental Pitch

A common and useful tool for analysis of impedance comes in the form of the EFP.[1, 11, 8] This takes an arbitrary reference frequency f_0, and compares the measured resonance frequencies to the harmonic resonances of f_0. The units of measurement are cents. The shape of the plot is independent of the value of f_0;

nevertheless, common values include setting f_0 equal to the fundamental or equal to $f_4/4$.[1]

While the EFP can assist to an extent in judging the strength of cooperative regimes, it is also beneficial in comparing different sets of readings, such as BIAS data against our model data.

2.1.3 Sum Function

Judgement of cooperative regime strength and prominently sounded pitches can be represented graphically and numerically by the sum function. It helps give a rough idea of how well aligned the impedance peaks are for a given fundamental frequency, and hence give an indication of whether this particular frequency will produce a strong, easily playable regime.

The sum function was first suggested by Wogram in 1972 - for in-depth information see [17]. The theory is straight forward: for each frequency value in the experimental range, the harmonic partials are calculated and their corresponding impedance values summed. This is summarised by the following formula:

$$S(f_0) = \sum_{f_p=f_0}^{f_p=f_h} Z(f_p) \tag{2.1}$$

where

f_0 = Fundamental frequency

f_p = Harmonic partial of the fundamental

f_h = Highest partial

$Z(f_p)$ = Input impedance for partial f_p

Plotting the numerical values helps indicate the most easily attainable frequency regimes for the instrument under investigation. However, care is needed when drawing conclusions from the data as it can over-represent low frequencies or frequencies whose integer multiples lie on the sides of some particularly tall peaks, giving a distorted representation. This issue can be lessened by using a scaled version of the sum function where the total sum is divided by the number of components it was comprised of. Another method of improving the resolution is to introduce different weighting on successive harmonics.

[1] f_4 is the 4th resonance peak. One reason for its use is that the fourth resonance is commonly used by musicians for tuning.

2.2 Accuracy Tests

On first impressions, the impedance plots look reasonably good. They have prominent, well defined peaks and obvious decay due to the modelling of combined energy losses. The peak width on the linear plots is one area immediately identified for improvement. Nevertheless, the general shape is largely what we would expect. Of course, the real importance lies in whether the model can reproduce plots accurately when compared against measured data, hence this will be the main focus of this section.

Accuracy of the impedance generator is explored by comparing measured and computed impedances of bore profiles through use of some of the methods described above. Two selected tests are presented below, with all physical readings obtained using BIAS. The first is rather simplistic, intended to demonstrate that the model is satisfactory at a basic level. The second is more detailed and complicated, with the intention of testing the model to its limits. Comments will be made on accuracy throughout, with a view to improving the model as much as possible. All computational measurements were carried out at a sample rate of 44100Hz.

2.2.1 Test 1: Simple Cylinder

The profile of a simple cylinder consists of a single section that can be recreated perfectly by the generator. Hence it is a plausible choice when there is a desire to keep things as simple as possible. The cylinder in question is a brass tube of length 1006mm and inner radius 12.5mm, open at one end and closed at the other, with no other features of note besides its dimensions. Fig 2.1 displays the resulting impedance curves in linear and logarithmic plots.

The graphs display a set of odd harmonics gradually decreasing in magnitude, which is what we would expect from a tube open at one end and closed at the other.[10] Initial inspection reveals that the peak locations seem to agree to a reasonable extent. The frequencies of the lower resonance peaks are closely matched between the calculated and measured cases, but as the frequency increases the calculated peaks gradually become sharper in intonation.

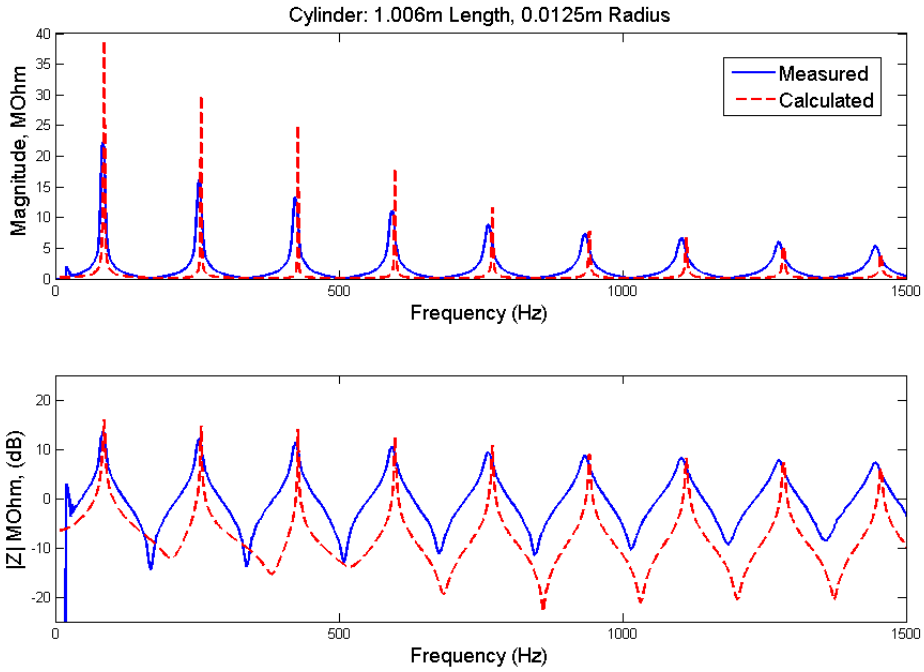

Figure 2.1: Linear and Logarithmic Impedance Plots for a Simple Cylinder

To give a clearer picture of how accurate this case is, we can quantify these differences and observe them in an equivalent fundamental pitch plot. The full results for the first 10 peaks of 2.1 are presented below and compared to the measured impedance values. Note that since the even harmonics are missing, figure 2.2 must label the peaks relative to their place in the theoretical complete harmonic series for the cylinder.

Table 2.1: Impedance values for the first 10 resonance peaks

Peak	Measured Freq.	Model Freq.	Absolute Difference (cents)
1	84	85	20
2	254	257	20
3	423.5	428	18
4	593.5	599	16
5	763.5	770	15
6	933.5	942	15
7	1104	1113	14
8	1276	1284	11
9	1445	1455	12
10	1616	1626	11

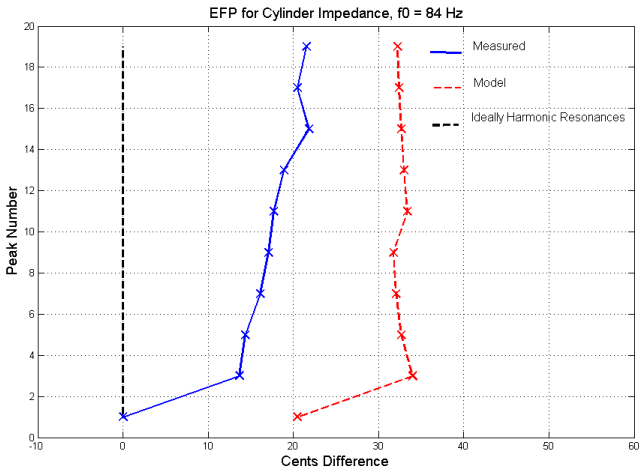

Figure 2.2: EFP for Simple Cylinder. $f_0 = 84$Hz

Both table 2.1 and figure 2.2 show that despite the initial observed discrepancies, the model is actually very accurate. The maximum peak difference is 20 cents, which is acceptably low - to add a bit of perspective, 100 cents corresponds to one semitone in an equally tempered scale. Although the frequency gap slowly increases with peak number, the absolute difference actually falls due to the nature of intervals being calculated by frequency ratio. Note also that all the peaks lie close to the ideal resonance line, supporting the well known theory that cylinder resonances form good harmonic series.

These results confirm that the model is at least accurate enough to try on more complicated designs. One other area that has not yet been mentioned is peak magnitude; it is noticeable from the plots that the magnitudes from the model are consistently higher than those from the physical measurements. Further experiments and observations in [1] indicate that this over-estimation is down to the plane wave assumptions made when modelling Webster's equation. Regardless, magnitude accuracy is not overly important, as the impedance magnitudes in instruments depends much on the techniques of the players themselves. The size of magnitude decrease between consecutive peaks is consistent between the modelled and measured values; so the discrepancies should be irrelevant even when attempting to analyse the strength of any cooperative regimes.

2.2.2 Test 2: A More Complicated Bore Profile

As the modelling process has been deemed acceptable, we can try it on a more realistic instrument profile. The acoustics department has records of detailed measurements of several brass instruments, along with their measured impedance.

The geometrical measurements are very precise and meticulous, which will minimise experimental error in this area. The particular design selected was a Hofmaster natural trumpet in E♭ of 2.1m length, with accurate measurements of the lead pipe, jump sections and bell dimensions all included. The resulting reconstruction displayed in 2.3 and 2.4 is an 11 part bore consisting of 7 cones, 3 cylinders and a bessel section - a full description of the parameters for each section is included in Appendix III.

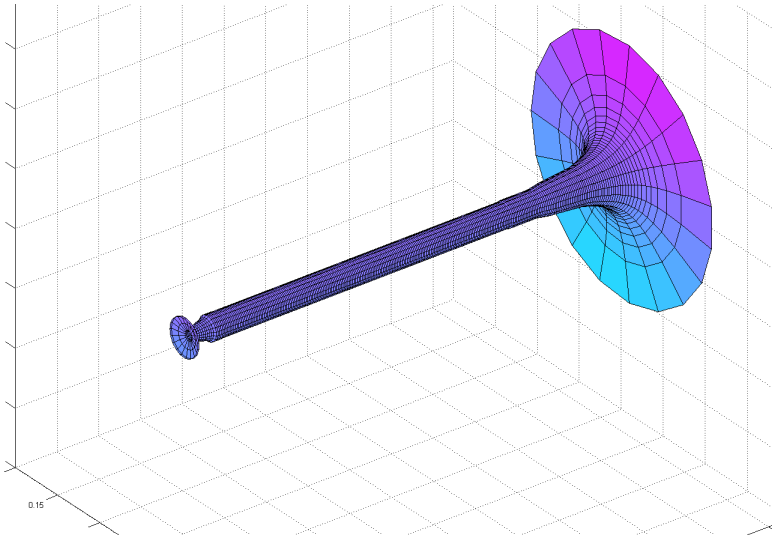

Figure 2.3: 3D plot of detailed bore profile

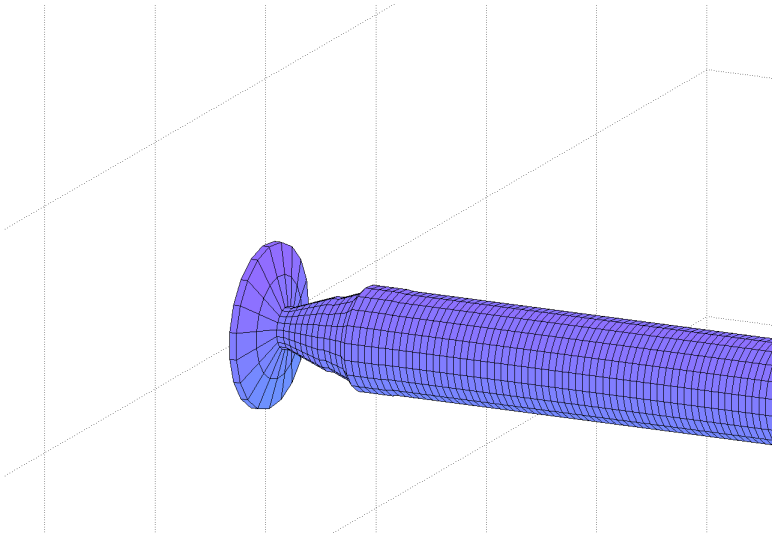

Figure 2.4: Close up of mouthpiece section

Similarly to the previous test, examination of various plots can provide us with plenty of information on accuracy:

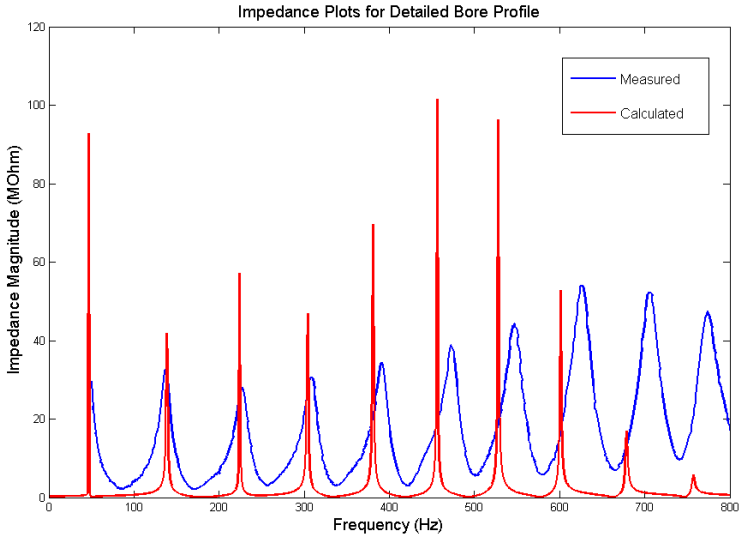

Figure 2.5: Impedance Plot for Detailed Bore Profile

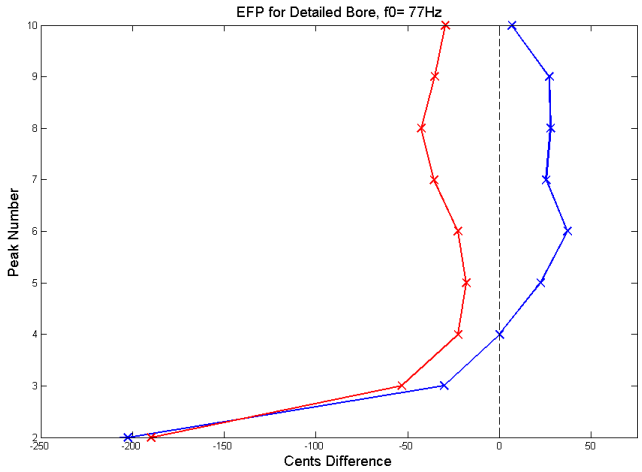

Figure 2.6: EFP for Detailed Bore. $f_0 = 77$Hz

Again, the lower frequency peaks generally seem to line up, but this time the peaks produced by the model become flatter as the frequency increases. It is clear that the discrepancies are much greater than in our previous case; then again, this is to be expected from such a complicated design. Frequency difference between the higher peaks sits at around 16Hz, while the maximum and minimum differences measured in cents are 71 (peak 8) and 13 (peak 2) respectively. Although this is quite a significant rise, it should be remembered that this still isn't even near a semitone in pitch difference, and hence the results could still be considered reasonable. Another positive to note is that the pattern of harmonicity in the peaks is followed closely by the model, as seen clearly in figure 2.6. Also, the

peaks around 400-600Hz have been boosted in height, which is likely due to the influence of the mouthpiece included in the design. One other point of interest is the flatness in pitch of the second peak, which highlights how these graphs can indicate areas of instruments that potentially need improved.

2.2.3 Further Comments

Analysis has showed that with increase in sophistication of bore profile comes a decrease in impedance accuracy. Test 1 had an average error of 15.2 cents and test 2 an average of 44.4 cents. Despite this, even in the most complicated cases the computational results still hold a degree of plausibility. The modelled lower frequencies were generally always in agreement with the measured values. This is a strong result in itself, as these lower peaks are often fundamentals or have a large influence in the pitch of the resulting sound. Performance-wise, the code runs smoothly and can calculate impedances for even the most complicated designs in under 2 seconds.

Sources of Error and Improvements

Even with the model being deemed acceptable, it is essential to discuss possible sources of error along with tabling some potential improvements to minimise it. Experimental error can roughly be attributed to one of two cases: modelling error or measurement error.

Modelling Error: There were many assumptions made when modelling wave dynamics in a tube using Webster's equation. Two of the most error-prone were that no transverse modes existed, and that all wave propagation was planar. Unsurprisingly, real acoustic behaviour in tubes is much more complicated, with mutli-modal oscillations existing and waves with curved wavefronts being produced by flaring horns.[9] Hence it would be advantageous to further adapt the model to include these features. Further experimentation in [1] (Ch.8) demonstrates that using a multi-modal model has a significant effect in shifting the frequency and magnitude of modelled peaks closer to the measured values.

Assumptions made in modelling viscothermal losses could also contribute to overall error. Values for the damping parameter σ_0 were only set experimentally, and the model does not account for the level of energy absorbtion relative to material of the tube. Furthermore, the characteristics of wave propagation have been modelled under the assumption that the viscous boundary layer at the inner tube wall remains isothermal. Keefe ([18]) points out that there is in fact local

temperature change at the tube walls. Incorporating this into our model is beyond the scope of this project, but it is still certainly worth highlighting as a source of error.

Finally, it should also be remembered that finite difference schemes only offer an approximate solution to partial differential equations. Hence, they can additionally be responsible for some error.

Measurement Error: The significance of errors due to measurement lies in the fact that impedance results can be sensitive to even very small discrepancies in the geometry of the bore.[1] Despite successfully being able to construct detailed virtual instruments using only three core sections, there are bound to be small areas where there is slight deviation from the real design. Additionally, the lack of curvature in modelled profiles will contribute to reduced accuracy. One practice known to offer good improvements is to construct a bell with multiple concatenated bessel horns instead of just one.[1]

Inaccuracies from the BIAS equipment could have affected accuracy in the physical measurements, which in turn will have introduced error into the accuracy measurements for the impedance generator. Recently, tests have been performed using a larger frequency sweep in order to improve the robustness of BIAS.

Improvements in all these areas would undoubtedly improve the alignment and accuracy of our impedance curves. The bandwidth, magnitude and shape could also benefit, but it should be remembered that these features can also be heavily influenced by the embouchure of the player.

2.3 Practical Uses

The previous section shows that despite the discrepancies and difficulty in achieving perfection, the impedance generator is still accurate enough to be used as a basic design, analytical and investigative tool. To complete the chapter, this section briefly presents two example applications of the system.

2.3.1 Application 1: Prediction of Notes with the Sum Function

The first application explores how the impedance generator can be used to roughly predict the pitches sounded by any bore shape. This can be done by inspection of the peak positions and/or by use of the sum function. To demonstrate, we

need run the impedance generator on a virtual model of an instrument and then compare our pitch predictions to those measured in playing tests on the real instrument. Conveniently, all the relevant data is available from a previous project by the author ([19]), so it will be suitable to re-use it here.

The instrument to be analysed is a Serpent. Despite being quite complicated acoustically, when all holes are closed it can be estimated as a truncated cone. The Serpent used in the playing tests has a total length of 2.38m, consisting of a mouthpiece of length 0.025m, a crook of 0.41m and a main bore section of 1.94m. The internal radius at the left end is 5.5mm, while at the right end it begins at 48mm and gradually rounds off to 54.5mm. This bore was modelled in the impedance generator using four sections (3 conical and 1 bessel), producing the curve shown in 2.7.

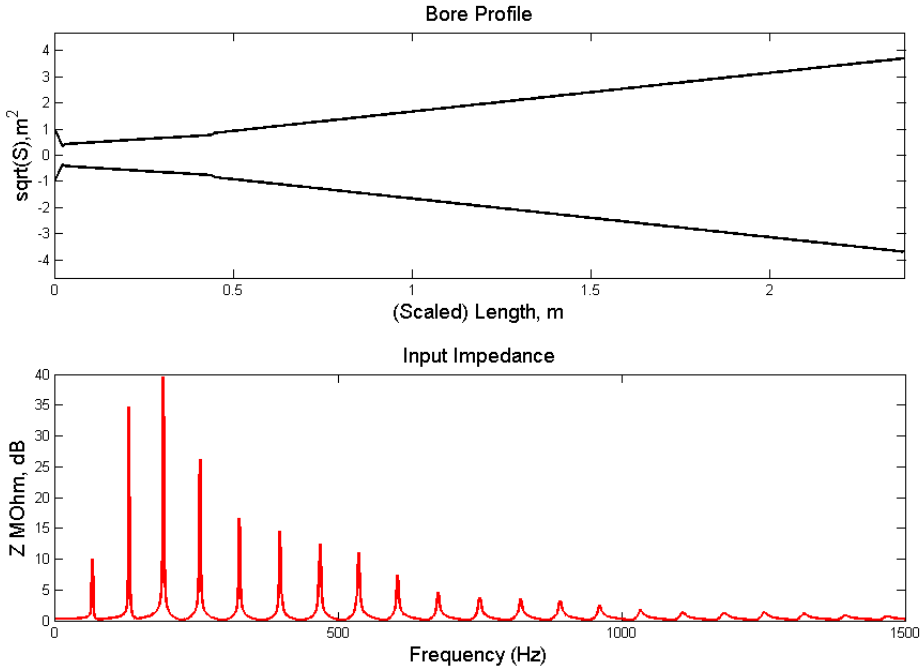

Figure 2.7: Bore Profile and Impedance Curve for Serpent

This curve generally agrees with observed values on the BIAS measurements of the actual instrument. In order to predict what pitches may be easily played, we must look for evidence of strong cooperative regimes - in other words, we look for strong harmonically related peaks supporting some base frequency. This is where the sum function comes in handy, as it produces a graph from which the strongest frequency regimes can be read off. Fig 2.8 is the relative sum function plot for the impedance curve in 2.7. Any partials above 3000Hz are ignored, as they will contribute little, if anything to the overall intonation.

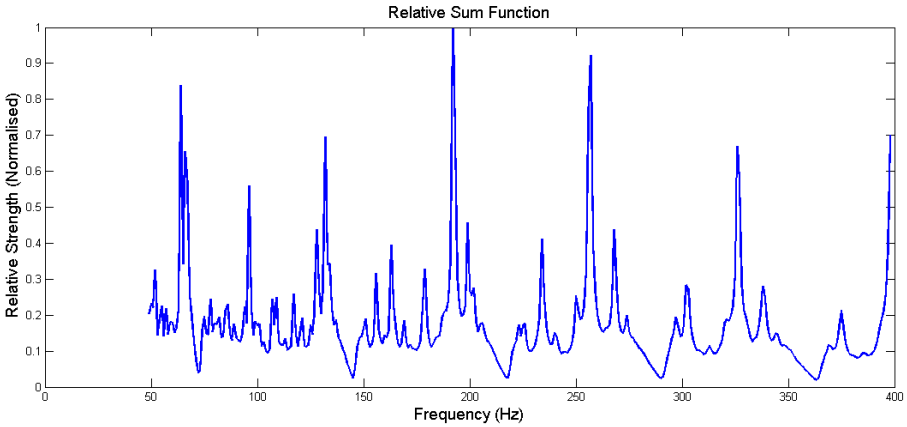

Figure 2.8: Relative Sum Function

The resulting plot has several well defined peaks, representing the frequencies at which cooperative regimes may be most easily formed. The five tallest suggest that with all holes closed the serpent will pitch at 64Hz, 132Hz, 192Hz, 257Hz and 326Hz. Playing tests confirm that pitches of 66Hz, 135Hz, 200Hz, 265Hz and 340Hz are sounded, which are all within the vicinity of our predictions.[19] These correspond to the notes C_2, C_3, G_3, C_4 and E_4 on the Serpent. This demonstrates that the impedance generator can be used adeptly in prediction of how instruments may sound.

2.3.2 Application 2: Bore Alteration

The second application focuses on impedance behaviour when certain features of a typical brass instrument are added or removed. It is intended that this could be used as a tool for basic demonstrations, or to simply entertain the curiosity of the more adventurous user.

It is already common knowledge that items such as a mouthpiece or bell will alter the shape and positioning of impedance peaks.[20] Starting with a simple cylinder, we will use the impedance generator to produce and study two cases where a new part is attached.

Figure 2.9 demonstrates the effect of an attached mouthpiece. The main points to note from the plot is that the peak heights have been boosted and the higher resonance frequencies slightly decreased. The magnitude increase is due to the mouthpiece acting like a Helmholtz resonator while the lowering of frequencies is because it also effectively lengthens the tube, which becomes more profound at higher frequency.[20]

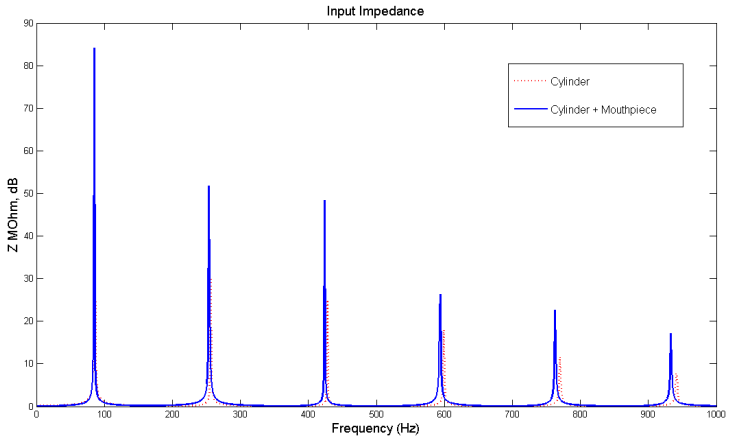

Figure 2.9: Comparison when Mouthpiece is attached

The attachment of a bell is explored in 2.10. The plot indicates that it roughly does the opposite to the mouthpiece; there is a reduction in magnitude at higher frequencies along with an increase in the frequency values of lower pitched resonances. Physical experiment shows that a bell provides more efficient radiation of high frequency waves, giving the characteristic 'brassy' sound. Since more energy is radiated, there will be less remaining inside the bore, as shown on the plot. The rise in frequency of the lower resonances is attributed to the bell effectively shortening the the tube, which will most affect behaviour in the lower frequencies.[20]

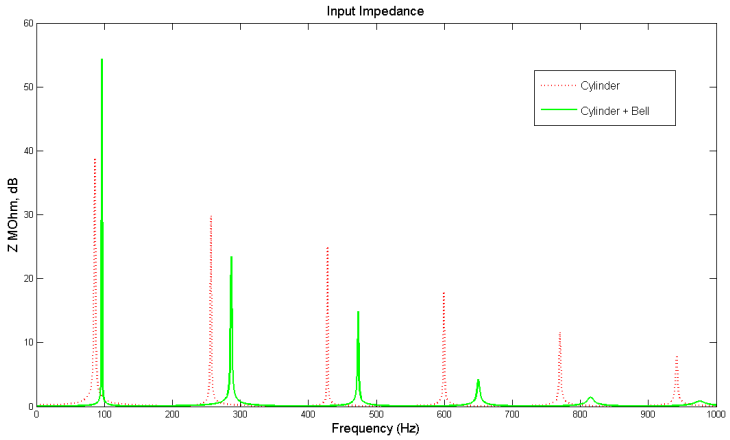

Figure 2.10: Comparison when Bell is attached

This brief discussion has demonstrated that data produced from altering the bore profile is consistent with physical observations, and therefore the system is also acceptable for this type of application.

Chapter 3

Bore Optimisation

3.1 Initial Discussion

We now turn our attention to the reverse problem, where the aim is to generate a bore that matches some pre-specified impedance values as closely as possible. This is arguably a more challenging task than the forward problem; however, once understood, the theory for carrying out such a computation actually seems fairly intuitive and simple. The key to tackling the problem is to run through lots of possible bore designs and quantitatively compare the impedance curve of each one to the target curve by using an objective function to rate the suitability of each design. The aim will be to develop some sort of 'search method' that can find the closest match, which will correspond to the minimisation of the objective function.

Informally, the above essentially describes an optimisation process. The general form of an optimisation problem, adapted for the notation used herein, is as follows:[21]

$$\begin{aligned} \min \quad & O(\boldsymbol{\alpha}) \\ \text{subject to} \quad & g(\boldsymbol{\alpha}) \leq 0 \\ & \boldsymbol{\alpha} \in \mathbb{R}^{N_v} \end{aligned}$$

where $O : \mathbb{R}^{N_v} \to \mathbb{R}$ is the objective function and $g : \mathbb{R}^{N_v} \to \mathbb{R}^{N_v}$ is a system of constraints on the various instrument parameters. Nv represents the total number of instrument parameters, which are arranged into a vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2 ... \alpha_{N_v})$. For example, an instrument made up of a cylinder and a cone is

written $\boldsymbol{\alpha} = (r_0, d_1, r_0, r1, d_2)$.

To take this initial theory further, we need to choose a suitable optimisation algorithm and define an appropriate objective function.

3.2 Rosenbrock Algorithm

There are many different types of optimisation algorithms, including direct search, gradient-based and stochastic. The Rosenbrock method[22, 1, 23] is a 0th order direct search algorithm which is suited to our problem. As hinted by '0th order', it does not require any gradient information in order to work, which is advantageous since gradient analysis requires calculation of many computationally-expensive derivatives if not skilfully formulated. Additionally, the algorithm is fairly easy to understand, which should make it easier to alter and improve the code in any future work.

3.2.1 Outline

As previously stipulated, the aim is to find a vector of parameters $\boldsymbol{\alpha}$ which gives the minimum value of the objective function O, denoted $O(\hat{\boldsymbol{\alpha}})$. The algorithm takes a series of steps in strictly orthogonal directions, rating each new set of parameters and regularly updating the overall direction of movement to point in the direction of best success.[1, 22] This can be roughly summarised in two sections: the **Exploratory Stage** and the **Orthogonalisation Stage**. Much of the theory here is based on the detailed articles of [1], [22] and [24]. Figure 3.1 is a good demonstration of the algorithm in action, showing the navigation towards the minimum of a function of two variables. In most applications, the number of variables will be higher and hence the movements will be difficult to visualise.

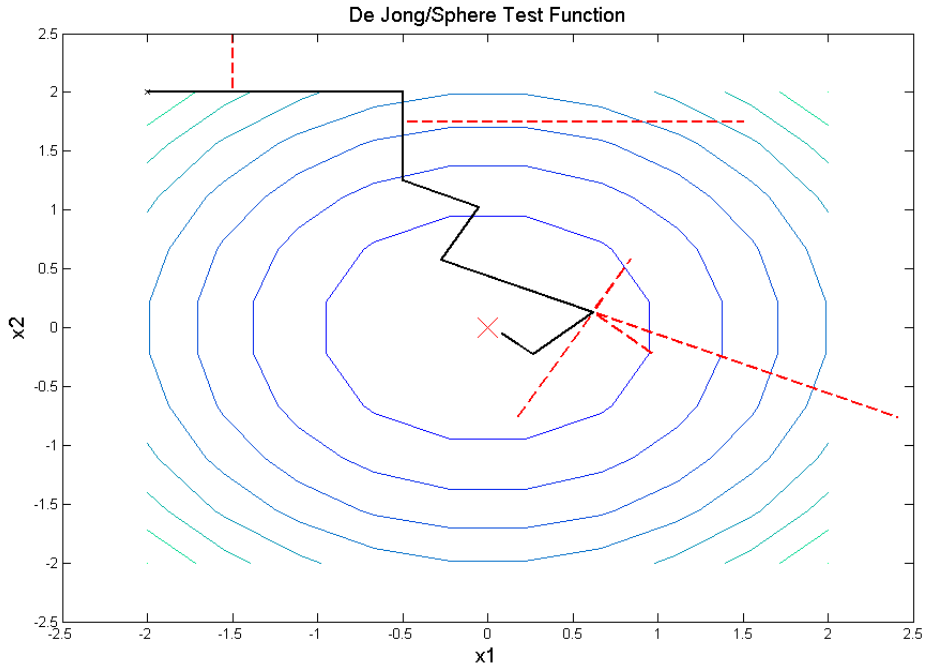

Figure 3.1: Rosenbrock Optimiser working on a contour plot. Black lines denote a successful step; dotted red lines denote a failed step

Initial Settings

Working with an instrument of N_v parameters can be thought of as working in N_v-dimensional space, with each parameter lying on a coordinate axis. Various quantities must be defined to initialise the algorithm. An N_v x N_v direction matrix $\boldsymbol{D}$ with mutually orthogonal columns holds direction vectors $\boldsymbol{d}_i$ for each parameter, with $|\boldsymbol{d}_i| = 1$. In default settings, the starting directions are simply coordinate directions, hence $\boldsymbol{D}$ will be the identity matrix $\mathbb{I}_{\boldsymbol{N_v}}$. The step lengths $\boldsymbol{l} = (l_1, l_2..., l_{N_v})$ for each direction $\boldsymbol{d}_i$ also need to be defined. along with values for two tuning parameters $a > 1$ and $0 < b < 1$. Finally, we must specify an initial value for $\boldsymbol{\alpha}$ in order to calculate a starting value for O.

Exploratory Stage

Using the terminology just defined along with $\boldsymbol{\alpha_c}$ to represent the current set of parameters, the algorithm takes a step of distance l_i in direction $\boldsymbol{d}_i$ from $\boldsymbol{\alpha_c}$. This can be expressed as

$$\boldsymbol{\alpha} = \boldsymbol{\alpha_c} + l_i \boldsymbol{d}_i^T \tag{3.1}$$

This provides us with an updated set of parameters which describe a new bore design. It is then necessary to check if the impedance curve of this new design is any closer to matching the target curve. Therefore, the impedance generator is used to get $Z(\boldsymbol{\alpha})$, and this data is in turn fed into the objective function to return a numerical rating. From this point, the algorithm proceeds in one of two ways:

- If $O(\boldsymbol{\alpha}) \leq O(\boldsymbol{\alpha_c})$, then the step is deemed successful as the objective function indicates that this design is closer to matching the target. $\boldsymbol{\alpha_c}$ is set to $\boldsymbol{\alpha}$ and the step size l_i is multiplied by a, so that it will search further in this direction next time.
- If $O(\boldsymbol{\alpha}) > O(\boldsymbol{\alpha_c})$, then the step is deemed a failure. $\boldsymbol{\alpha}$ is discarded and $\boldsymbol{\alpha_c}$ is retained as the current preferred design. l_i is multiplied by b, which reverses and reduces the direction of travel for the next cycle.

This procedure is repeated for each dimension i until both a success and a failure have been found in each direction. This indicates that the search has been exhausted for the current set of direction vectors contained in $\boldsymbol{D}$, and triggers the beginning of the orthogonalisation stage.

Orthogonalisation Stage

The purpose of this stage is to recalibrate the direction vectors so that they point in the direction of best progress so far, while at the same time maintaining the orthogonal nature between direction vectors. It can also be thought of as rotating the rigid 'coordinate frame' to further guide the optimiser towards the likely optimal point.

Again, it is necessary to define some initial quantities in order to carry out the orthogonalisation. During the exploratory stage, the total distance travelled in each direction $\boldsymbol{d}_i$ is denoted by λ_i. We now define a new matrix $\boldsymbol{\Lambda}$, which logs the λ_i's column-wise as follows:

$$\boldsymbol{\Lambda_k} = \sum_{i=k}^{N_v} \lambda_i \boldsymbol{d}_i^{(0)} \tag{3.2}$$

where $\boldsymbol{d}_i^{(0)}$ indicates the direction vectors from the previous stage. The structure of this matrix is easiest to understand when we consider the initial case where the $\boldsymbol{d}_i$ are simply coordinate vectors. Then 3.2 produces the matrix

$$\mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & & \\ \lambda_2 & \lambda_2 & & \\ \vdots & \vdots & \ddots & \\ \lambda_{N_v} & \lambda_{N_v} & \dots & \lambda_{N_v} \end{pmatrix} \tag{3.3}$$

where the first column contains all successful steps in an exploratory stage, and hence joins the beginning and end points of the stage. The procedure to obtain new direction vectors is based on the Gram-Schmidt technique[25] and uses these new quantities along with the direction vectors from the previous stage, $\boldsymbol{d}_i^{(0)}$.

The explicit derivation will not be covered here, but it is well documented in [1], [22] and [24]. One point of note, however, is that the version provided in [22] is susceptible to numerical instability in cases where the total distance travelled in a certain direction is equal to zero (this is possible when the algorithm 'doubles back' on itself in a particular direction). To avoid this occurrence, the modified version of Palmer ([24]) is used. The new direction vectors are then be expressed as

$$\boldsymbol{d}_i^{(1)} = \frac{\lambda_{i-1}\mathbf{\Lambda_i} - \boldsymbol{d}_{i-1}^{(0)}|\mathbf{\Lambda_i}|^2}{|\mathbf{\Lambda_{i-1}}||\mathbf{\Lambda_i}|} \tag{3.4}$$

for $2 < i < N_v$, with

$$\boldsymbol{d}_1^{(1)} = \frac{\mathbf{\Lambda_1}}{|\mathbf{\Lambda_1}|} \tag{3.5}$$

for $i = 1$. In the special case where $\lambda_{i-1} = 0$, $\boldsymbol{d}_i^{(1)} = -\boldsymbol{d}_{i-1}^{(0)}$.

This new set of direction vectors is then used in another exploratory stage, and the whole process keeps repeating until certain termination criteria have been met.

Constraints and Termination Criteria

It is in our interest to restrict the solution space as much as possible without ruling out sensible instrument designs. The most obvious constraint is to ensure that all parameter measurements α_i are positive, which of course keeps the problem within the bounds of physical reality. It is worth imposing more bounds to further restrict the solution space; additionally, upper and lower bounds are assigned to all lengths, radius and flare parameters. If the algorithm strays outside these boundaries, the objective function is simply set to a maximum, and so the optimiser will take no further interest in that particular set of parameters. Other

constraints on the system include eliminating the physically incompatible case of $r_1 > r_0$ for bessel horns, and restricting the order sections can be arranged in to ensure the aforementioned shape can only be placed at the end of an instrument.

A range of termination criteria can be utilized, but the more practical are time and distance restrictions. Time restrictions are self explanatory; a threshold can be set and when the elapsed time exceeds this, the algorithm terminates. Slightly more practical is the latter case, which measures the maximum distance travelled in coordinate directions by subtracting $\boldsymbol{\alpha}$ from $\boldsymbol{\alpha_c}$ after each iteration. Setting an acceptably small threshold will ensure that the algorithm has enough time to converge on a possible solution before termination.

3.3 Objective Function

To complete the bore optimiser we need to define an objective function that provides a suitable rating of how closely a curve matches the target. Since peak location and magnitude are the two most influential features of an impedance curve, it seems logical to base any objective rating system around these. We also wish the objective function to be as smooth as possible and to have a prominent, obvious global minimum - something similar to De Jong's function in figure 3.1 would be an ideally suitable case in two dimensions.

The strategy we will adopt is to use two separate objective functions combined together with different weights to give the overall rating as a score between 0 and 1, with 0 being a perfect match and 1 being no resemblance whatsoever. This is keeping within the standard practices of optimisation techniques where the primary aim is to minimise the objective function. The two functions rate peak location and height relative to the target curve. They are each based on a normalised inverted Gaussian function[26], which possesses the basic characteristics that we seek along with flexible parameters to alter the general shape. The peak location function is expressed as

$$O_1(\alpha) = \frac{1}{Npk} \sum_{i=1}^{Npk} 1 - \exp\left(\frac{-\mu_p \delta\phi_i^2}{\nu_p^2}\right) \tag{3.6}$$

where $\delta\phi_i = |\phi_i - \bar{\phi}_i|$ is the difference between the test and target locations of peak i, μ_p is a strictness parameter, ν_p is a windowing parameter and Npk is the total number of peaks being tested. μ_p and ν_p can be used to control the width of objective curve bell. Using this Gaussian-based approach results in a much steeper function gradient at even moderate distances from the solution, resulting

in improved convergence speed.[1]

The peak height function is very similar to 3.6 above, with slightly different parameters:

$$O_2(\alpha) = \frac{1}{Npk}\sum_{i=1}^{Npk} 1 - \exp\left(\frac{-\mu_h \delta\xi_i^2}{\nu_h^2}\right) \tag{3.7}$$

where $\delta\xi_i = |\xi_i - \bar{\xi}_i|$ is the difference between the test and target heights of peak i. μ_h is again a strictness parameter, while ν_h is a windowing function based on magnitude rather than frequency. The full function is then defined by combining 3.6 and 3.7 and adding weights to each:

$$O(\alpha) = \frac{w_1 O_1 + w_2 O_2}{w_1 + w_2} \tag{3.8}$$

where the w_i are generally set between 0 and 1. Since the length of an instrument holds a greater influence than bore radius over the overall sound of an instrument, the default values of w_i are often set with more emphasis on peak location.

For the sake of clarity, it will help to visualise what had just been mathematically described. When tackling a two dimensional problem, such as a cylinder, the objective function has the typical (inverted) Gaussian shape with a large dip towards the minimum as the length coordinate (y) becomes closer to the optimal solution. The change of gradient in the radius (x) coordinate depends on the weighting assigned to 3.7.

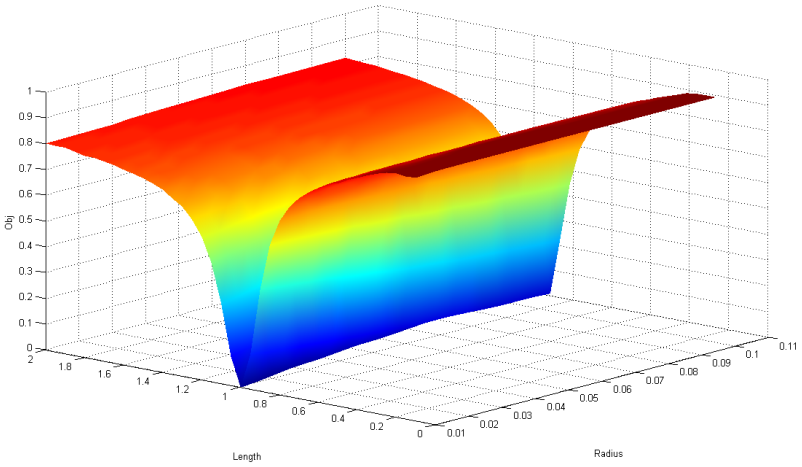

Figure 3.2: Objective Function based on length and radius, w = [1,0.2]. Optimal solution is at $\boldsymbol{\alpha}$ = [0.01,1]

Intuitively, increasing the number of instrument parameters increases the dimension of the problem and hence the complexity of the objective function. The implications are this are discussed in further details in section 4.2.

3.4 Implementation of Bore Optimiser

The code for the bore optimiser provides a relatively high degree of user customisable features. Users must specify the number of sections along with the corresponding number of variables that they wish the output bore to have. They also must define a vector containing the nature of each section type, using the number prefix system as described in 1.4.4. Using this approach results in the optimiser having to stick to the section order defined at the beginning; this is an advantage as it reduces the size of the solution space, but also a disadvantage as the order of sections may not be interchanged during the optimising process. Future work could attempt to rectify this issue, but how big an 'issue' it actually is depends on the readers preference between flexibility and efficiency. One further restriction is that bessel horns may only be placed at the end of an instrument, which is of course a justifiable action.

Above everything else, the user must input some target impedance data. For the time being, only data on peak locations and magnitudes is needed due to the nature of our objective function. Data for a 'complete curve' can be trimmed down by using Matlab's *peakfinder* function. Alternatively, the user may specify a target bore shape, from which a target curve can be calculated. This option has been used mostly for testing the optimiser. Note that scoring instrument designs on peak information alone eliminates the uniqueness of solutions. Starting parameters can either be defined by the user or generated randomly.

Optional inputs include specified bounds and step sizes for radii, lengths and flare constants, along with time and step thresholds for the termination of the optimiser. These are set to default values if not specified by the user. Care should be taken in setting the upper and lower bounds, as being too flexible may vastly increase calculation time and produce poor results, whilst being too restrictive risks ruling out reasonable solutions. In cases where the optimal solution falls outside of the solution space, the program will attempt to find as close a match as possible.[1]

Once the bore optimiser has terminated, it returns the optimised bore param-

[1]This, for example, could happen if we are estimating a detailed trombone as only a cylinder and bessel horn. Hence the impedance curve for the detailed trombone will be difficult to match exactly with only these sections.

eters $\boldsymbol{\alpha}$ and (if requested) various plots of the optimised impedance curve, target curve and optimised bore profile.

Chapter 4

Experiment and Analysis of the Bore Optimiser

Like with the impedance generator, we may now test the accuracy and robustness of the bore optimiser as well as discussing improvements and applications. The chapter begins by examining how the optimiser performs with single section bores, and then moves onto a more complex problem. Any issues or challenges are then discussed along with possible performance improving modifications. The optimiser is quite susceptible to changes in the algorithm parameters, hence much of the work presented in this chapter has been produced after lots of fine-tuning.

4.1 Accuracy & Performance Tests

4.1.1 Test 1: Optimising Single Sections

To verify the performance of the optimiser on a basic scale, it is tested on the constituent parts that are used to make up full bore profiles. What follows are a set of simple tests on a cylinder, cone and bessel horn. A target bore is set for each test, from which a target curve can be calculated using the impedance generator. All tests are carried out using the following parameters: $Npk = 10$, $\nu_p = 60$, $\nu_h = 60$, $\mu_p = 1$, $\mu_h = 5$, $a = 3$ and $b = 0.5$. Maximum weights of 1 are assigned to both the peak locations and peak heights, and the time threshold is set to 60 seconds. These values were all selected after a period of rigorous pre-testing. Step sizes have been set to rather large values, as suggested by the informal article in [27]. The reasoning behind this is that even if the search path falls into a local minimum, the step values will still be large enough to offer a good chance of finding a lower value of the objective function, allowing

the algorithm to proceed. Since the neighbourhood of points around a local minimum will generally be higher, small step sizes will only increase the chance of getting permanently stuck in a local minimum.

Convergence on the global solution may depend on the selection of the starting point - the further away the algorithm starts, the greater chance it has of getting stuck along the way. Therefore, the tests are run several times for each shape using randomly generated starting points within the appropriate bounds. (rounding?)

Cylinder

The bore optimiser was applied from 4 starting points to try and find the closest matching parameters for a target cylinder of radius 0.01m and length 1m. The results are summarised in table 4.1:

Starting Parameters	**Optimised Parameters**	**Time (Perfect Match)**
0.0231, 1.4125	0.01, 1.005	-
0.0672, 2.4157	0.01, 1.005	-
0.0152, 2.4294	0.019, 0.995	-
0.0672, 1.2649	0.01, 1	34

Table 4.1: Optimisation of Cylinder. Parameters = $[r_0, L]$

The optimiser clearly deals with cylinders well, which is the least we could have hoped for. It will often return a perfect solution well within 60 seconds; when this doesn't happen, it still gets within a few millimeters. Removing the time threshold altogether will almost always result in a perfect solution being attained. Regardless of starting point, the algorithm will often be within a close region of the optimal solution inside 10 seconds.

Cone

The same procedure was applied to a target cone with $r_0 = 10$mm, $r_1 = 40$mm and $L = 800$mm.

Starting Parameters	**Optimised Parameters**
32, 65, 2001	16, 66, 793
67, 48, 186	15, 63, 791
60, 66, 1729	10, 47, 801
54, 53, 1041	11, 41, 798

Table 4.2: Optimisation of Cone. Parameters = $[r_0, r_1, L]$

Table 4.2 generally demonstrates very good results, but overall accuracy is slightly decreased due to the increase in dimension of the problem. Regardless, some results still get very close to a perfect solution - row 4 of the table being a good example. The two worst performing tests seemed to be the ones with initial lengths at the extremities of the solution space (rows 1 and 2), verifying that distance of starting point can affect convergence rate and success. Lengths tend to converge quicker that radii, and become close to the solution very quickly. Bearing in mind the time threshold was still 60 seconds, the performance would likely improve if this threshold was lifted.

Bessel Horn

The parameters selected for the target bessel horn were $r0 = 20$mm, $r_1 = 100$mm, $L = 1000$mm and $\gamma = 0.7$.

Starting Parameters	**Optimised Parameters**
66, 72, 1911, 0.655	15, 90, 1013, 0.642
70, 137, 490, 0.624	15, 72, 1009, 0.665
52, 96, 917, 0.717	17, 96, 999, 0.757
26, 76, 712, 0.701	25, 84, 993, 0.714

Table 4.3: Optimisation of Bessel Horn. Parameters = [r_0, r_1, L, γ]

As expected, dealing with a 4-dimensional problem further affects the accuracy. The radii values are on average about 10mm out, while the flare constant never really shows any pattern of convergence - in tests 3 and 4 it actually diverges from the optimal point despite starting close to it. This may be because the flare of a bell can influence both peak position and magnitude, so the objective function essentially has a harder job to do when analysing γ. However, the length accuracy is still very good, coming within 1mm of the optimal solution in test 3. Additionally, it still converges very quickly to the optimal neighbourhood. This is a satisfying result when it is remembered that length is a formidable influence on the acoustics of an instrument. Again, longer time thresholds or further experimentation with the parameter values may yield improved results. Despite the inaccuracies, it is still worth applying the bore optimiser to a more complicated problem to see how far it can guide us towards an optimal solution.

4.1.2 Test 2: A More Challenging Optimisation

In practice the bore optimiser will largely be used on multi-section bores, so it is sensible to see how it performs with this more complicated task. Dealing with multiple bore sections of course further increases the dimension of the problem, and this will likely result in the plentiful identification of areas to be improved later.

The chosen target was the impedance curve for a 3-section bore made up of a concatenated cylinder, cone and bessel horn designed to roughly estimate the profile of the trumpet in section 2.2.2. This particular setting therefore involves 9 dimensions. All other core parameters remained the same as in 4.1.1, apart from the time threshold which was raised to 120 seconds. Starting parameters were again generated randomly.

A problem rapidly emerged with the multi-bore case, where some of the suggested designs were physically unsatisfactory or completely untenable for calculating a proper impedance curve, which can crash the program. Figure 4.1 is one such example. This problem stems from a lack of restrictions on 'jumps' in input and output radii in consecutive sections - the optimiser is free to create unrealistically large jumps, and the issue can be further compounded if the starting design is also unrealistic. In response, a 'jump limiter' was added into the code, which only permits very small jumps and enhances the smoothness of output designs.

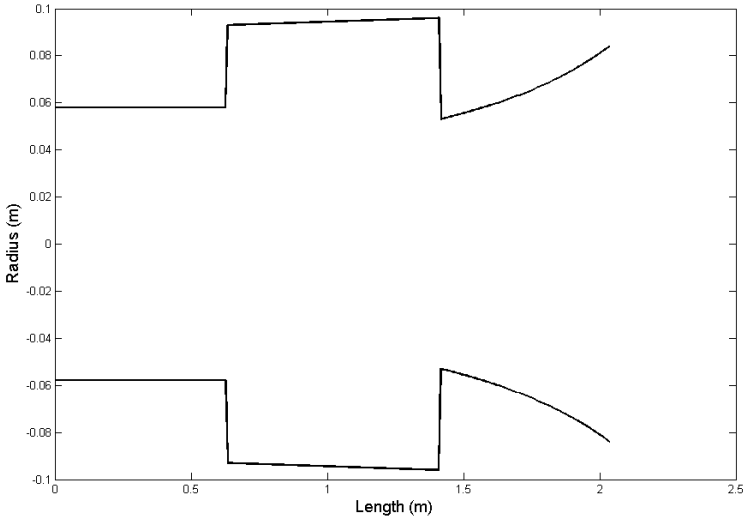

Figure 4.1: An Unsatisfactory Bore Design

When initiated with random starting points, the optimiser returned mixed results. Some results (eg. 4.2) are very good, taking a difficult, unrealistic original

design and converging towards the optimum reasonably well. Again, lengths appear to converge the best. However, in other cases the bore designs, despite being feasible, were still unsatisfactory - one common flaw was production of unusually large radius values on the left side of the tube - values of 30mm or more occurred often. Various other initialised values immediately got 'stuck', resulting in a culmination of failed steps, terminating the algorithm very quickly due the step sizes having been rapidly reduced below the threshold.

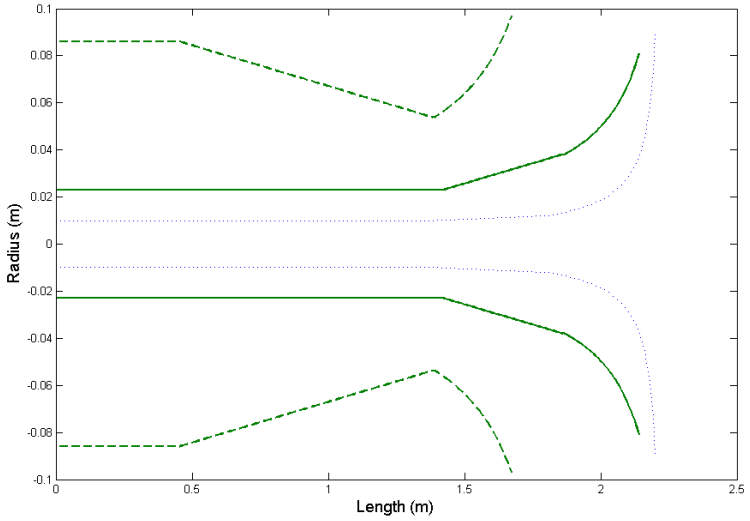

Figure 4.2: Bore Optimisation from randomly generated initial parameters. Dotted green line is starting design, solid green line is optimised design, dotted blue line is target bore.

Taking these observations into account paved the way for more improvements, this time related to the flexibility of the initial parameters. It is not unreasonable to assume that when optimising, we already have a good idea of the instrument type and hence the rough shape. This then motivates the introduction of templates of initial parameters,[1] which rule out the vast majority of the unusual cases previously talked about. Therefore, all further test results presented in this section were based on a generic 3-section trumpet template. Impedance and bore plots are presented in 4.3 and 4.4.

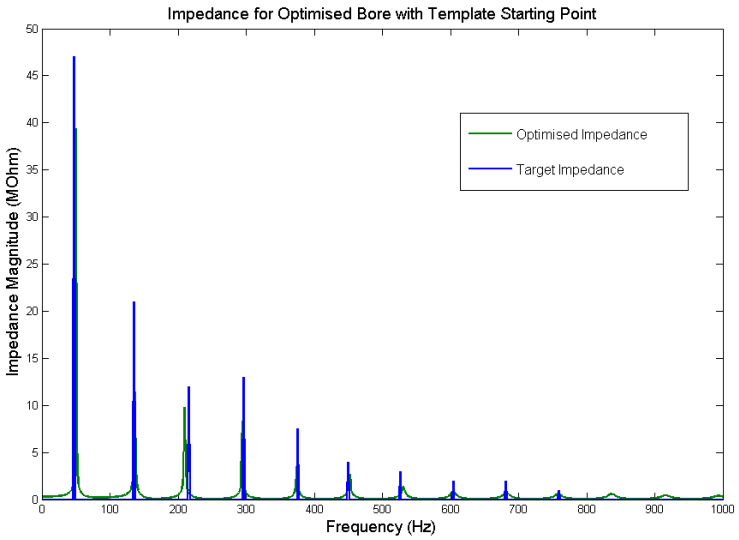

Figure 4.3: Impedance Curve for 3-section Bore Optimisation

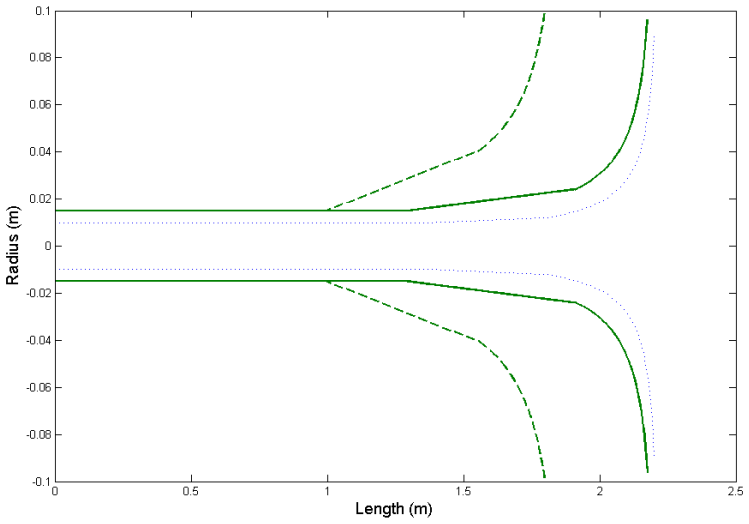

Figure 4.4: Bore Profile for 3-section Bore Optimisation

The optimiser performance seems to improve when using the template method. In figure 4.3 the peak locations match up quite well, which is of course always one of the primary objectives. The peak height accuracy is less impressive, which means convergence of radius parameters is still a sticking point, as confirmed in figure 4.4. Another notable trait of the optimiser is that it will rarely return the correct ratios of section lengths. This is only a minor observation however, as we are only interested in finding a bore that matches the impedance values - not the target bore that these values are obtained from. Given that many designs may

fit the specified data, we should bear in the mind that the target bore shape is only included to allow the reader gauge performance of the results.

As a last piece of analysis, table 4.4 shows numerical values placed on the accuracy of the first 5 peaks of the multi-bore case.

Table 4.4: Comparison of Optimised and Target values

Peak	**1**	**2**	**3**	**4**	**5**
Target Freq. (Mag)	47 (47)	135 (21)	216 (12)	296 (13)	376 (7.5)
Opt. Freq. (Mag)	49 (39)	136 (11)	210 (10)	295 (8)	375 (4)
Freq. Diff. (cents)	72	12	49	6	5
Mag. Diff (%)	17	48	17	38	47

4.1.3 Further Comments

Analysis results show that the bore optimiser can deliver reasonable results even in complex cases. However, it is far from perfect - so naturally we should be actively looking into how to improve it as much as possible. For the time being, the program can at least be used to gain a 'rough' design and further refinements can then be made manually by using the impedance generator.

If time permitted, it would be interesting to run the optimiser without a time threshold and see if it could make any more headway towards the optimal solution. The two minutes allocated for calculations for the mutli-bore case was really quite restrictive; certainly, similar experiments carried out in [1] took anywhere from 15 minutes to several hours to attain accuracy within 0.01mm.

Note that all the target impedance curves in this section were produced using the impedance generator and not from experimental BIAS measurements. Furthermore, the target designs were always within the solution space and so getting a perfect match was always theoretically achievable. If we do use experimental measurements, the target design will often be outside the solution space as it will be impossible to model it perfectly. In this case, the optimiser will still attempt to find a solution as close to the optimal curve as possible.

4.2 Difficulties & Improvements

As a result of the discussion about optimiser performance, the focus of the project has shifted from the practicalities of the program to offering explanations of its

shortcomings and various methods to improve the likelihood of complete convergence.

4.2.1 Sources of Difficulty

Objective Function

As already mentioned several times, with increase in bore complexity comes an increase in the dimension of the overall problem; the consequences being a tougher navigation space for the optimiser. It is the occurrence of local minima in the objective function that inhibit the algorithm from finding a perfect solution all the time, for if it finds itself in one then there are no immediate points in the neighbourhood that will lower the objective and allow the search to continue. As the complexity of the problem increases, so inevitably do the number of local minimum too. Measures have already been taken to try and avoid this happening; setting large default step sizes was one such measure.

Despite not being able to explicitly visualise any objective function above two dimensions, we can still gain an idea of higher-dimensional behaviour by examining 2D subspaces of $O(\boldsymbol{\alpha})$.[1]

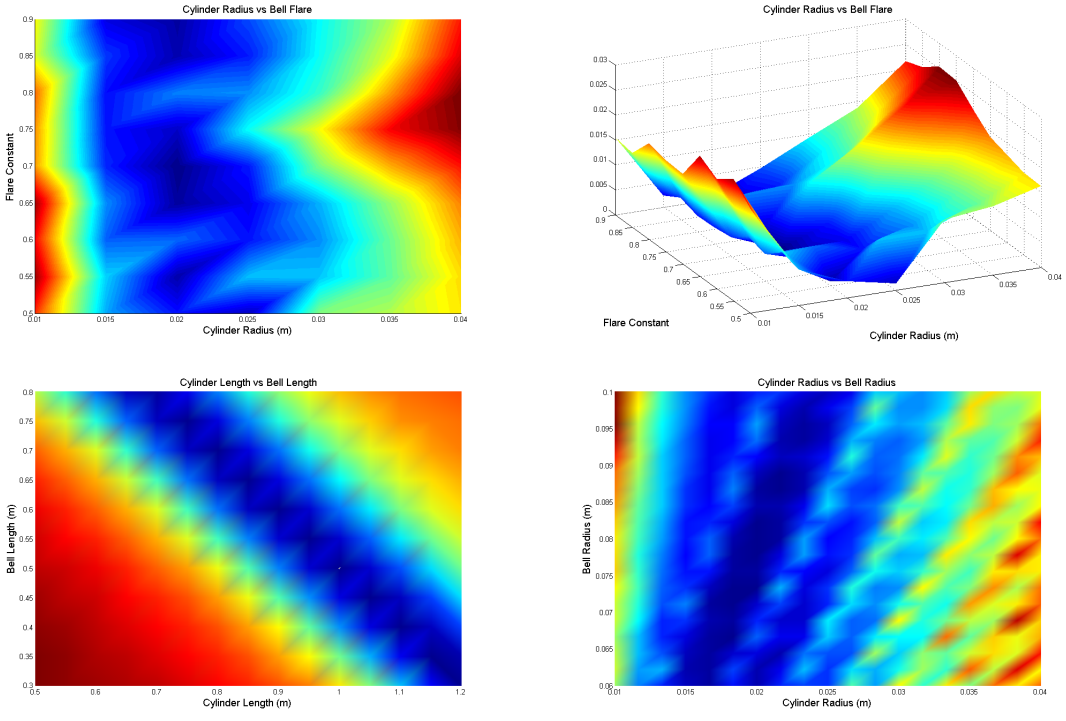

Figure 4.5: 2-dimensional Slices of 6-dimensional objective function. Top Line: Cylinder Radius vs Bell Flare; Bottom Line: Cylinder Length vs Bell Length, Cylinder Radius vs Bell Radius

Figure 4.5 shows several 2D subplots of a 6 dimensional problem made up of a cylinder and bessel horn. To produce each plot, four of the parameters have remained fixed while the two being examined are varied. It quickly emerges, as repeatedly predicted, that the objective function becomes much more complicated, with some subspaces revealing the presence of several local minima and non-smooth behaviour. Therefore, despite our desire to keep the objective function as smooth and simple as possible, non-convex behaviour is unavoidable when studying high-dimensional problems.

Initial Parameter Selection

The issue of initial parameter selection has already been well documented and addressed during the multi-bore test. Regardless, even with templates the initial parameter choice can still have a bearing on how well the algorithm will converge to the optimum. For example, in figure 4.4 the radius of the cylindrical section does not converge at all. This is presumably because the region of the objective function it is based in must be too complex to navigate out of.

4.2.2 Improvements/Future Work

Much work has already gone into improving the optimiser as we go along, but there are plenty more suggestions of areas that could be improved, or possible methods to utilize to enhance performance. Implementing these is beyond the scope and timescale of this project; nonetheless, they are summarised in this section, and have much potential to be used in future work to further improve the robustness of the optimiser.

Objective Function Improvements

The reader may have noticed that our objective function does not take into account any data on peak width and shape, which can further influence the acoustics of a final design. The reason for this was mainly because the curves produced by the impedance generator generally have poor shape definition, and hence would not contribute much useful data anyway. Regardless, if further improvements were made in the impedance generator, inclusion of a **least squares comparison** of all points on test and target curves could enhance the reliability of the optimiser. In the accompanying code, the objective function already includes a bypassed version of this, so it would be an easy feature to alter in the future. A further simple modification would be to introduce **weightings on individual**

peaks. In cases like plot 4.3, the lower impedance peaks will influence the overall sound much more than the upper ones. Therefore, prioritising matching of the lower peaks may help the algorithm converge faster, as it will spend less time 'fine tuning' the minor peaks.

Another more ambitious improvement would be to look at so called **smoothing algorithms**. These generally aim to create some sort of approximating function to capture the most important information in a data set. Intuitively, this could be applied to our objective function to try and iron out some of the local minima that hinder the optimiser. A good introduction to the theory along with some algorithms and a useful applet is located at [28].

Algorithm Improvements

There are a wealth of methods that could potentially improve the algorithm; here we will briefly look at two. The general idea is to try and provide ways for the optimiser to converge to as good a solution as possible.

The first technique is so-called **'brute force methods'**. The general idea is trivial; the solution space is scanned exhaustively in an attempt to find the optimal solution. It is not suggested that the method is used in this exact way, as it would take huge amounts of time to do a conclusive search. Rather, we could proceed in a similar way to 4.1.1, where a finite number of random points/templates are tested, the results logged and the closest solution selected. This is not a particularly imaginative fix; though it is simple to initiate and can occasionally lead to very good results.

A more sophisticated method is called **simulated annealing**. This is a stochastic method, which (generally speaking) sometimes allows the search algorithm to accept a *rise* in objective function if there is a good probability that the corresponding direction points towards the optimal solution. A detailed overview of the process is given in [29].

Permitted Bore Designs

The most notable restriction related to accepted bore designs was the lack of a mouthpiece structure. This absence is problematic when attempting to find a bore to match an impedance curve generated for a real instrument, as the magnitude 'boosting' effect of the mouthpiece cannot be replicated. In this situation the optimiser will try to compensate as much as possible, but inevitably cannot produce a near-perfect match. Mouthpiece modelling was briefly attempted using reversed cones, but problems arose when the optimiser would increase its length

to unrealistic values, rendering the whole exercise useless. To rectify this issue, it would be necessary in the future to develop a separate mouthpiece section with its own set of bounds.

Efficiency Improvements

Since the code performs hundreds of evaluations a minute, any improvements in efficiency are greatly sought after. One such improvement is related to step sizes. Currently, the default step sizes vary according to parameter, with length steps being at least an order of magnitude above radius steps. Normalising the steps to unit size for use in the optimiser (with de-normalisation for objective function calculations) is evidenced to offer better efficiency.[1]

On a more general note, there are undoubtedly many parts of the code where efficiency can be actively improved upon. Many adjustments and improvements carried out during the testing were implemented as rough patches. Given a bit more time and attention, the author is confident that parts of the code can be implemented with better finesse.

4.3 Practical Uses

Despite the main focus of this chapter having gradually veered towards detailed discussion of performance, drawbacks and improvement, it would still be nice to outline uses of the bore optimiser in its current guise. Therefore, to round off the project we will study a simple example where the optimiser is used to some sort of acoustical feature of an instrument.

4.3.1 Application: Shifting Pitch-Standard

This is a relatively basic task in which we simply aim to shift the pitch standard of an instrument down a semitone. This essentially involves asking the optimiser to reduce the frequency value of each impedance peak by about 50 cents. The motivation for doing this could come from the fact that there sometimes exist different pitch standards between acoustically identical instruments. For example, it is observed in [8] that 17th century and modern day cornetts operate acoustically in exactly the same way, but with a pitch standard about a semitone apart. Therefore, it may be interesting to determine how we could alter such a modern-day instrument to play at the same pitch as an older model, or vice versa.

A test 'instrument' was constructed from a cylinder and a bessel horn, with $\boldsymbol{\alpha} = [0.01,\ 1.5,\ 0.01\ ,0.09,\ 0.5,\ 0.7]$. The location of each frequency peak was noted and the value for each detuned peak calculated. This was then assigned as the target for the optimiser, with the bore design $\boldsymbol{\alpha}$ set as a starting point. The algorithm was run for 60 seconds with $Npk = 10$ and full weighting on both peak location and magnitudes, producing the results displayed in figure 4.6.

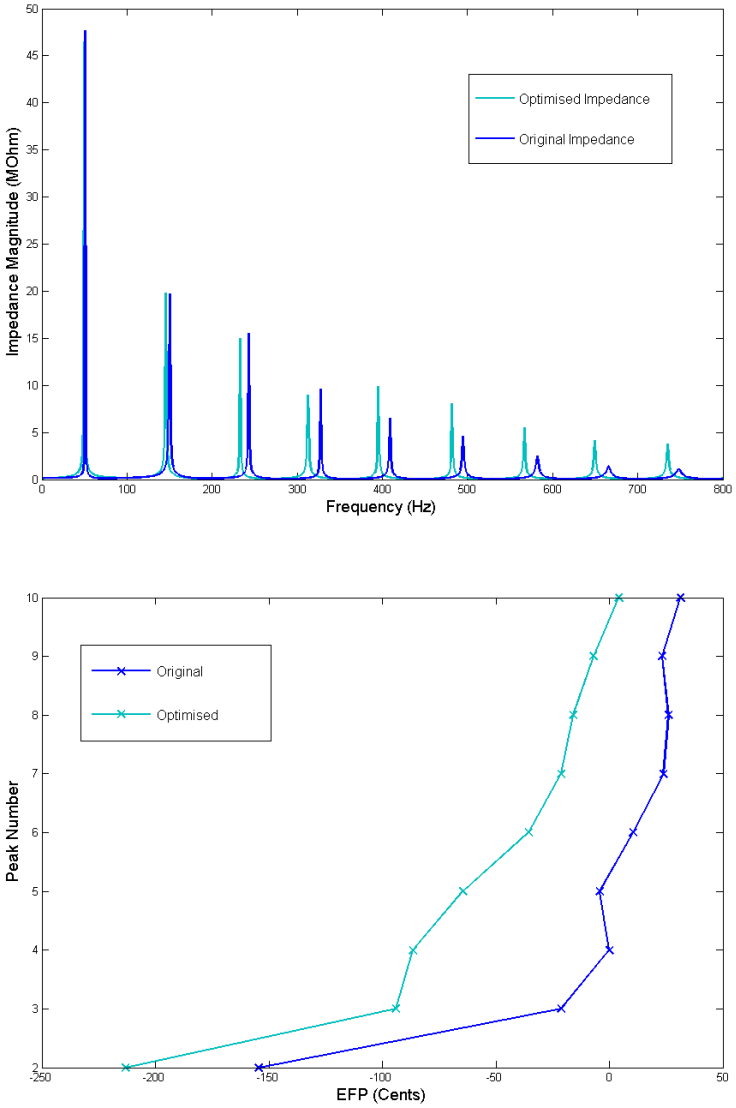

Figure 4.6: Impedance plot and EFP for detuned instrument. $f_0 = 82$ Hz

The impedance plot indicates that a reduction in pitch has resulted, but to get a proper idea of how close it is to our target we may examine another equivalent fundamental pitch plot. The results are good; the EFP clearly shows an approximate reduction of around a semitone for each peak, and the harmonic

relationship between peaks is almost perfectly preserved. Numerically, the majority of peaks were reduced between 40-60 cents in pitch, with extreme values of 86 (peak 4) and 27 (peak 10). The average reduction was 50.2. Therefore, the optimiser has been verified as a useful tool for simple design tasks like the one we have just covered.

4.3.2 Further Applications

The optimiser can potentially be used to perform many other design alterations, including magnitude reduction and individual peak tuning.[1] The latter may be motivated by the observation back in section 2.2.2 that a typical characteristic of many brass instruments is their production of a significantly flat second peak. We then may wish to use the bore optimiser to try and sharpen this peak and hence improve the overall harmonic alignment of the resonances. Performing this task effectively will require further experimentation, as preliminary tests showed that the objective function was often over-influenced by the comparatively large peak 1 or the target impedance was too close to the initial impedance, both of which restricting the optimiser's flexibility. Individual peak weighting would be a relevant solution in any future work on this problem.

Chapter 5

Conclusions & Appendices

This project originally set out to construct user-friendly software which could produce impedance curves from bore data and vice versa. In general, implementation of this aim was successful - but there is always room for improvement.

The impedance generator is robust and has reasonable accuracy. While it may not be suited to extremely fine precision, its applications can be extremely useful for learning more about the behaviour of a particular instrument. The accuracy of the bore optimiser needs plenty of improvement; regardless, it can still be usefully used to produce rough designs that can be fine tuned later. Despite the relatively simple theory, it was trickier to construct a reliable implementation of the optimiser. The reasons for this are well-documented. While constructing a super-accurate optimiser would have been a big ask in the time constraints available, it was still appealing enough to implement the algorithm and see how it performed. This provided a wealth of further motivation to explore the more intricate details of optimisation techniques,

Asides from the theory, implementation and application, one of the enduring ideas in this project is that the work may be constantly updated and improved in the future. In this vein, there have been extensive discussions on improvements needed to both the impedance generator and bore optimiser, and suggestions tabled on how to carry them out. Many of these could be tried and tested as part of a future project.

Appendix I

Finite Difference Scheme Operators

This is a glossary of all the operator notation for schemes used with finite difference methods. Spacial grid points and steps are denoted as l and h respectively, while the temporal equivalents are denoted as n and k. The identity operator is denoted as e with an attached subscript indicating whether to move forward or backward on the space or time axis of the grid being used. For example, $e_{x+}u_l^n = u_{l+1}^n$.

Spacial Operators

$\delta_{x+} \approx \frac{1}{h}(e_{x+} - 1)$

$\delta_{x-} \approx \frac{1}{h}(1 - e_{x-})$

$\delta_{x\cdot} \approx \frac{1}{2h}(e_{x+} - e_{x-})$

$\delta_{xx} = \delta_{x+}\delta_{x-} = \frac{1}{h^2}(e_{x+} - 2 + e_{x-})$

Temporal Operators

$\delta_{t+} \approx \frac{1}{k}(e_{t+} - 1)$

$\delta_{t-} \approx \frac{1}{k}(1 - e_{t-})$

$\delta_{t\cdot} \approx \frac{1}{2k}(e_{t+} - e_{t-})$

$\delta_{tt} = \delta_{t+}\delta_{t-} = \frac{1}{k^2}(e_{t+} - 2 + e_{t-})$

Averaging Operators

$\mu_{x+} \approx \frac{1}{2}(e_{x+} + 1)$

$\mu_{x-} \approx \frac{1}{2}(1 + e_{x-})$

$\mu_{x\cdot} \approx \frac{1}{2}(e_{x+} + e_{x-})$

$\mu_{xx} = \mu_{x+}\mu_{x-} = \frac{1}{4}(e_{x+} + 2 + e_{x-})$

Appendix II

Viscothermal Losses System

A lossy version of Webster's equation can be expressed as

$$S\Psi_{tt} = \gamma^2(S\Psi_x)_x - \epsilon S^{1/4} w_t \tag{5.1}$$

where

$$w_{tt} = \epsilon S^{1/4}\Psi_t - 2\sigma_0 w_t - \omega_0^2 w \tag{5.2}$$

and

$$\epsilon = c\sqrt{\frac{2\rho}{M}}\left(\frac{\pi}{S_0}\right)^{1/4} \tag{5.3}$$

M is the mass per unit area of the tube walls, ρ is the density of air, c is the speed of sound, σ_0 is a damping coefficient and ω_0 is a fundamental frequency parameter. For simplicity, M is set to equal 1. w is scaled with tube radius to represent the ratio of radius to the viscous and thermal boundary layers.[18]

A finite difference scheme for the above is given in [12]:

$$\mu_{xx} S\delta_{tt}\Psi = \gamma^2\delta_{x+}((\mu_{x-}S)\delta_{x-}\Psi) - \epsilon S^{1/4}\delta_{t\cdot} w \tag{5.4}$$

$$\delta_{tt} w + 2\sigma_0\delta_{t\cdot} w + \omega_0^2 w = \epsilon S^{1/4}\delta_{t\cdot}\Psi \tag{5.5}$$

Like with the lossless case, this can then be expanded and rearranged into a recurrence relation, which is clearly laid out in the code. The impedance generator uses this fully lossless model for all its calculations.

Appendix III

Detailed Bore Measurements for Section 2.2.2

Presented is a detailed description of all 11 bore sections used to reconstruct the detailed instrument in section 2.2.2. Identifier tags are used to refer to the type of shape used each time - to recap, 1 represents a cylinder, 2 a cone and 3 a bessel horn. All measurments are expressed in millimeters.

Section	**Type**	r_0	r_1	L	γ	**Comments**
1	2	18.17	16.44	5.27	-	Mouthpiece
2	2	16.44	4.22	16.53	-	Mouthpiece
3	2	4.22	7.39	48.64	-	Mouthpiece
4	2	7.39	9	25.55	-	
5	2	10.2	10.5	6	-	
6	1	10.5	-	27.9	-	
7	1	10.3	-	1405	-	Lead Pipe
8	1	10.8	-	124	-	
9	2	10.8	12.7	70	-	
10	2	12.7	14.88	140	-	
11	3	14.88	108	232	0.6	Flare

Total Length = 2100mm

Appendix IV: Source Code

The code provided here consists of the main scripts for the impedance generator and bore optimiser. This makes up the bulk of the overall program, but for clarity some of the more complicated or frequent processes were implemented as functions in their own right. As this hard copy of the code is only intended to act as a rough guide, these functions are not included here. Full descriptions and directions of use for the code are included with the digital copy.

Impedance Generator

Main Script

```
1
2 %%%%%%%%%%%%%%%%%%%%%%%%IMPEDANCE GENERATOR
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3
4 % Script      : Impedance Generator 2.1
5 % Written by : Craig Meek (s0791739@sms.ed.ac.uk)
6 % Created on : 24/06/2012
7 % Last Update: 16/08/2012
8 % Purpose     : Returns impedance curves for input bore
    profiles
9 %
10 %
11 % Description: User specifies total instrument length
    and number of
12 %              sections. The script then prompts the
    user to enter further
13 %              information about each section. Each
    section must be defined
14 %              with an identification number as the
    first entry in the
15 %              vector. 1 = Cylinder, 2 = Cone, 3 =
    Bessel Horn.
16 %
17 %              Example: Cone = [2, r0, r1, d]
18 %
19 %              See the file descriptions for more
    detail.
20 %
21 %
22 %
23 %
24 %-------------------------------------------------------
    -------------------%
25 clear
26
```

```
%---User Input---------------------------------------
------------------%

L = 1.5; %Total Bore Length
NS = 3; %Number of Bore Profile Sections


%---Global Parameters--------------------------------
------------------%

Fs = 44100; %Sample Rate (Hz)
c  = 347; %Sound Velocity at 27C (m/s)
tf = 1; %Simulation Duration (s)

NF = round(tf*Fs); %Total number of required time
    samples
gamma = c/L;
k = 1/Fs; %Time Step

h = gamma*k; N = floor(1/h);
h = 1/N; %Spacial Step
lambda = gamma*k/h; %Courant Number


%---Instrument Profile-------------------------------
------------------%

R = bore_construction(NS,N,L); %Prompts user to define
    each bore section
%R = ones(1,N+1);               %Use this line for non-
    prompted definition

tic

S = pi.*(R.^2)'; %Bore surface area
S0 = S(1);       %Left hand surface area
S = S/S(1);      %Scaling S

Sav = [S(1); 0.25*(S(3:N+1)+2*S(2:N)+S(1:N-1)); S(N+1)
    ]; %Averaging S


%---Initialisation-----------------------------------
------------------%

%Initialise Grid Functions and Output Vectors
Psi = zeros(N+1,1);
Psi1 = zeros(N+1,1);
Psi2 = zeros(N+1,1);

w = zeros(N+1,1);
w1 = zeros(N+1,1);
w2 = zeros(N+1,1);
```

```
73 pre = zeros(NF,1); %To store impedance reading at
       mouthpiece
74 y = zeros(NF,1); %Output Vector (for playing sound)
75
76 %Input Vector (Impulse)
77 u = zeros(NF,1);
78 u(1) = 0.5;
79
80 %Scheme Templates
81 sr = 0.5*lambda^2*((S(2:N)+S(3:N+1))./Sav(2:N)); %
       scheme_right
82 sl = 0.5*lambda^2*((S(2:N)+S(1:N-1))./Sav(2:N)); %
       scheme_left
83 s0 = 2*(1-lambda^2); %scheme_central
84
85
86 %---Boundary Conditions-----------------------------
       ----%
87
88 %q1,q2,r1,r2 used to construct the Dirichlet & Neumann
       Boundary Conditions
89
90 alf = L/(0.6133*sqrt(S0*S(N+1)/pi)); %Loss Coefficients
91 bet = 0.6647/gamma;
92
93 Sr = 1.5*S(N+1)-0.5*S(N); %Average of right end surface
       area
94
95 q1 = alf*gamma^2*k^2*Sr/(Sav(N+1)*h); q2 = bet*gamma^2*
       k*Sr/(Sav(N+1)*h);
96 r1 = 2*lambda^2/(1+q1+q2); r2 = -(1+q1-q2)/(1+q1+q2);
97
98
99 %---(Viscothermal) Loss Parameters-----------------
       ----------%
100
101 rho = 1.1769e-3; %Density
102 ep = c*sqrt(2*rho)*(pi/S0)^0.25; %Coupling coefficient
103 sig = 0.6; %Damping coefficient
104 w0 = 100; %Fundamental frequency parameter
105
106 Sp = S.^0.25;
107
108 %Calculation of parameters
109 a0 = ep*k/2;
110 a1 = (ep^2*k^2)/(4*(1+k*sig));
111 a3 = a0*((2-k^2*w0^2)/(1+k*sig));
112 a4 = a0 - a0*((k*sig-1)/(1+k*sig));
113
114
115 a1 = (Sp(2:N).*a1);
116 a2 = -1+(Sp(2:N).*a1);
117 a3 = Sp(2:N).*a3;
118 a4 = Sp(1:N-1).*a4;
```

```
119
120 a1 = 1./(1+a1);
121
122
123 %---Main Loop-----------------------------------------
    -------------------%
124
125 for n = 1:NF;
126
127     %Recursive Relation
128     Psi(2:N) = s0*Psi1(2:N) + sl.*Psi1(1:N-1) + sr.*
        Psi1(3:N+1) + a2.*Psi2(2:N) - a3.*w1(2:N) + a4.*
        w2(1:N-1);
129     Psi(2:N) = a1.*(Psi(2:N));
130
131
132     %Boundary Conditions
133     Psi(N+1) = r1*Psi1(N) + r2*Psi2(N+1);
134     Psi(1) = s0*Psi1(1) + 2*lambda^2*Psi1(2) - Psi2(1)
        + u(n);
135
136     w(N+1) = w(N);
137     w(1) = 1;
138
139     %Pressure reading for output end
140     y(n) = Fs*(Psi(N+1) - Psi1(N+1));
141
142     %Pressure reading for mouthpiece end
143     pre(n) = Fs*(Psi(1) - Psi1(1));
144
145     %Update of Grid Varibles
146     Psi2 = Psi1; Psi1 = Psi;
147     w2 = w1; w1 = w;
148
149 end
150
151 Z = abs(fft(pre)); %Input impedance
152 Z = Z*10^-5;
153 Z = Z(1:20000); %Limiting range to 20kHz
154
155 %---Plots----------------------------------------------
    -------------------%
156
157 x = linspace(0,L,length(S));
158 f = [0:NF-1]*Fs/NF; %Frequency
159 f = f(1:20000);
160
161 %Plot 2D Bore Profile
162 figure
163 plot(x,sqrt(S),'k',x,-sqrt(S),'k');
164 axis([0 L -sqrt(S(end))-1 sqrt(S(end))+1])
165 xlabel('(Scaled) Length, m')
166 ylabel('sqrt(S),m^2')
167 title('Bore Profile')
```

```
168
169 figure
170
171 %Plotting Logarithmic Impedance Curve
172 subplot(2,1,1)
173 lZ = 10*log10(Z);
174 plot(f, lZ,'b')
175 axis([0 1000 min(lZ)-5 max(lZ)+5])
176
177 xlabel('Frequency (Hz)')
178 ylabel('Z MOhm, dB')
179 title('Input Impedance')
180
181
182 %Plotting Linear Impedance Curve
183 subplot(2,1,2)
184 plot(f, Z,'b')
185 axis([0 1000 0 max(Z)+10])
186 xlabel('Frequency (Hz)')
187 ylabel('Z MOhm, dB')
188 title('Input Impedance')
189
190 %Plot 3D Bore Profile
191 figure
192
193 NR = length(R);
194 y1 = zeros(20,NR);
195 z = zeros(20,NR);
196
197 theta=linspace(0,2*pi,20);
198
199 for n = 1:20
200
201     y1(n,:) = R*cos(theta(n));
202     z(n,:) = R*sin(theta(n));
203
204 end
205
206 surf(x,y1,z)
207 axis([0 L -0.2 0.2 -0.2 0.2])
208 colormap(cool)
209
210 toc
211
212
213 %-------------------------------------------------------
      -------------------%
214 % Versions  : 1.0 - 1st successful implemenation of
      Impedance Generator
215 %           : 2.0 - Modified to model viscothermal
      losses.
216 %           : 2.1 - 3D bore profile plots added
217 %-------------------------------------------------------
      -------------------%
```

Bore Optimiser

Main Script

```
%%%%%%%%%%%%%%%%%%%%%%%BORE OPTIMISER
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Script      : Bore Optimiser 2.4
% Written by  : Craig Meek (c.meek@gmail.com)
% Created on  : 11/07/2012
% Last Update: 16/08/2012
% Purpose     : Returns suggested bore profiles for user
    -specified impedance
%               curves.
%
%
% Description: User specifies total instrument length,
    number of
%               sections, the nature of each section,
    the number of peaks
%               to be tested and the target impedance
    data (if desired). The
%               script then prompts the user for their
    preferences on how to
%               implement the algorithm. Unfamiliar
    users are recommended to
%               read the description of files, which
    explains the various
%               input methods.
%
%
%
%------------------------------------------------------------
    -------------------%
clear
close all


%---Preliminary Inputs-----------------------------------------
    ------------------%

NS = 2; %Number of desired bore sections
Nv = 6; %Number of variables
Npk = 10; %Number of peaks to use with objective
    function

section_type = [1,3]; %Order of section types

%Target impedance data (if using)
Zt_loc = [49 146 236 319 397 481 565 647 727 811]; %
    peak location
Zt_peaks = [47 19 15 10 6 5 3 1 1 1]; %peak height

```

```
38
39
40 %---Preliminary Optional Inputs------------------------
       -------------------%
41
42 radius_bounds = [0.005,0.1];
43 length_bounds = [0.1,2.5];
44 bessel_bounds = [0.6,0.8];
45
46 radius_step = 0.5;
47 length_step = 1;
48 bessel_step = 0.5;
49
50 time_thresh = 60;
51 step_thresh = 1e-5;
52 jump_limit = 0.001; %Largest permitted radius `jump'
       between sections
53
54 a = 3; %Step multiplication parameter (Success)
55 b = 0.5; %Step multiplication parameter (Fail)
56
57 Wt = [1,1]; %Weighting for objective function
58
59 rd = 1000; %Rounding parameter (1000 = 3 d.p)
60
61
62 %---Prompted Input-(Do not edit below here)------------
       -------------------%
63
64 %Selecting a target impedance curve or target bore
65 INPUT = input(sprintf('Would you like to set a target
       impedance or a target bore? 1 = impedance, 2 = bore:
       '));
66
67 %-----------------------------------------------------
       -------------------%
68 if INPUT == 1 %Target impedance
69
70    Zt = zeros(1,22050); %Initialising target impedance
         vector
71    Zt(Zt_loc) = Zt_peaks;
72    TL = 0;
73    TR = 0;
74
75    %Defining the nature of the initial parameters
76    START = input(sprintf('Initial Parameters? 1 =
         Random, 2 = Manual:'));
77
78       if START == 1 %Random initial parameters
79
80         Alp = initial_point_generator(section_type,
              radius_bounds,length_bounds,bessel_bounds,NS
              ,Nv);
```

```
81          Alp = jump_limiter(section_type,Alp,NS,
                jump_limit);
82          StartParam = Alp;
83
84        elseif START == 2 %Manually input initial
              parameters
85
86          Alp = input(sprintf('Input starting point:'))';
87          Alp = jump_limiter(section_type,Alp,NS,
                jump_limit);
88          StartParam = Alp;
89
90        end
91
92
93      %Calculating initial impedance values and objective
            function rating
94      L = length_generator(section_type,Alp,NS);
95      IR = radius_generator(section_type,Alp,NS,L);
96      Z = impedance_generator_opt(L,IR);
97
98      thresh = ones(1,Nv);
99
100     Obj = objective_funct(Z,Zt,Npk,Wt); %Initial
            objective value
101
102
103   elseif INPUT == 2 %Target bore
104
105     %Defining target bore
106     Target = input(sprintf('Input target bore
            dimensions:'))';
107
108
109     START = input(sprintf('Starting Point? 1=Random, 2=
            Manual:'));
110
111       if START == 1 %Random initial parameters
112
113         Alp = initial_point_generator(section_type,
                radius_bounds,length_bounds,bessel_bounds,
                NS,Nv);
114         Alp = jump_limiter(section_type,Alp,NS,
                jump_limit);
115         StartParam = Alp;
116
117       elseif START == 2 %Manually input initial
              parameters
118
119         Alp = input(sprintf('Input starting point:'))
                ';
120         Alp = jump_limiter(section_type,Alp,NS,
                jump_limit);
121         StartParam = Alp;
```

```
122
123         end
124
125       Alp = round((Alp*rd))/rd;
126
127       %Calculating initial impedance values and
              objective function rating
128       L = length_generator(section_type,Alp,NS);
129       IR = radius_generator(section_type,Alp,NS,L);
130       Z = impedance_generator_opt(L,IR);
131
132       %Calculating target impedance values
133       TL = length_generator(section_type,Target,NS);
134       TR = radius_generator(section_type,Target,NS,TL);
135       Zt = impedance_generator_opt(TL,TR);
136
137
138       NF = length(Zt);
139       thresh = Alp - Target; %Step threshold vector
140
141 Obj = objective_funct(Z,Zt,Npk,Wt); %Initial objective
      value
142
143 end
144
145 %-------------------------------------------------------
      --------------------
146
147
148 %---Derived Parameters---------------------------------
      --------------------%
149
150 %Upper and lower bounds
151 bounds = bounds_generator(section_type,length_bounds,
      radius_bounds,bessel_bounds,NS,Nv);
152 lower      = bounds(:,1);
153 upper      = bounds(:,2);
154
155 %Step sizes for each dimension
156 stepsize = step_generator(section_type,length_step,
      radius_step,bessel_step,NS,Nv);
157
158 %Direction matrix
159 D = eye(Nv);
160
161
162 %-------------------------------------------------------
      --------------------%
163 %---MAIN ALGORITHM-------------------------------------
      --------------------%
164
165 tic
166
```

```
%---Exploratory Stage-----------------------------------
    -------------------%

while max(abs(thresh)) > step_thresh | max(abs(l)) >
    step_thresh | toc < time_thresh %Termination
    criteria


X=zeros(2,Nv); %Vector to keep track of success/failure
     in each direction
lambda = zeros(1,Nv); %Vector to hold sum of successful
     steps
l = stepsize;


  while nnz(X) ≠ numel(X) %Won't exit the loop until
      success and failure in each direction

    for ii = 1:Nv

        Alp_temp = Alp; %Storing current value of
            alpha
        Alp = Alp + l(ii)*D(:,ii); %Exploring new
            bore design

        Alp = round((Alp*rd))/rd; %Rounding

        Alp = jump_limiter(section_type,Alp,NS,
            jump_limit); %Restricting any jumps over
            threshold
        L = length_generator(section_type,Alp,NS); %
            Calculating length of new design
        R = radius_generator(section_type,Alp,NS,L);
            %Calculating new radius values


        thresh(ii) = Alp(ii) - Alp_temp(ii); %
            Updating threshold vector


%---Checking if new design is outside bounds
        bounds_test = zeros(1,Nv);

         for jj = 1:Nv

            if  Alp(jj) > upper(jj) | Alp(jj) <
                lower(jj)

                bounds_test(jj) = 1;

            else

                bounds_test(jj) = 0;
```

```
207                 end
208
209             end
210
211
212             if nnz(bounds_test) > 0 | R == 1;
213
214                 %Design unacceptable; outside bounds
215                 Obj_1 = Inf;
216
217             else
218
219                 %Design acceptable; calculating new
                        impedance and rating
220                 Zn = impedance_generator_opt(L,R);
221                 Obj_1 = objective_funct(Zn,Zt,Npk,Wt)
                        ;
222
223             end
224
225
226 %---Determining success or failure of new design
227             if Obj_1 ≤ Obj
228
229                 l(ii) = a * l(ii); %Success;
                        increasing step size
230                 Obj = Obj_1;
231
232                 disp('Improved Design:')
233                 Alp
234
235                 lambda(ii) = lambda(ii) + l(ii); %
                        Updating total displacement
236                 X(1,ii) = 1; %Logging success
237
238             else
239
240                 l(ii) = -b * l(ii); %Failure;
                        decreasing step size & reversing
                        direction
241                 Alp = Alp_temp; %Reverting to old
                        design
242
243                 X(2,ii) = 1; %Logging fail
244
245             end
246
247
248
249       end
250 %---End of exploration loop; this keeps running until
      every entry in X is 1
251 %   (ie. Success and fail in each direction)
252
```

```
253
254 %---Mid-algorithm termination criteria
255     if max(abs(l)) < step_thresh
256
257         disp('Algorithm Terminated: Step size below
                threshold')
258         disp('Initial Parameters:')
259         StartParam
260
261         disp('Optimised Design:')
262         Alp
263
264         toc
265
266         opt_plot(Zn,Zt,R,L,TL,TR,IR) %Plotting
                results
267         return
268
269     elseif toc > time_thresh
270
271         disp('Algorithm Terminated: Time Threshold
                Met')
272         disp('Initial Parameters:')
273         StartParam
274
275         disp('Optimised Design:')
276         Alp
277
278         toc
279
280         opt_plot(Zn,Zt,R,L,TL,TR,IR) %Plotting
                results
281
282         return
283
284     elseif Obj == 0
285
286         disp('Perfect Match!')
287         disp('Starting Parameters:')
288         StartParam
289
290         disp('Optimised Design:')
291         Alp
292
293         toc
294
295         opt_plot(Zn,Zt,R,L,TL,TR,IR) %Plotting
                results
296
297         return
298
299     else
300
301     end
```

```
302
303
304     end
305
306
307
308 %---Orthogonalisation Stage---------------------------------------------%
309
310
311 %---Preamble
312 LD = zeros(Nv); %Step matrix
313 D_store = D; %Storing distance matrix
314
315
316 %---Filling in step matrix
317     for ii = 1:Nv
318         for kk = ii:Nv
319             LD(:,ii) = LD(:,ii) + lambda(kk) * D(:,kk);
320         end
321     end
322
323
324 %---Calculating new direction vectors
325 D(:,1) = LD(:,1) / norm(LD(:,1)); %First direction
326
327
328     %Palmer orthogonalisation up to Nv-1
329     for ii = 2:Nv-1
330
331
332         if lambda(ii-1) == 0;
333
334             %Avoiding the zero denominator case
335             D(:,ii) = -D_store(:,ii-1);
336
337         else
338
339
340             numerator = (lambda(ii-1) * LD(:,ii)) - (
                D_store(:,ii-1) * norm(LD(:,ii))^2);
341             denominator = norm(LD(:,ii-1)) * norm(LD(:,ii))
                ;
342
343             D(:,ii) = numerator./denominator;
344
345         end
346
347     end
348
349
350     %Palmer orthogonalisation for Nv - a patch to avoid
          zero denominator
```

```
351 if lambda(Nv) == 0
352
353     D(:,Nv) = D_store(:,Nv);
354
355 elseif lambda(Nv-1) == 0
356
357     D(:,Nv) = -D_store(:,Nv-1);
358
359 else
360
361     numerator = (lambda(Nv-1) * LD(:,Nv)) - (D_store
        (:,Nv-1) * norm(LD(:,Nv))^2);
362     denominator = norm(LD(:,Nv-1)) * norm(LD(:,Nv));
363
364     D(:,Nv) = numerator./denominator;
365
366 end
367
368 disp('Orthogonalisation complete')
369
370 end
371
372 toc
373 disp('Algorithm Terminated; Threshold met')
374 disp('Initial Parameters:')
375 StartParam
376
377 disp('Optimised Design:')
378 Alp
379
380
381 %--Plot Results
382 opt_plot(Zn,Zt,R,L,TL,TR,IR)
383
384
385 %-----------------------------------------------------
      ----------------------%
386 % Versions  : 1.0 - 1st successful implemenation of
        Bore Optimiser
387 %             : 2.0 - Modified to produce plots.
388 %             : 2.1 - lambda(Nv) = 0 patch added.
389 %             : 2.2 - Radius, lengths, bounds and step
        generators added
390 %             : 2.3 - jump_section patch added
391 %             : 2.4 - Rounding parameter added
392 %-----------------------------------------------------
      ----------------------%
```

Bibliography

[1] Braden, A.C.P (2006) *Bore Optimisation and Impedance Modelling of Brass Musical Instruments* PhD Thesis, University of Edinburgh.

[2] Braden, A.C.P (2005) *Optimisation Techniques for Solving Design Problems in Modern Trombones* (Proceedings of Forum Acusticum, Budapest, pp.2569-2572)

[3] Noreland, D.J.O (2003) *A Gradient Based Optimisation Algorithm for the Design of Brass-Wind Instruments* PhD Thesis, Department of Information Technology, Uppsala University.

[4] Noreland, D.J.O.; Udawalpola, M.R.; Berggren, M.O. (2010) *A Hybrid Scheme for Bore Design Optimization of a Brass Instrument* Journal of the Acoustical Society of America, Vol 128, No. 3 (Sept. 2010) pp.1391-1400

[5] Kausel, W (1999) *Computer Optimization of Brass Wind Instruments* Institut für Wiener Klangstil, Vienna, Austria. Available at: http://iwk.mdw.ac.at/mitarbeiter/english/wk/paper%20diderot.pdf [accessed 27 June 2012]

[6] Amir, N.; Shimony, U.; Rosenhouse, G (1995) *A Discrete Model for Tubular Acoustic Systems with Varying Cross Section - The Direct and Inverse Problems* Acustica, Vol 81, No.5 (Sept. 1995) pp.450-462

[7] Campbell, Murray; Greated, Clive (1987) *The Musician's Guide to Acoustics* (Ch9: Brass Instruments, pp.303-366). Cambridge Unversity Press

[8] Campbell, Murray (1996) *Cornett Acoustics: Some Experimental Studies* The Galpin Society Journal, Vol 49 (Mar. 1996) pp.180-196. Available at: http://www.jstor.org/stable/842398 [accessed 23 March 2012]

[9] Fletcher, Neville H.; Rossing, Thomas D. (1991) *The Physics of Musical Instruments* (Ch14: Lip Driven Brass Instruments, pp.365-375, 383-384). Cambridge University Press

[10] Wolfe, Joe. (2012) *What is Acoustic Impedance?* (Music Acoustics, University New South Wales). Available at: http://www.phys.unsw.edu.au/jw/z.html [accessed 10 March 2012]

[11] Braden, Alistair C.P; Newton, Michael J; Campbell, Murray (2005) *Optimising the Harmonicity of Trombones*. University of Edinburgh

[12] Bilbao, Stefan (2009) *Numerical Sound Synthesis* (Ch5: Grid functions and finite difference operators in 1D, pp.93-105; Ch9: Acoustic Tubes, pp.249-278). John Wiley & Sons, Ltd

[13] Bilbao, Stefan (2011) *Time Domain Simulation of Brass Instruments* University of Edinburgh. Available at: http://edinburgh.academia.edu/StefanBilbao/Papers/1014125/Time_Domain_Simulation_of_Brass_Instruments [accessed 14 Apr 2012]

[14] Webster, A.G (1919) *Acoustical impedance, and the theory of horns and the phonograph* (Proceedings of the National Academy of Sciences of the United States of America, pp.275-282)

[15] Silva, F.; Guillemain, P.; Kergomard, B.; Mallaroni, B.; Norris, A. (2009) *Approximation formulae for the acoustic radiation impedance of a cylindrical pope* (Journal of Sound and Vibration, Vol 322, pp.255-263)

[16] Zhilin Li (2001) *Finite Difference Methods Basics* (Computational Mathematics: Models, Methods and Analysis.) Center for Research in Scientific Computation, North Carolina State University. Available at: http://www4.ncsu.edu/~zhilin/TEACHING/MA402/notes1.pdf [accessed 31 July 2012]

[17] Wogram, Klaus (1972) *The Summation Principle* (A Contribution to the Measurement of the Intonation of Brass Instruments, Ch 3.2.3). Carolo-Wilhelmina Technical University of Braunschweig

[18] Keefe, Douglas (1984) *Acoustical wave propagation in cylindrical ducts: Transmission line parameter approximations for isothermal and nonisothermal boundary conditions* Journal of the Acoustical Society of America, Vol 75, no. 1, pp.58-62.

[19] Meek, C (2012) *An Initial Study into the Acoustics of the Serpent*, MSc mini-project, University of Edinburgh.

[20] Backus, J (1976) *Input impedance curves for the brass instruments* J. Acoust. Soc America, Vol 60, No. 2 pp.470-480.

[21] Gondzio, Jacek (2012) *Convex Optimisation*, from MATH11044 Practical & Large Scale Optimisation. School of Mathematics, Edinburgh University.

[22] Rosenbrock, H.H (1960) *An automatic method for finding the greatest or least value of a function* The Computer Journal, Vol.4, pp.175-184.

[23] Pošík, Petr (2001) *Rosenbrock's Algorithm: Should We Reset the Multipliers After Each Coordinate System Update?* Dept. of Cybernetics, Czech Technical University, Prague. Available at: http://labe.felk.cvut.cz/~posik/papers/Rosenbrock/RosOrigVsRosMod.pdf [accessed 20 July 2012]

[24] Palmer, J.R (1969) *An improved procedure for orthogonalising the search vectors in Rosenbrock's and Swann's direct search optimisation methods* The Computer Jorunal, Vol.12, No.1, pp.69-71.

[25] Weisstein, Eric W. (2012) *Gram-Schmidt Orthonormalization* From MathWorld–A Wolfram Web Resource. Available at: http://mathworld.wolfram.com/Gram-SchmidtOrthonormalization.html [accessed 29 July 2012]

[26] Weisstein, Eric W. (2012) *Gaussian Function* From MathWorld–A Wolfram Web Resource. Available at: http://mathworld.wolfram.com/GaussianFunction.html [accessed 30 July 2012]

[27] Author Unknown. (2007) *Rosenbrock Method* Available at: http://hi.baidu.com/fwso/blog/item/25bc2b3f7b3ce4ee54e72396.html [accessed 23 July 2012]

[28] Efstathiou, C. E. *Signal Smoothing Algorithms* Department of Chemistry, University of Athens. Available at: http://www.chem.uoa.gr/applets/appletsmooth/appl_smooth2.html [accessed 8 Aug 2012]

[29] Kirkpatrick, S. Gelatt, C. D., Vecchi, M. P. (1983) *Optimization by Simulated Annealing* Science, Vol.220, No.4598, pp.671-680. Available at: http://www.fisica.uniud.it/~ercolessi/MC/kgv1983.pdf [accessed 8 Aug 2012]

[30] Kausel, W. *Brass Instrument Analysis System.* http://www.bias.at

[31] *'Levels' & 'Fourier' Analysis Software* Resources for Acoustics, University of Edinburgh. Available at: http://www2.ph.ed.ac.uk/acoustics/teaching/acoustics/index.html