1995

Modelling and visualisation of wave propagation for echolocation

Shao-Min Zhu
University of Wollongong

Recommended Citation
NOTE

This online version of the thesis may have different page formatting and pagination from the paper copy held in the University of Wollongong Library.

UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site. You are reminded of the following:

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Modelling and Visualisation of Wave Propagation for Echolocation

A thesis submitted in fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

(Computer Science)

from

UNIVERSITY OF WOLLONGONG

by


Department of Computer Science

1995
Declaration

This is to certify that the work presented in this thesis was carried out by the author in the Department of Computer Science at the University of Wollongong and has not been submitted for a degree to any other university or institution.

Shao-Min Zhu
Abstract

The research reported in this thesis concerns modelling the behaviour of acoustic waves in echolocation, and subsequent visualisation on a computer screen. The aim of this work is to assist researchers to study the motion of waves. Three simulation models are reviewed, two discrete and one continuous. The discrete models include the Lattice Gas model and the Transmission Line Matrix model. The continuous model discussed here is based on mathematical equations. The features of the three models are studied and compared. Their problems are specified and some solutions are proposed. Finally, visualisations of several common robotic sensing situations utilising these three models are presented.

The contributions of this thesis are in (i) the 2D implementation, characterisation and extension of the Transmission Line Matrix (TLM) and Lattice Gas (LG) models of lossless acoustic wave propagation and (ii) the modelling of specular reflection off curved single curved surfaces and (iii) modelling of multiple reflection paths.
Acknowledgments

The author would like to thank her supervisor Dr Phillip McKerrow for his guidance, encouragement and support; and Dr Greg Doherty for his help with the mathematics. The author also wishes to express her gratitude to colleagues in her research group, especially to Mr Neil Harper for proof-reading the draft of this thesis; and Mr Pushkar Piggott for his many valuable suggestions. Thanks also to Mr Ben Stanley and his wife Mrs Tammy Stanley for their review of the final draft of this thesis for both physics and written English (except Chapter 6).

Thanks are also due to the University of Wollongong, for providing the University Postgraduate Research Awards scholarship, and Apple Computer Australia for providing the Quadra 950 workstation through their Apple University Development Fund.

Finally, the author must thank her elder brother Mr Xiao-Dong Zhu and his wife Mrs Xiao-Hong Wang for generous financial support at the expense of personal sacrifice.
Publications from this Thesis


## Contents

Declaration                      i  
Abstract                        ii  
Acknowledgments                 iii 
Publications from this Thesis   iv  

Contents                      v  

**Chapter 1 Introduction**  1  
1.1 Overview of wave modelling  2  
1.2 Overview of wave visualisation  3  
   1.2.1 Introduction to wave visualisation  3  
   1.2.2 Visualisation techniques  6  
      1.2.2.1 Listening  6  
      1.2.2.2 Mechanical analogues  6  
      1.2.2.3 Measurement  7  
      1.2.2.4 Simulation  8  
1.3 Philosophy of visualisation  9  
   1.3.1 Introduction to visualisation  9  
   1.3.2 General procedure  11  
   1.3.3 Previous work by other groups  11  
   1.3.4 Planned work by other groups  13  
1.4 Work covered in this thesis  14  

**Chapter 2 Transmission Line Matrix Model**  15  
2.1 Introduction  15  
2.2 Transmission Line Matrix model  16  
   2.2.1 One-dimensional TLM model  16  

2.2.1.1 Wave equation 16
2.2.1.2 Differential equation for transmission line 20
2.2.1.3 Differential equation for the pressure of sound waves in air 23
2.2.1.4 Standing waves 23
2.2.1.5 Dispersion 24
2.2.1.6 Reflection and transmission 27
2.2.1.7 Iterative form 30

2.2.2 Two-dimensional TLM model 32
2.2.2.1 Two-dimensional transmission line and its equivalent LC circuit 32
2.2.2.2 Iterative form 34

2.2.3 Visualisation 35
2.2.3.1 Examples of visualisation in one-dimension 36
2.2.3.2 Coloured direction of wave propagation in two-dimensions 38

2.2.4 Algorithm and data structure 40

2.3 Conclusion 46
2.3.1 Complexity in one-dimension 46
2.3.1.1 Space complexity 46
2.3.1.2 Time complexity 46
2.3.2 Complexity in two-dimensions 47
2.3.2.1 Space complexity 47
2.3.2.2 Time complexity 47

Chapter 3 Continuous Models 48
3.1 Wave equation 48
3.1.1 Solution for wave equation in one-dimension 49
3.1.2 Solution for wave equation in two-dimensions 49
3.1.3 Solution for wave equation in three-dimensions

3.2 Reflection

3.2.1 Reflected waves in one-dimension

3.2.2 Reflected waves in two-dimensions

3.3 Diffraction

3.4 Interference

3.5 Directional characteristic of sound source

3.6 Conclusion

Chapter 4 Lattice Gas Model

4.1 Introduction

4.2 Lattice Gas model

4.3 Boundary rules

4.3.1 Simulation for a smooth boundary

4.3.1.1 Rules at cells near a smooth boundary

4.3.1.2 Implementation results

4.3.2 Simulation for a rough boundary

4.3.2.1 Rules at cells near a rough boundary

4.3.2.2 Implementation results

4.3.3 Data structure

4.4 Window edge problem

4.4.1 Problem

4.4.2 Solutions

4.4.2.1 Symmetric Duplication method

4.4.2.2 Calculation method with the Continuous model

4.4.2.3 Speed Tapering method

4.4.2.4 Amplitude Tapering method

4.4.2.5 Amplitude Compensation method

4.4.2.6 Post History method
4.5 Noise removal 89
4.6 Conclusion 93

Chapter 5 Algorithms for Calculating the Shape of a Wavefront Scattered by a Curved Obstacle for Specular Reflection 99

5.1 Introduction 99
5.2 Wave pressure 100
5.3 Imaginary source 103
5.4 Wavefront algorithms 105
  5.4.1 a-r algorithm 106
  5.4.2 X-Y algorithm 109
5.5 Simple examples 111
5.6 Complex examples 113
5.7 Comparison of the X-Y and a-r algorithms 121
5.8 Conclusion 123

Chapter 6 Algorithms for Calculating the Geometry Of Multipath Reflection 124

6.1 Introduction 124
6.2 Algorithm of multipath reflection off plane surfaces 124
  6.2.1 Wave path from a transducer directly to a receiver 125
  6.2.2 Wave paths for single-path reflection 126
  6.2.3 Wave paths for multipath reflection 129
6.3 Algorithm for wavefront from multipath reflection off curved surfaces 131
6.4 Conclusion 138

Chapter 7 Application to Robotics 141

7.1 Beam pattern of a Polaroid sensor 141
  7.1.1 Validation of line source visualisation 143
7.2 One transmitter and two receivers 150
Chapter 1
Introduction

As we cannot see sound waves, the ability to visualise waves is useful for understanding wave motion, wave interference, wave diffraction and the scattering of waves off obstacles. In physics laboratories, springs and ripple tanks are often used for visualisation. While these mechanical devices provide an accurate representation of wave physics, they are cumbersome to use, can simulate only a limited range of situations, and require the use of a camera to freeze motion.

In contrast, a well designed computer simulation is easy to use, can simulate a wide range of situations and gives the user fine control over the progress of the simulation. The speed of wave motion can be changed, waves can be stopped or even reversed, and a time sequence of images can be stored for later analysis. To achieve a useful visualisation, we need accurate models of wave physics, and efficient algorithms to compute those models.

Echolocation is the process of locating and recognising objects by analysing the echoes of ultrasonic chirps off them. Echoes may be reflected waves or the superposition of reflected waves and diffracted waves.

The research work reported in this thesis is concerned with finding a way to produce simulation data suitable for visualisation of ultrasonic waves in air for use in understanding ultrasonic sensing (echolocation) for mobile robots. Visualising ultrasonic waves can use data sampled from the real world through measurement devices or simulated through theoretical models. The process for sampling data is simple, but devices are expensive and the number of sample positions is limited by the number of devices. By using computer simulation techniques, the problem in sampling methods can be avoided, but the accuracy of data depends on the modelling results, and the efficiency of simulation depends on the modelling algorithm.

This work began with understanding ultrasonic wave motion, the echolocation principle, and mobile robot navigation theory. To choose a suitable model among a number of simulation models, understanding of the theory behind models is needed, to
decide whether they can be used to represent ultrasonic wave motion. Visualising ultrasonic waves is useful in the robotics field. Researchers in this field use echolocation to navigate and map an environment. Great efforts are made to find echolocation models, and to analyse the wavefront or time sequence signals at receivers, then to detect the objects' locations, orientations, surface textures, and geometric shapes. During this research, visualisation provides an assistive tool.

1.1 Overview of wave modelling

The study of waves is a broad subject. Waves are physical phenomena which include sound waves, water waves, light waves and many others. One of the first properties noted about waves is that they can transport energy and information from one place to another through a medium. The medium itself is not transported, although some streaming can occur if the source is continually generating waves. A change of some physical properties is passed along from point to point as waves propagate. In the case of sound waves the change is in the pressure and density; in the case of radio waves it is in the electric and magnetic fields.

All waves have certain things in common. For example, they can be reflected, refracted and diffracted. They have energy and they transport energy from one point to another. However, waves of different types propagate with widely varying speeds.

Sound waves can be modelled using geometric and physical models. Physical models are based on the wave equation and are the most accurate, but take the longest to compute. Geometric models are based on ray tracing used in computer graphics and are useful for tracing the path of a wave envelope, but give no detail of the wave structure.

One model, discussed in Chapters 2, 5 and 6, based on the wave equation is a continuous model. It is a mathematical description of wave motion and is a function of time, the positions of waves, and other parameters in waves. This model can depict wave motion flexibly, but the complexity of the calculation increases rapidly with the number of
obstacles in an environment and their complexity of their shapes. This model may only be able to simulate wave motion in an environment with a few obstacles of simple geometric shapes.

The Transmission Line Matrix (TLM) model (Pomeroy, 1989) is a discrete model, which is based on the electromagnetic wave theory. It has the advantage of a simple iterative calculation at each simulation step. The complexity of the calculation does not vary with the complexity of a simulated environment. Thus, a simulation using TLM may be able to produce a dynamic display of wave motion in complex environments. The disadvantage of this model is the quantisation error which is introduced because it is a discrete model. As a result, the waveforms on the graphical display are rougher than those produced by a continuous model.

Another discrete model is the Lattice Gas (LG) model. Krutar (Krutar, 1991) implemented this model on parallel computers where it provided a means of rapidly computing and displaying the time evolution of an acoustic field.

It is difficult to achieve dynamic displays on a personal computer with all but the simplest models. An example of a simple model used to produce a dynamic display for robotics research is the arc model (McKerrow, 1990). In this technique, a sound chirp is modelled as an arc with a solid angle equal to the beam angle of the transducer. An example of reflection from specular surfaces using a simulator, based on this model, is given in Figure 1.1. This model has proven useful in understanding the problems that occur when using ultrasonic sensing to guide a mobile robot.

1.2 Overview of wave visualisation

1.2.1 Introduction to wave visualisation

Attempts to use ultrasonic range finding in robotics have achieved varied success. Many researchers feel that robots should use vision because humans use vision. Bats navigate and hunt using echolocation with chirps of ultrasonic energy. The problem facing
humans is that we cannot see sound. Therefore, we have difficulty visualising the effect of an object on an ultrasonic chirp and consequently, struggle to understand the process of echolocation.

![Figure 1.1 Simulator using arc model to show motion of chirp.](image)

1. Reflection off surface B
2. Reflection off surface A, then B, then A
3. Reflection off surface D, then A
4. Reflection off corner between surface B and E
5. Reflection off surface B travels through opening between C and E
6. Reflection off surface C during return of chirp.

This problem is not new. Last century, engineers visualised the sound reaching ships from navigation sirens by drawing detailed graphs (Maw, 1899). To guide a ship through a channel, two sirens were placed on either side (positions B and T in Figure 1.2) and the length of time for sound to travel from one place to the other was measured (8 seconds in this example). At regular intervals (60 seconds), the beacon at T emitted a sound pulse (note). The beacon at B immediately answered with a similar note as soon as it heard the note from T. The captain of a vessel would hear the two notes and measure the period between them. From the graph in Figure 1.2, he could read off his position in the channel.

More recently, Winston Kock developed a simple mechanical apparatus for photographing sound (Kock, 1971). A microphone with a light attached was scanned in front of a sound source. The brilliance of the light was modulated in proportion to the
amplitude of the sound recorded by the microphone mixed with the source signal. The light trace was photographed with a long exposure in a darkened room.

Figure 1.2 Visualisation of the propagation of sound waves from acoustic beacons used to guide ships - reproduced from (Maw, 1899).

The most common means of visualising sound propagation is with a ripple tank, where an observer can watch the motion of waves in an artificial physical environment. The aim of this work is to produce the computer equivalent of a ripple tank by displaying the propagation of sound waves on a graphics display. While the motivation for this is to produce a visualisation system for research, such a system could also be very useful for teaching wave motion. The problem is that educational software must run on a personal computer, placing considerable constraints on the selection of algorithms. Models for visualising wave propagation, reflection, interference and diffraction have been examined including mathematical models, the Transmission Line Matrix model and the Lattice Gas model.
1.2.2 Visualisation techniques

Techniques for visualising sound fall into four groups: listening, mechanical analogues, measurement, and computer simulation. To some extent, we all perceive aspects of our environment by listening to sounds. Physics teachers use springs and ripple tanks to visualise waves. Acoustics engineers scan microphones through sound fields and record the signals for display on a computer screen. Computer scientists develop simulation programs to model waves and display the results with computer graphics.

1.2.2.1 Listening People usually perceive sound through their ears. We have highly developed auditory perception for speech, music and other sounds. Wenzel et al. utilised this ability to develop an acoustic display for a virtual environment workstation (Wenzel, 1990). This system generates localised acoustic cues in real time over headphones. Their aim is to make their virtual world sound real as well as look real.

The frequency range of humans is limited, so we cannot hear the ultrasonic chirps used by bats in echolocation. While humans can discriminate variations as small as 0.5% of a pure tone (within the frequency range we can hear), we can only detect sound pulses that are separated by more than 10 milliseconds (Roos, 1988). This is because the precedence effect of the human auditory system inhibits perception of the second pulse. For this reason, pulsed sonar systems usually present their information visually. Some frequency modulated systems present it aurally as well by converting it into an audible frequency range.

1.2.2.2 Mechanical analogues Physics teachers commonly use mechanical analogues to visualise sounds during laboratory classes. They demonstrate one dimensional wave propagation with springs and wave machines, and two dimensional wave motion with ripple tanks. Some ripple tanks have transparent bases to enable an image of the waves to be projected onto a screen using an overhead projector.

Repeatable demonstrations are difficult to achieve due to the cumbersome nature of ripple tanks. For this reason, teachers refer their students to the photographs of wave
motion in ripple tanks printed in most physics text books (Tipler, 1991). Figure 1.3, a typical example, illustrates the ability of a ripple tank to visualise wave propagation, reflection, interference and diffraction.

![Figure 1.3](image)

**Figure 1.3** Plane waves in a ripple tank meeting a barrier with an opening that is much larger than the wavelength. The barrier has a noticeable effect only near the edge of the opening - reproduced from (Tipler, 1991).

1.2.2.3 **Measurement** While a researcher can observe a ripple tank and record it photographically, accurate measurement of wave propagation is required in order to do calculations. All measurement techniques use microphones to measure the instantaneous sound pressure. This measurement can be recorded and displayed for visualisation purposes. To build up a picture of a sound field, a microphone is scanned as in acoustic holography (Maynard, 85) or an array of microphones is used. Recently, researchers have attempted to measure the velocity amplitude of acoustic waves using a laser doppler photon counting technique (Sharpe, 1987).

Prior to the availability of low cost computers, Winston Kock developed a method for recording an acoustic field on a photographic plate (Kock, 1971). He attached a microphone to the end of a scanning device. It consisted of a long rod oscillated by a motor so as to move one end in an arc transverse to the axis of the field to be measured. A second motor was used to move the scanning device linearly along the axis of the field to be measured. Fixed to the microphone was a lamp whose intensity was modulated by
the measured sound field. A time lapse camera recorded the intensity of the lamp as it moved in the two dimensional scanning plane.

To show the intensity of the sound waves at points in the scanning plane, the measured signal was summed with the reference signal to produce a standing wave pattern. With this apparatus Kock was able to photograph sound including diffraction at the edges of shadowing objects (Figure 1.4).

Figure 1.4 Photograph of plane sound waves arriving from the left which proceed unhindered at the top. In the shadow region below, circular wave fronts are seen, caused by diffraction at the edge of the shadowing object - reproduced from (Kock, 1971).

1.2.2.4 Simulation The problem with real measurements is that many points have to be measured in the acoustic field to gain an accurate picture of wave motion. This process takes a long time and requires expensive equipment. To gain an understanding of wave propagation and scattering, simulations have been developed with computer graphics from this research work.

To develop a useful simulation we have to choose a suitable model. This choice is constrained by the desired accuracy of the simulation, the required update rate of the
graphical display and the method of presenting the information on the display. Complex models and complex rendering can often result in simulations that are too slow for real time visualisation.

1.3 Philosophy of visualisation

1.3.1 Introduction to visualisation

Visualisation is a method which is used to extract knowledge from data. Visualisation utilises colour, intensity, transparency, texture and a myriad of other techniques to form images, which can convey a tremendous amount of information in a short period of time. Visualisation offers a method of seeing the unseen. For example, temperature distribution in a 3D volume is invisible to our eyes, but a colour coded volume rendering with blues for low temperatures and reds for high temperatures can convey a great deal of information to the combustion scientist. Visualisation transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations.

Visualisation is a form of communication that transcends application and technological boundaries. It is a tool for discovery and understanding, and a tool for communication and teaching. Scientists use it to create graphics to verify the integrity of their simulation, to gain insights from their analysis, and to communicate their findings to others. In many fields visualisation is already revolutionising the way that scientists do their jobs.

The idea of visualisation is not new. People have used graphs, and diagrams to represent mathematical equations for centuries. More recently, large amounts of numeric data from simulations and experiments have been displayed by computers in various forms for human interpretation. Visualisation is becoming an intimate part of many areas of research. Techniques for visualisation, however, are still being developed and visualisation research is just beginning to be recognised as a cornerstone of future computational science. As scientists bring increasingly complex problems to computation
science, visualisation will become an even more essential tool for extracting science out of numbers.

Scientific visualisation is a new and emerging area that is having an impact on how computers are used in research. The foundation material for the techniques of scientific visualisation is derived from many areas including computer graphics, image processing, computer aided design, signal processing, numerical analysis, and so forth. Scientific visualisation is concerned with techniques that allow scientists and engineers to extract knowledge from the results of simulations and computations. Advances in scientific computation are allowing mathematical models and simulations to become increasingly complex and detailed.

The graphs required by scientists and engineers of today are more complex than a mere curve in an X-Y Cartesian coordinate system. The relationships of interest often involve multidimensional quantities in both domain and range. "Graphing" these relationships presents quite a challenge. The same basic principals of graphics and geometry used in the creation of simpler graphs can be applied to yield useful scientific visualisation techniques.

Computational science and engineering now relies heavily on scientific visualisation to represent its solutions, enabling scientists to turn mountains of numbers into movies and to graphically display measurements of physical variables in space and time.

Graphical visualisation techniques are useful tools for scientists. They can be used to analyse such real-valued functions as temperature, pressure, rainfall, and ozone depletion over the earth's surface. Effective visual representation of 3D data sets has produced new understanding. Using visualisation, doctors plan surgical procedures, geophysicists study our changing planet, and molecular modellers produce new drugs.

Computer visualisation is playing an increasingly significant role in many fields, and has gained rapid and widespread acceptance. With its basis in computer graphics and its interdisciplinary nature, it affects both science and how society performs its daily tasks.
1.3.2 General procedure

Visualisation of scientific data includes three processes: analysis, display, and interaction. This data is produced by simulation or measurement.

Analysis is the exploration of the data using mathematical techniques to obtain the information required to answer a specific question or to illustrate a specific phenomenon. It can be applied to multivariate data from general purpose techniques, such as contour surface extraction. It can use highly specific techniques such as shock surface extraction and vortex core extraction in the study of fluid flow data sets.

Display is the presentation of the data in a visual form. The display technique must match the capabilities of the human visual system. Tailoring is used and includes the use of colours, surfaces instead of isolated curves and orientation cues. Display techniques include the lighting and shading of surfaces, depth cues such as stereopsis and animation.

Interaction is the process of the user changing the visualisation. It can include simple refinement of the colour coding of contour levels and complex manipulation of virtual probes that simulate physical behaviour or measurements.

1.3.3 Previous work by other groups

Chernoff (1973) suggested the use of cartoon faces with variable attributes (such as length of the nose, or curve of the mouth) to graphically represent higher dimensional data. These examples provide evidence that these faces can be used for clustering, discrimination, and time-series analysis. In 1976, J. R. Eagleman published a book, titled "The Visualisation of Climate" (Eagleman, 1976). At that time, visualisation was very simple, only using strip charts, and schematic diagrams.

In the early 1980s, some scientific disciplines, such as molecular modelling and medical imaging, battled to obtain funding to perform experiments in visualisation. They wanted to demonstrate the benefits of applying visualisation techniques to the analysis of experimental and simulated data. In many scientific fields raster colour plots were unusual, and high-resolution animations non-existent. Business and commercial applications of graphics largely focused on the entertainment industry.
By the mid- to late 1980s, display hardware had developed rapidly, and prior restrictions in both dimension and number of bits per pixel began to disappear. Sensors used for collecting scientific measurements generated data at increasing rates. Researchers had to meet the challenge of developing new methods of data presentation to gain insight into these volumes of data. The journal *Computer* contains much of that information combined with a good introduction to scientific visualisation. In the late 1980s, scientific visualisation began to flourish.

The majority of visualisation applications is computational fluid dynamics (CFD). It is an important scientific topic because of its widespread application, ranging from oceanic, atmospheric, and astronomical studies to analysing the change in the properties of metals due to temperature change. CFD has emerged as a key visualisation area because computer power now permits the large simulations required to model complex processes. Many scientists are performing research in this area (Helman, 1991).

A key area of scientific visualisation is *volume visualisation*, which projects a multi-dimensional data set onto a 2D image plane. Volume visualisation went from its initial applications in medical imaging to use across science. It portrayed clouds, water, molecules, and other phenomena from both empirical and simulated scientific data. Nelson and Elvins have developed a clinical application with volume visualisation methods to assist physicians with their patient diagnosis and treatment (Nelson, 1993).

Grotch (1985) used a 3D approach to analyse the "greenhouse effect" on the atmosphere. His interest in this problem stemmed from his research on constructing interpolating functions defined over the entire sphere based on discrete samples at arbitrary locations.

Max *et al.* reported (Max, 1993) on techniques they developed to visualise climate variables, particularly clouds and wind velocities: volume rendering, textured contour surfaces, and vector field rendering. Schroder reported (Schroder, 1993) visualising clouds with barycentric interpolation and fractals.

Visualising 3D fluid-flow fields remains one of the most challenging applications in scientific visualisation. Wijk studied the representations for 3D vector fields and
introduced the concept of surface particles (Wijk, 1993). The use of surface particles is a flexible and versatile technique for flow visualisation.

To help scientists visualise 3D tensor data, Delmarcelle and Hesselink (Delmarcelle, 1993) have developed a methodology based on the concept of a hyperstreamlines.

MacLeod et al. at the University of Utah have developed a medical visualisation system to assist in the study of the heart’s bioelectric activity (MacLeod et al., 1993). This system can do rendering in real time, equipped with powerful hardware. In their system, the potentials are recorded from 192 electrodes and interpolated using a Laplacian smoothing algorithm.

Muraki introduced a method of obtaining a unique and hierarchical volumetric shape description of 3D data. He applied 3D orthogonal wavelet transforms to real volume data generated from a set of magnetic resonance images, and showed the method’s efficiency in describing 3D objects (Muraki, 1993).

Visualisation in scientific computing falls into the following categories:

• Molecular modelling;
• Medical imaging;
• Brain structure and function;
• Mathematics;
• Geosciences (meteorology);
• Space exploration;
• Astrophysics;
• Computational fluid dynamics; and
• Finite element analysis.

### 1.3.4 Planned work by other groups

Several topics continue to grow in importance. These include research into new methods for examining high-dimensional data and into the use of new interface technology such as virtual environments and multi-media. Other emerging areas include the synthesis of graphics and imaging, and better integration of physics and visualisation. Perhaps the
most important need today in the scientific arena is for truly useable but still powerful and extensible tool sets. In the way that many people now use word processors, it is expected that we will see scientists routinely using visualisation tools on their desks.

Visualisation can produce scientific knowledge to benefit humanity. The integration of visualisation technology into a variety of new, emerging, information age technologies will change how we work and play.

1.4 Work covered in this thesis

This thesis looks at models for simulating wave motion and techniques for visualising the data they produce. Each model was coded in either Modula 2 or C++ to produce the visualisation in the diagrams in following chapters. It was chosen to code the models rather than using any of the shelf visualisation tools because none of the available tools applied directly to the application and comparison between algorithms was desired.

In Chapters 2, 3, 4, three approaches to modelling waves are discussed. In Chapters 5 and 6, models are developed for complex situations. In Chapter 7, the application of these visualisation tools to ultrasonic sensing in robotics is studied.
Chapter 2
Transmission Line Matrix Model

2.1 Introduction

The Transmission Line Matrix (TLM) model was developed by P. B. Johns and his co-workers in the early 1970s (Johns and Beurle, 1971). This model is a computer simulation of electromagnetic waves which can be used to visualise diffusion processes, network analysis and other field problems.

The conceptual framework of TLM consists of lumped circuit components. These components are represented by transmission-line segments. A mesh of these elements is used to form a discrete model of the field in space and time.

The advantages of this model are computational simplicity and inherent parallelism. As the model is based on circuit components which are easy to understand, the nature and significance of errors can be intuitively analysed.

The TLM model is used in simulations of:

• electromagnetic waves (So and Hoefer, 1991);
• underwater acoustics and acoustic behaviour of sonar transducers (Coates et al., 1990);
• the propagation of sound in hot flowing gas (Davies, 1991);
• water diffusion in white rice (Hendrickx et al., 1986);
• acoustic fields (Pomeroy et al., 1989; Saleh, 1991);
• the complex interactions between sound waves and objects (Pomeroy et al., 1991);
• the phenomena of heat diffusion (Webb, 1991; de Cogan and Enders, 1991);
• the temperature field during cyclic glass pressing processes (Pulko and Phizackerley, 1991);
• electromagnetic scattering and radiation problems (Simons, 1991);
• electromagnetic field problems (Simons, 1991); and
2.2 Transmission Line Matrix model

TLM modelling is a numerical method for solving scattering problems. This method produces a computer simulation of electric fields in both space and time. The model operates by propagating sine waves along transmission lines. Due to the simulated discontinuity at the cells, transmitted and reflected waves are scattered back into the lines. These scattered waves then become incident on adjoining cells at the next time instant.

2.2.1 One-dimensional TLM model

TLM can be applied to one-, two- and three-dimensional space. The theoretical foundations of the TLM model for two-dimensions is given in (Johns and Beurle, 1971) and for three-dimensions in (Akhtarsad and Johns, 1975). In this section, the theoretical foundations of the TLM for the one-dimensional model are reviewed, and the iterative form for simulating wave propagation and reflection is derived.

2.2.1.1 Wave equation The linear wave equation \( f(z,t) \) for wave propagation within a medium is

\[
\frac{\delta^2 f}{\delta z^2} = \frac{1}{V_\phi} \frac{\delta^2 f}{\delta t^2}
\]  

where

- \( V_\phi \) is the velocity of wave travelling along the \( z \) axis,
- \( f \) is the physical property that varies as the wave travels through the medium, and
- \( t \) is the time.
This equation gives a complete description of wave motion and can be applied in general to various types of waves moving through a non-dispersive medium. For electromagnetic waves, $f$ corresponds to electric or magnetic field components. For sound waves, $f$ corresponds to variations in the pressure $P$ of a gas,

$$\frac{\delta^2 P}{\delta z^2} = \frac{1}{c^2} \frac{\delta^2 P}{\delta t^2}$$

where $c$ is the speed of sound.

Figure 2.1 is produced by a one-dimensional simulator with the TLM model, and shows wave propagation at different moments in time. The vertical axis represents the wave pressure and the horizontal axis the location. In the simulator, wave motion at different times is positioned adjacently, one higher than the other. This enables viewers to obtain the concept of wave movement by comparison.

Figure 2.1 Waves move forward in the direction of the velocity. The wavelength is 200 cells.
In Figure 2.2, the wavelength has been altered.

Sound waves are longitudinal waves (i.e., vibrations are along the wave propagation direction). They can travel through any material medium with a speed that depends on the properties of the medium. As sound waves travel through a medium, the particles in the medium vibrate to produce density and pressure changes along the direction of the motion of the wave. The speed of sound \( c \) in a medium of compressibility \( B \) and density \( \rho \) is given by

\[
c = \sqrt{\frac{B}{\rho}}
\]  

If \( s(z,t) \) is the displacement of a small volume element measured from its equilibrium position with respect to time, we can express this harmonic displacement function as

\[
s(z,t) = S_m \cos(kz - \omega t)
\]
where

\[ S_m \] is the displacement amplitude,
\[ k \] is the angular wavenumber (measured in radians per meter), and
\[ \omega \] is the angular frequency (measured in radians per second).

If \( \lambda \) is the wavelength (measured in meters per cycle), \( T \) is the period (measured in seconds per cycle), then \( k \) and \( \omega \) can be written as

\[
k = \frac{2\pi}{\lambda} \quad 2.5
\]
\[
\omega = \frac{2\pi}{T} \quad 2.6
\]

The relationship of \( c \) with the \( k \) and \( \omega \) is

\[
c = \frac{\lambda}{T} = \frac{\omega}{k} \quad 2.7
\]

A sound wave can be considered as a pressure wave. The pressure function \( P(z,t) \) is:

\[
P(z,t) = P_m \sin(kz - \omega t), \quad 2.8
\]

where \( P_m \) is the amplitude of the pressure (Serway, 1986, p. 370), and it is equal to:

\[
P_m = \rho c \omega S_m. \quad 2.9
\]

The intensity \( I \) of a sound wave, which is the power per unit area, is given by (Serway, 1986, p. 371)

\[
I = \frac{\text{power}}{\text{area}} = \frac{1}{2} \rho (\omega S_m)^2 c^2. \quad 2.10
\]
2.2.1.2 Differential equation for transmission line The one-dimensional transmission line is shown in Figure 2.3.

![Figure 2.3](image)

**Figure 2.3** Transmission line driven at $z = 0$ and extending to infinity.

The equivalent circuit for the one-dimensional transmission line is:

![Figure 2.4](image)

**Figure 2.4** Equivalent LC circuit of a lossless transmission line.

where $L$ is the inductance and $C$ is the capacitance per unit length, and the subscript $n$ indicates the discrete location.

According to circuit theory, we have the following discrete equation:

$$V_{n+1}(t) - V_n(t) = -L\Delta z \frac{di_n(t)}{dt}$$  \hspace{1cm} (2.11)
We can rewrite these equations as continuous functions of the points $n-1$, $n$ and $n+1$:

\[ V_n(t) \equiv v(z,t) \]  
\[ V_{n+1}(t) \equiv v(z + \Delta z,t) \]  
\[ I_n(t) \equiv i(z,t) \]  
\[ I_{n-1}(t) \equiv i(z - \Delta z,t) \]

Replacing $V$ and $I$ in Equations 2.11 and 2.12 with $v$ and $i$, and dividing them by $\Delta z$, and then letting $\Delta z \to 0$, we get:

\[
\lim_{\Delta z \to 0} \frac{v(z + \Delta z,t) - v(z,t)}{\Delta z} = -L \frac{di(z,t)}{dt}
\]

and

\[
\lim_{\Delta z \to 0} \frac{i(z,t) - i(z - \Delta z,t)}{\Delta z} = -C \frac{dv(z,t)}{dt}
\]

Equations 2.17 and 2.18 lead to the partial differential equations:

\[
\frac{\partial v(z,t)}{\partial z} = -L \frac{\partial i(z,t)}{\partial t}
\]
\[ \frac{\partial i(z,t)}{\partial z} = -c \frac{\partial v(z,t)}{\partial t}. \]  

Furthermore,

\[ \frac{\partial^2 v(z,t)}{\partial z^2} = LC \frac{\partial^2 v(z,t)}{\partial t^2}. \]

Comparing the wave equation (Equation 2.1) with Equation 2.21, the velocity \( V_\phi \) for waves of voltage on a transmission line is given by:

\[ V_\phi = \frac{1}{\sqrt{LC}}. \]

The shunt capacitance per unit length and series inductance per unit length for two infinitely long, straight, parallel wires are given by (Crawford, 1968)

\[ C = \frac{1}{4 \ln \left( \frac{D+r}{r} \right)}, \]

\[ L = \frac{4}{c_i^2} \ln \left( \frac{D+r}{r} \right), \]

where

\( c_i \) is the speed of light,

\( r \) is the radius of each wire, and

\( D \) is the distance between the wires.

Substituting Equation 2.23 into Equation 2.22, we have:

\[ V_\phi = \frac{1}{\sqrt{LC}} = c_i. \]
Thus the propagation velocity for travelling waves of voltage on a transmission line consisting of two straight parallel wires is $c_1$.

### 2.2.1.3 Differential equation for the pressure of sound waves in air

Let $P(z,t)$ stand for a pressure of air, and $u(z,t)$ stand for particle speed. The sound wave equation is given by:

$$\frac{\partial P(z,t)}{\partial z} = -\rho \frac{\partial u(z,t)}{\partial t} \quad 2.25$$

$$\frac{\partial u(z,t)}{\partial z} = -\frac{1}{B} \frac{\partial P(z,t)}{\partial t} \quad 2.26$$

$$\frac{\partial^2 P(z,t)}{\partial z^2} = \frac{\rho}{B} \frac{\partial^2 P(z,t)}{\partial t^2} \quad 2.27$$

This is one form of the differential sound wave equation. The sound wave is obviously the mechanical analog of the electric wave in the lossless transmission line. The analogous relationships among the electrical and mechanical parameters are as follows:

<table>
<thead>
<tr>
<th>Electrical Parameters</th>
<th>Mechanical Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage $V$</td>
<td>Pressure $P$</td>
</tr>
<tr>
<td>Current $i$</td>
<td>Particle speed $u$</td>
</tr>
<tr>
<td>Inductance $L$</td>
<td>Density $\rho$</td>
</tr>
<tr>
<td>Capacitance $C$</td>
<td>Inverse bulk modulus</td>
</tr>
<tr>
<td></td>
<td>of elasticity $B^{-1}$</td>
</tr>
<tr>
<td>Impedance $\sqrt{\frac{L}{C}}$</td>
<td>Impedance $\sqrt{B\rho}$</td>
</tr>
</tbody>
</table>

**Table 2.1** The analogous relationships between parameters of the electrical and of acoustic waves.
2.2.1.4 Standing waves  Suppose the source wave has the form \( f(t) \) (where \( f(t) = 0 \) when \( t < 0 \)), located at \( z = 0 \). A travelling wave will have the form \( f(t - \frac{z}{V_p}) \).

The travelling wave will be reflected when it encounters a boundary. The reflected wave will have the form \( Rf(t - \frac{2l - z}{V_p}) \), where \( R \) is the reflection coefficient which depends on the impedance change at the boundary and \( l \) is the location of the boundary. When \( t = \frac{2l}{V_p} \), the wave has travelled back to \( z = 0 \). The superposition of the waves travelling forward and backward form a standing wave in the system. They will have the form:

\[
f(t - \frac{z}{V_p}) + Rf(t - \frac{2l - z}{V_p}).
\]

2.2.1.5 Dispersion  The term dispersion means that waves at different frequencies travel at different speeds in same media which causes the wave forms to change shape as they propagate (Simons and Sebak, 1991). In TLM model, the ratio of mesh length between nodes to wavelength \( \frac{\Delta z}{\lambda} \) varies the ratio of propagation velocity in the equivalent circuit to that in transmission line \( \frac{V_p}{c} \). Simons and Sebak (1991); Simon, Sebak and Bridges (1991); Hoefer (1989); and Johns and Beurle (1971); provide more details about two-dimensional dispersion analysis. In this section dispersion for one-dimension is discussed.

In the equivalent circuit of the transmission line, we have:

\[
C\Delta z \frac{dV_n(t)}{dt} = I_{n-1}(t) - I_n(t)
\]  

(2.12)

\[
C\Delta z \frac{d^2V_n(t)}{dt^2} = \frac{dI_{n-1}(t)}{dt} - \frac{dI_n(t)}{dt}
\]

\[
= -\frac{1}{LA\Delta z} (V_n(t) - V_{n-1}(t)) + \frac{1}{LA\Delta z} (V_{n+1}(t) - V_n(t)) \quad \text{(from Equation 2.11)}
\]
\[
V_{n}(t) = \frac{1}{L\Delta z} (V_{n-1}(t) - 2V_{n}(t) + V_{n+1}(t)).
\]

Supposing the driving voltage is a sinusoidal wave, and

\[
V_{n}(t) = A\sin[\omega(t - \frac{n\Delta z}{V_{\phi}})]
\]

we have

\[
\frac{d^2V_{n}(t)}{dt^2} = -\omega^2V_{n}(t)
\]

\[
V_{n-1}(t) + V_{n+1}(t) = A\sin[\omega(t - \frac{(n-1)\Delta z}{V_{\phi}})] + A\sin[\omega(t - \frac{(n+1)\Delta z}{V_{\phi}})]
\]

\[= 2A\sin[\omega(t - \frac{n\Delta z}{V_{\phi}})]\cos[\frac{\omega \Delta z}{V_{\phi}}]
\]

\[= 2V_{n}(t)\cos[\frac{\omega \Delta z}{V_{\phi}}]
\]

From Equations 2.32, 2.30 and 2.33,

\[-\omega^2V_{n}(t) = \frac{1}{LC\Delta^2 z} [V_{n-1}(t) - 2V_{n}(t) + V_{n+1}(t)]
\]

\[= \frac{1}{LC\Delta^2 z} [2V_{n}(t)\cos(\frac{\omega \Delta z}{V_{\phi}}) - 2V_{n}(t)]
\]

\[= \frac{2}{LC\Delta^2 z} V_{n}(t)[\cos(\frac{\omega \Delta z}{V_{\phi}}) - 1]
\]

\[= - \frac{4}{LC\Delta^2 z} V_{n}(t)\sin^2(\frac{\omega \Delta z}{2V_{\phi}})
\]

It can be further simplified,

\[
\omega = \frac{1}{\sqrt{LC}} \left| \sin \left( \frac{\omega \Delta z}{2V_{\phi}} \right) \right|
\]

2.34
As \( \omega = V_\phi k \), from Equation 2.35, we have

\[
V_\phi = \frac{1}{\sqrt{LC}} \left| \sin\left(\frac{k\Delta z}{2}\right) \right|
\]

Replacing \( k \) with \( \frac{2\pi}{\lambda} \) and \( \frac{1}{\sqrt{LC}} \) with \( c_l \) in Equation 2.36, then,

\[
V_\phi = c_l \frac{\sin\left(\frac{\pi}{\lambda} \Delta z\right)}{\frac{\pi}{\lambda} \Delta z}
\]

We can find that the velocity \( V_\phi \) varies with mesh size \( \Delta z \) and the velocity \( V_\phi \) will become zero when \( \Delta z \) is at certain values. These values are referred to cutoff points. The first cutoff point occurs at

\[
\Delta z = \lambda
\]

Consider Equation 2.37, if we let \( \Delta z \) approach zero, it leads to

\[
\lim_{\Delta z \to 0} V_\phi = c_l
\]

That is, the velocity in the equivalent circuit approaches that in the transmission line.

In terms of dispersion, Equation 2.37 usually is in a form of

\[
\frac{V_\phi}{c_l} = \frac{\sin\left(\frac{\pi}{\lambda} \Delta z\right)}{\frac{\pi}{\lambda} \Delta z}
\]
Figure 2.5 is a plot of Equation 2.40, the one-dimensional dispersion relation. The plot of the two-dimensional dispersion relation looks similar except that the maximum value of \( \frac{V_0}{c_i} \) is \( \frac{1}{\sqrt{2}} \).

![Figure 2.5 Dispersion of the velocity of waves in a one-dimensional TLM network.](image)

### 2.2.1.6 Reflection and transmission

Suppose we have a semi-infinite tube with sound characteristic impedance \( Z_1 \) extending from \( z = -\infty \) to \( z = 0 \). At \( z = 0 \) an end having an impedance \( Z_2 \) is attached.

![Figure 2.6 a) Semi-infinite tube with sound characteristic impedance \( Z_1 \). An end having an impedance \( Z_2 \) is attached.](image)
The reflected coefficient for sound pressure is given by

$$R_{12} = \frac{Z_2 - Z_1}{Z_2 + Z_1}$$  \hspace{1cm} 2.41

and transmission coefficient is given by

$$T_{12} = 1 + R_{12}$$
$$= \frac{2Z_2}{Z_2 + Z_1}$$  \hspace{1cm} 2.42

At $z = -\infty$ there is a transmitter emitting travelling waves in +z direction, thus there is an incident travelling wave given by

$$P_{inc}(z,t) = A\cos(\omega t - kz),$$  \hspace{1cm} 2.43

where $P_{inc}(z,t)$ stands for incident pressure at position $z$ and time $t$.

Let $P_{ref}(z,t)$ stand for the reflected pressure at position $z$ and time $t$. Then suppose the travelling wave reaches at position $z = 0$ at time $t_0$, thus:

$$P_{ref}(0,t_0) = R_{12}P_{inc}(0,t_0) = R_{12}A\cos(\omega t_0).$$  \hspace{1cm} 2.44
If the medium is uniform, that is, the equivalent inductance $L$ and the equivalent capacitance $C$ remain constant, the wave travels with a constant velocity $V_\varphi = \frac{1}{\sqrt{LC}}$.

The motion of the moving element at position $z$ and time $t$ is the same as that of the moving element at $z = 0$ and the earlier time $t'$, where $t'$ is earlier than $t$ by the time that the wave takes to travel the distance $z$ at the velocity $V_\varphi$:

$$t' = t - \frac{z}{V_\varphi}. \quad 2.45$$

The reflected wave travels in $-z$ direction, thus we have the form of travelling wave,

$$P(z, t) = P(0, t') \quad 2.46$$

$$P_{\text{ref}}(z, t) = P_{\text{ref}}(0, t')$$

$$= R_{12} P_{\text{inc}}(0, t')$$

$$= R_{12} P_{\text{inc}}(0, t - \frac{z}{V_\varphi})$$
\[ R_{12} = R_{12}A \cos[\omega(t + \frac{z}{V_p})] \]
\[ = R_{12}A \cos(\omega t + kz) \]

In general, when the properties of the medium change at position \( z \), we have

\[ P_{\text{ref}}(z, t) = R_{12} P_{\text{inc}}(z, t) \]
\[ P_{\text{tra}}(z, t) = T_{12} P_{\text{inc}}(z, t). \]

where \( P_{\text{tra}}(z, t) \) is the transmitted pressure which travels through the second medium.

### 2.2.1.7 Iterative form

These equations for reflection and transmission of sound pressure in a tube, Equations 2.47, 2.48 2.49, are analogous to electric voltage on a transmission line.

![Diagram](image)

**Figure 2.7** The incident voltage \( V(t_\cdot) \) is divided into two parts, the travelling part \( V_{\text{tra}}(t_\cdot) \) and the reflected part \( V_{\text{ref}}(t_\cdot) \) when passed through a cell.

\[ V_n(t_+) = V_{n, \text{tra}}(t_+) + V_{n, \text{ref}}(t_+) \]
\[ = R_{n+1} V_n(t_-) + T_{n+1} V_{n, \text{tra}}(t_-) \]
\[ V_n(t_-) = V_{n, \text{tra}}(t_-) + V_{n, \text{ref}}(t_-) \]
\[ R_{n+1} = \frac{Z_{n+1} - Z_n}{Z_{n+1} + Z_n} \]  \hspace{2cm} 2.52

\[ T_{n+1} = 1 + R_{n+1} \]  \hspace{2cm} 2.53

\[ R_{n+1} = \frac{Z_n - Z_{n+1}}{Z_n + Z_{n+1}} = -R_{n+1} \]  \hspace{2cm} 2.54

\[ T_{n+1} = 1 + R_{n+1} = 1 - R_{n+1} \]  \hspace{2cm} 2.55

where \( V \) or \( V \) denotes the wave of voltage goes in \(-z\) direction or \(+z\) direction, and

\[ t_+ = \lim_{\Delta t \to 0} t + \Delta t \]

\[ t_- = \lim_{\Delta t \to 0} t - \Delta t \]

Taking the voltage at the source, which is a function of time \( t \), \( f(t) \), and boundary conditions into account, we have:

\[ V_{i-}(0_+) = 0, \ (n = 0, 1, 2, ..., m - 1) \]  \hspace{2cm} 2.56

\[ V_{i-}(0_-) = 0, \ (n = 0, 1, 2, ..., m - 1) \]  \hspace{2cm} 2.57

\[ V_{i-}(t_+) = f(t) \]  \hspace{2cm} 2.58

\[ V_{i-}(t_+) = T_{n+1} V_{i-}(t_-) + R_{n+1} V_{i-}(t_-) \ (n = 0, 1, 2, ..., 1 - 1, t > 0) \]  \hspace{2cm} 2.59

\[ V_{i-}(t_+) = T_{n+1} V_{i-}(t_-) + R_{n+1} V_{i-}(t_-) \ (n = 0, 1, 2, ..., 1 - 1, t > 0) \]  \hspace{2cm} 2.60
\[V_{n}(t_{-} + 1) = V_{n-1}(t_{+}) \quad (n = 0, 1, 2, ..., 1 - 1) \]  
2.61

\[V_{n}(t_{-} + 1) = V_{n+1}(t_{+}) \quad (n = 0, 1, 2, ..., 1 - 2) \]  
2.62

\[V_{L-1}(t_{-} + 1) = 0 \]  
2.63

Equations 2.56 - 2.63 are the iteration equations for simulating wave motion in one-dimensional space with the TLM model.

2.2.2 Two-dimensional TLM model

The two-dimensional TLM model introduced by Johns and Beurle is based on a rectangular shaped lattice. The dispersion relation for propagation along the main mesh axes (Johns and Beurle, 1971) is

\[\sin(\beta_{n} \Delta l / 2) = \sqrt{2} \sin(\omega \Delta l / (2c_{0}))\],

where \(\beta_{n}\) is the propagation constant in the matrix network. It appears that the first cutoff point occurs for \(\Delta l = 0.25\lambda\).

2.2.2.1 Two-dimensional transmission line and its equivalent LC circuit

The basic building block of a two-dimensional TLM network is a shunt cell (Figure 2.8 (a)). Such a configuration can be approximated by the lumped-element model shown in Figure 2.8 (b). There is a direct equivalence between the voltages and currents on the lines in the mesh and the pressure and intensity of sound in air. With this mesh we can model two-dimensional wave problems.
The model operates by propagating sine waves along the lines and, due to the simulated discontinuity at the cells, results in transmitted and reflected waves being scattered back into the lines. These scattered waves then become incident on adjoining cells at the next time instant. Each iterative step includes two processes, scattering and connection (Figure 2.9). The scattering process is that waves scatter from a cell after impulses incident on the cell; the connection process is that waves propagate toward its neighbours after waves scatter from a cell. The computation for the scattering process at each cell, within each iteration, is the weighted sum of impulses incident on the cell.
2.2.2.2 Iterative form The weighted sum of impulses incident on a cell provides a solution for the acoustic pressure at that point in space and time. The computation at each cell for the scattering process, within each iteration, is defined by the following equation

\[ V_{n+1}^{r} = \frac{1}{2} \left( \sum_{m=1}^{4} V_{m}^{i} \right) - k V_{n}^{r}, \quad (n = 1, 2, 3, 4) \]  

2.64

and for the connecting process is

\[ V_{k+1}^{i}(z, x) = k+1 V_{z}^{r}(z, x - 1) \]  

2.65

\[ k+1 V_{3}^{i}(z, x) = k+1 V_{r}^{r}(z, x + 1) \]  

2.66

\[ k+1 V_{2}^{i}(z, x) = k+1 V_{z}^{r}(z - 1, x) \]  

2.67

\[ k+1 V_{4}^{i}(z, x) = k+1 V_{z}^{r}(z + 1, x) \]  

2.68

where

the superscript \( i \) of \( k V_{m}^{i} \) denotes the incident impulse,
the superscript \( r \) of \( k V_{n}^{r} \) denotes the reflected impulse,
the subscript \( k \) denotes the time step, and
the subscript \( n \) denotes line.
2.2.3 Visualisation

The mathematics of the TLM model is simple and hence rapid to compute at each iterative step. With this model, values are calculated at cells in the mesh. To render a continuous or a smooth image, we must have sufficient cells per wavelength to enable accurate interpolation of the values between cells. If we use linear interpolation, such as straight lines connecting the cell values, we can get results quickly, but the image looks rough (Figure 2.10). If we use nonlinear interpolation, such as spline techniques, we can get better results, but at the cost of increased calculation time.

Figure 2.10 Wave propagation and reflection visualised with the TLM model. Line rendering with linear interpolation between cells.

To smooth the image and to avoid the long time required for nonlinear interpolation, colour hue was used to render the sound pressure at all points in 2D space. Our eyes are much more sensitive to the straightness of a line than to the intensity of a colour. This can be seen by comparing Figure 2.11 where the waves are visualised using hue to Figure 2.10 where the waves are visualised using line rendering.
Figure 2.11 Time sequence of wave propagation and reflection off a flat surface visualised with the TLM model. Grey hue rendering with linear mapping from pressure to grey scale.

2.2.3.1 Examples of visualisation in one-dimension A visualisation system with a graphical user interface has been developed. Several examples in one-dimension are provided here. They include the amplitude sonagraph, bar sonagraph, and line rendering.
Figure 2.12 Amplitude sonagraph. Simulation of wave propagation in one dimensional space.

Figure 2.13 Bar sonagraph. Simulation of wave propagation in one dimensional space.

Figure 2.14 Waves with 1., 0.5, 0.5, 0.5 peak to peak amplitude of first, third, fifth and seventh - frequency harmonic separately.
2.2.3.2 Coloured direction of wave propagation in two-dimensions The values at each cell in a two-dimensional model are stored with four directions. They are rendered with four different colours to represent different directions of the wave propagation. In Figure 2.16, red is used to represent particle movement towards the right forward, green represents towards the left, yellow up, and blue down.

Figure 2.16 shows the modelling of a short line source with many source cells along a line. Also visible is the amplitude pattern due to beam forming with fixed amplitude plane waves and falling amplitude circular waves at the edges. However, the noise due to discretisation is very noticeable.
Figure 2.16 Time sequence of wave propagation and reflection off a flat surface visualised with the TLM model. Red colour represents the direction towards the right, green towards the left, yellow up and blue down.
2.2.4 Algorithm and data structure

In this section, the algorithms and data structures used in the TLM model are presented. The code for these algorithms is included in Appendix E.

To simulate a harmonic wave, the final signals are a combination of all of the different frequency harmonics. Figures 2.14 and 2.15 show seven harmonic components. The Algorithm 2.1 is used to simulate a transmitter emitting harmonic waves. The type of the variable ‘incidence’ in the algorithm is a two-dimensional array, defined as:

\[
\text{type } \text{Array1d} = \text{array}[1..n, 1..2] \text{ of real;}
\]

which stores the instantaneous amplitude of wave pressure at all simulation cells in two directions, left and right. The source is set on the ‘source_position’, emitting toward the right direction. ‘source_position’ is the index number of the source position in the array ‘incidence’. ‘harmonAmplitude’ is a one-dimensional array, which contains an envelope value of each frequency harmonic.

```
Procedure Source_emit(Var incidence : Array1d);
begin
    incidence[source_position, right] := 0.;
    for i := 1 to Num_harmonic do
        incidence[source_position, right] := incidence[source_position, right] - harmonAmplitude[i] * sin((\omega + \pi) * i);
end Source_emit;
```

Algorithm 2.1 Simulation of a harmonic wave source.

Algorithm 2.2 is to simulate the connection process during wave propagation; that is, the incident pressure at a cell is the reflected pressure from its neighbour cells. The
reflected value from the boundary is assumed to be zero. The array ‘incidence’ contains incident pressures at cells and the variable ‘reflect’ contains reflected pressures.

**Procedure** Connection\_1d(reflect: Array1d; Var incidence: Array1d);

begin

for i := 2 to n - 1 do  
{where n is the width of simulation space}

incidence[i, left] := reflect[i-1, right];

incidence[i, right] := reflect[i+1, left];

end;

incidence[n, left] := reflect[n - 1, right];

incidence[1, right] := reflect[2, left];

end Connection\_1d;

**Algorithm 2.2** Simulation of connection process with TLM model in one-dimension.

Algorithm 2.3 is to simulate the scattering process during wave propagation, that is, the scattering pressure at one cell is the sum of the reflected components from the cell itself and the transmitted components from its neighbours. The array ‘impedance’ contains the impedance value at each cell. ‘R12’ is the reflection coefficient, and ‘T12’ and ‘T21’ are the transmission coefficients in a certain direction. The values of the scattered pressures are stored in the array ‘reflect’.

**Procedure** Scattering\_1d(incidence: Array1d; Var reflect: Array1d);

begin

for i := 1 to n do

R12 := (impedance[i] - impedance[i-1]) / (impedance[i] + impedance[i-1]);

T12 := 2 * impedance[i] / (impedance[i] + impedance[i-1]);

T21 := 2 * impedance[i-1] / (impedance[i-1] + impedance[i]);

reflect[i, left] := R12 * incidence[i, left] + T21 * incidence[i, right];

reflect[i, right] := T12 * incidence[i, left] - R12 * incidence[i, right];

end;

end Scattering\_1d;

**Algorithm 2.3** Simulation of scattering process with TLM model in one-dimension.
Algorithm 2.4 is the main procedure of the TLM simulation model in one-dimension. It includes three simulation steps: simulation for source emission (Source_emit), pressure scattering (Scattering_1d) and connection (Connection_1d). The procedure Display_result displays the results of the simulation on screen. The outside loop controls simulation steps, through the variable 's'. The farther the interesting obstacle is from a source, the more the simulation steps are needed for the simulation.

```
Procedure TLM_Simulation_1d();
begin
  for step := 1 to s do  \{ where s is the number of simulation steps\}
    Source_emit(incidence);
    Scattering_1d(incidence, reflect);
    Connection_1d(reflect, incidence);
    Display_result(incidence);
end TLM_Simulation_1d;
```

**Algorithm 2.4** TLM simulation model in one-dimension.

In a two-dimensional simulation, the main procedure (algorithm 2.8) is similar to that in one-dimension (algorithm 2.4), including scattering and connection steps. Algorithm 2.5 is a procedure to simulate the connection step, implementing Equations 2.65-2.68. The type 'Field' is defined as the following structure:

```
type Direction = record
  dir : array[1..4] of real;
end;
Field = point to array [1..n, 1..m] of Direction;
```

Similarly to waves in one-dimension, the value from the boundary is assumed to be zero. Though it is not a correct simulation, it will not result in a problem when waves do
not propagate beyond the simulation window. In Chapter 4 this problem will be discussed in detail.

Procedure Connection_2d( reflect: Field; Var incidence: Field);
begin
    for i := 1 to n do  
        { where n is the width of a simulation window}
        for j := 1 to m do  
            { where m is the height of a simulation window}
                if (i = 1) then
                    incidence^[i,j].dir[1] := 0.;
                else
                    incidence^[i,j].dir[1] := reflect^[i-1,j].dir[3];
                end;
                if (j = 1) then
                    current^[i,j].dir[2] := 0.;
                else
                end;
                if (i = n ) then
                    current^[i,j].dir[3] := 0.;
                else
                    incidence^[i,j].dir[3] := reflect^[i+1,j].dir[1];
                end;
                if (j = m ) then
                    current^[i,j].dir[4] := 0.;
                else
                end;
        end;
    end;
end Connection_2d;

Algorithm 2.5 Simulation of connection process with TLM model in two-dimensions.

The condition lines if - then - else in Algorithm 2.5 are used to judge whether current cells are on the boundary of arrays. To eliminate these lines and speed up calculation, we can expand the size of the array, that 'reflect' points to, from the range reflect^[1..n, 1..m] to the range reflect^[0..n+1, 0..m+1], and initialise the values in theses
extra cells to zero. So that the condition lines can be removed from Algorithm 2.5, it can be rewritten as:

```
Procedure Connection_2d(reflect: Field; Var incidence: Field);
begin
  for i := 1 to n do
    for j := 1 to m do
      incidence↑[i,j].dir[1] := reflect↑[i-1,j].dir[3];
      incidence↑[i,j].dir[3] := reflect↑[i+1,j].dir[1];
  end;
end Connection_2d;
```

**Algorithm 2.6** Simulation of connection process with TLM model in two-dimensions.

Algorithm 2.6 executes faster than Algorithm 2.5, but with the small cost of increased memory use. It saves n * m condition evaluations, in exchange for 16 (2n + 2m + 4) bytes of memory space.

Algorithm 2.7 is a procedure to simulate the scattering step, implementing Equations 2.64. The boolean array 'transmission' consists of an array of flags to denote whether a cell is connected with its neighbours. If the cell is on an open line, no waves will come to this cell. This simulates an obstacle.
Procedure Scattering_2d(incidence:Field; Var reflect Field);
begin
  for i := 1 to n do
    for j := 1 to m do
      sum := 0.;
      if (transmission[i,j]) then
        for k := 1 to 4 do
          sum := sum + incidence[i,j].dir[k];
        end
        sum := sum / 2.;
      end;
      for k := 1 to 4 do
        reflect[i,j].dir[k] := sum - incidence[i,j].dir[k];
      end;
  end;
end Scattering_2d

Algorithm 2.7 Simulation of scattering process with TLM model in two-dimensions.

Algorithm 2.8 is the main procedure of the TLM simulation model in two-dimensions.

Procedure TLM_Simulation_2d();
begin
  for step := 1 to s do
    Source_emit(incidence);
    Scattering_2d(incidence, reflect);
    Connection_2d(reflect, incidence);
    Display_result(incidence);
  end;
end TLM_Simulation_2d;

Algorithm 2.8 TLM simulation model in two-dimensions.
The above algorithms (Algorithms 2.1 to 2.8) are the main part of a real simulation system, but many optional parameters are not included in them.

2.3 Conclusion

The TLM model can be used to simulate acoustic motion in air. The theory which underlines this model is easy to understand. The algorithm is very simple, as well as being easy to develop and maintain. As it is an iterative calculation, it is suitable for producing time sequences of images.

Like all other numerical techniques, the TLM model is subject to various sources of error, such as truncation error, velocity error and coarseness error. Hoefer gave a detailed description about these errors and their possible correction in his paper (Hoefer, 1985).

The computing time and memory requirements are analysed as follows.

2.3.1 Complexity in one-dimension

2.3.1.1 Space complexity The simulation model in one-dimension needs two arrays, typed ‘Array1d’. Its size is 2 * n * 4 bytes, where n is the width of the simulation space and a data of ‘real’ type is 4 bytes. Two arrays need 16 * n bytes. The array impedance needs 4 * n bytes. The simulation model needs memory space of:

\[ 20n \text{ bytes} \]

2.3.1.2 Time complexity The time complexity of Algorithm 2.4 is O(s) where s is the number of simulation steps. Every loop in this algorithm contains four procedures. The second and the third are main parts in loops. The second procedure is the Algorithm
2.2, and its time complexity is $O(n)$ where $n$ is the width of the simulation space. The third is Algorithm 2.3, and its time complexity is $O(n)$. Therefore, the entire time complexity for TLM model in one-dimension is:

$$O(s \times n).$$

2.3.2 Complexity in two-dimensions

2.3.2.1 Space complexity In the simulation model, the main data type is 'Field', which requires $m \times n \times 4 \times 4$ bytes, where $m$ and $n$ are the height and width of a simulation window. The data 'reflect' and 'incidence' in Algorithms 2.5 and 2.7 need $32 \times m \times n$ bytes, but the data 'reflect' in Algorithm 2.6 needs $(m + 2) \times (n + 2) \times 4 \times 4$ bytes. The data 'transmission' in Algorithm 2.7 is a pointer pointing to a boolean array, size of $n \times m$. The simulation model, for the main parts, with Algorithms 2.6, 2.7 and 2.8 need:

$$16mn + 16(m + 2)(n + 2) + mn / 8 \text{ bytes.}$$

The main limitation of this TLM simulation model is the storage space. For example, if the size of a visual window is $300 \times 400$ cells. The memory space required is:

$$16mn + 16(m + 2)(n + 2) + mn / 8$$

$$= 16 \times 300 \times 400 + 16 \times 302 \times 402 + 300 \times 400 / 8$$

$$= 3877464 \text{ bytes} = 3.7 \text{ Mbytes.}$$

2.3.2.2 Time complexity The time complexity of the main loop in Algorithm 2.8 is $O(s)$. That of Algorithm 2.6 is $O(m \times n)$, and Algorithm 2.7 is $O(m \times n)$. Therefore, the time complexity of the simulation model is:

$$O(s \times m \times n).$$
Chapter 3
Continuous Models

To obtain more accurate models of both chirp envelope motion and detailed wave interference, a system using Continuous models has been developed. The physical properties of sound waves have been studied for a long time (Skudrzyk, 1971), and consequently most physics books include the equations for wave motion (Tipler, 1991). The properties of sound waves are similar in many aspects to light waves. Therefore, the properties developed in those areas are readily applied to acoustics with modification for medium properties and wave energy characteristics.

3.1 Wave equation

Sound propagation is described by the wave equation, which gives a description of wave motion in a homogeneous medium, and from it one can derive wave velocity. Furthermore, the wave equation is basic to many forms of wave motion.

The linear wave equation is in the form:

$$\nabla^2 f = \frac{1}{V_\varphi^2} \frac{\partial^2 f}{\partial t^2}$$

where

- $f$ is the physical property that varies as the wave travels through the medium,
- $V_\varphi$ is the propagation velocity of waves,
- $\nabla^2$ is the divergence of the gradient,
- $\nabla^2 = \frac{\partial^2 f}{\partial x^2}$ for one-dimension,
- $\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$ for two-dimensions,
- $\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2}$ for three-dimensions, and
This equation applies in general to various types of waves moving through nondispersive media.

**3.1.1 Solution for the wave equation in one-dimension**

The function representing an outgoing wave, caused by the harmonic wave $A \sin(\omega t)$ at $x = 0$, is

$$f = A \sin(\omega t - kx) = A \sin[\omega(t - \frac{x}{V_p})]$$

where

- $\omega$ is the angular frequency, and
- $k$ is the wavenumber, $k = \frac{\omega}{V_p}$.

**3.1.2 Solution for the wave equation in two-dimensions**

The wave equation in two-dimensions describes the wave motion from a cylindrical wave source. Although the cylindrical wave source and cylindrical wave propagation are three-dimensional, the two dimensional equations describe the solution accurately. The solution does not depend on the third dimension. As their wave components in the third dimension are constant, their mathematical expression is a form in two-dimensions. The intensity of a cylindrical wave decreases as a function of range $r$ and the pressure as a function of $\sqrt{r}$ (Morse and Ingard, 1948):

$$f = F e^{-i\omega} [J_0(kr) + iN_0(kr)]$$

$$\lim_{r \to \infty} \frac{2}{\pi kr} e^{i(kr - \omega t) - i(\pi/4)}$$

$$\lim_{r \to 0} \frac{2F}{\pi} \ln(r) e^{-i\omega}$$
where

\[ F \text{ is constant,} \]
\[ \omega \text{ is the angular frequency,} \]
\[ k \text{ is the wavenumber,} \]
\[ r \text{ is the distance from the source,} \]
\[ J_0 \text{ denotes the Bessel function of first kind of zero order,} \]

\[ J_0(kr) = \sum_{i=0}^{\infty} (-1)^i \left( \frac{kr}{2} \right)^{2i} \frac{1}{(i!)^2} \]  

(3.6)

\[ N_0 \text{ denotes the Bessel function of second kind of zero order,} \]

\[ N_0(kr) = \frac{2}{\pi} (\ln \frac{kr}{2} + \gamma) J_0(kr) - \frac{2}{\pi} \sum_{i=1}^{\infty} (-1)^i \left( \frac{kr}{2} \right)^{2i} \frac{1}{(i!)^2} \sum_{m=1}^{\infty} \frac{1}{m} \]  

(3.7)

\[ \gamma = 0.5772156649. \]

The real parts of Equations 3.4 and 3.5 are

\[ f = F \sqrt{\frac{2}{\pi kr}} \cos(kr - \omega t - \pi / 4) \quad (r \to \infty) \quad 3.8(a) \]

\[ f = \frac{2F}{\pi} \ln(r) \sin(\omega t) \quad (r \to 0) \quad 3.8(b) \]

### 3.1.3 Solution for the wave equation in three-dimensions

The function representing a circular outgoing wave, in three-dimensions, is

\[ f = \frac{A}{r} \sin \omega(t - \frac{r}{V \phi}) \]  

(3.9)

where

\[ r = \sqrt{x^2 + y^2 + z^2} \]
The intensity of a spherical wave decreases as a function of $r^2$ and the pressure as a function of $r$.

### 3.2 Reflection

Waves are reflected when they reach a boundary where there is a change in impedance, such as the surface of an obstacle.

#### 3.2.1 Reflected waves in one-dimension

The reflected wave in one-dimensional space is

$$f = -A\sin[\omega(t + \frac{x}{V_p} - \frac{2l}{V_p})]\quad 3.10$$

where

$l$ is the distance from the source to the obstacle.

#### 3.2.2 Reflected waves in two-dimensions

The law of reflection for specular surfaces states that the angle of reflection $\theta_r$ equals the angle of incidence $\theta_i$.

![Figure 3.1](image.png)  
**Figure 3.1** The angle of reflection $\theta_r$ equals the angle of incidence $\theta_i$.

To calculate the wavefront reflected off a surface, instead of using the law of reflection, the mirror equation (Semat, 1966) is used. This equation approximates the
law of reflection when the incident waves strike the obstacle at points in the paraxial area. This approximation speeds up the calculation. The mirror equation describes the relationship between the wavefront curvature of incident waves and reflected waves. That is:

\[
\frac{1}{l} + \frac{1}{l'} = \frac{2}{R}
\]

where

- \(l\) is the distance of the wave source from the surface,
- \(l'\) is the distance of the focus point from the surface, and
- \(R\) is the radius of the surface.

**Figure 3.2** Graphical representation of mirror equation for a point source \(S\) and a concave surface. \(S'\) is the focal point for the reflected paraxial rays.

In the case of a concave surface, the reflected rays near the axis pass through a focal point in front of the surface (\(S'\) in Figure 3.2). In the case of a flat surface, the reflected spherical waves appear to come from a virtual source behind the surface. This virtual source, called the imaginary point, is denoted by \(S'\) in Figure 3.3. It is the same distance from the flat reflector as the source \(S\) is.
Figure 3.3 Spherical wave reflected from a flat surface appears to come from the virtual source $S'$.

Spherical waves reflected from a convex curved surface appear to originate from a focal point behind the obstacle but closer to the surface than the source (Figure 3.4).

Figure 3.4 Spherical wave reflected from convex surface appears to originate at the virtual source $S'$.

Figures 3.5 and 3.6 show time sequences of reflection from a specular concave curved surface and an edge formed by two specular flat surfaces.
Figure 3.5 Time sequence showing plane waves reflected off a concave surface.
The mirror equation is used to simulate wave reflection off obstacles with simple shapes, i.e. planes and arcs. For more complex shapes, algorithms will be developed in Chapter 5 to calculate the amplitude of a reflected wave from the curvature of the object.

3.3 Diffraction

When waves encounter an edge of a surface, they tend to bend around the edge. This is an example of the phenomenon known as diffraction. When the waves pass a surface, a
shadow region is created, and diffraction is apparent (Region B in Figure 3.7). Interference is apparent when the diffracted waves are superimposed on the incident and reflected waves near the edges of the surface (Region D in Figure 3.8).

Diffraction can be seen in the shadow area behind a surface, where the pressure amplitude is very small. One effect of diffraction is to increase the wave amplitude before the shadow edge. Before the edge, the wave amplitude oscillates about its value with increasing amplitude. It reaches its maximum just before the edge of the shadow, and then drops monotonically to approach zero well inside the shadow. Similar diffraction effects occur when a wave reflects off an edge of a surface.

An experiment has been done with the following equation to display the diffraction from a knife edge (Morse, 1968):

\[
P(r, \phi) = A e^{-ikr \cos \Psi} E[\sqrt{2kr} \cos \frac{1}{2} \phi] + A e^{ikr \cos(\Psi + 2\Psi)} E[\sqrt{2kr} \cos(\frac{3}{2} \pi - \Psi - \frac{1}{2} \phi)]
\]

where

\[ P(r, \phi) : \text{wave pressure at a receiver point } (r, \phi) \text{ relative to the edge (Figure 3.7),} \]
\[ r : \text{the distance from the knife edge to the receiver point,} \]
\[ \phi : \text{the angle of the direction of incident waves to the direction of the receiver point,} \]
\[ \Psi : \text{the angle of the obstacle plane with the orthogonal direction of incident waves,} \]

\[ E(z) \text{ is defined as} \]

\[
E(z) = \frac{1}{\sqrt{i\pi}} \int_{-\infty}^{z} e^{u^2} du = \begin{cases} 
1 - \frac{1}{2} [1 - C(z^2) - S(z^2)] - \frac{1}{2} i [C(z^2) - S(z^2)] & z \geq 0 \\
\frac{1}{2} [1 - C(z^2) - S(z^2)] + \frac{1}{2} i [C(z^2) - S(z^2)] & z \leq 0 
\end{cases}
\]

The Fresnel integrals \( C(z^2), S(z^2) \) are defined as:

\[
C(w) = \int_{0}^{w} \cos\left(\frac{\pi u^2}{2}\right) du
\]
As shown in Figure 3.7, we can consider three distinct regions: reflection, transmission and shadow regions. In the reflection region, the diffracted waves interfere with both the reflected and incident waves. In the transmission region, the diffracted waves interfere with the incident waves. In the shadow region, only diffracted waves exist.

Equation 3.12 is used to model the diffraction which occurs on both the transmitted and reflected waves. The results for waves from a knife edge are illustrated in Figure 3.8. The diffraction of the transmitted waves can be seen in the shadow region (Region B in Figure 3.8). The diffraction of the reflected waves is visible in the area between the reflection region and the transmission region (Region D in Figure 3.8).
Figure 3.8 Wave reflection and diffraction, showing A the transmitted waves, B the diffracted waves from a knife-edge, C the reflected waves, D the diffracted waves from the reflected waves, E interference of the incident waves with the reflected waves. Note that only a partial wave train has been transmitted.

According to Huygens' wave construction principle, each point at a wavefront is considered to be the source of secondary wavelets that propagate outward. With this principle, a solution for waves going through an aperture can be obtained. We can consider that the pressure at a receiver point behind an aperture is the sum of the pressures from the sources of secondary wavelets at the aperture. The pressure \( dP \) at a receiver point (Figure 3.9) from a single secondary source is:

\[
dP = k f_1 \left( \frac{A}{r_o} \right) e^{ikr_0} \left( \frac{1}{r_s} \right) e^{ikr_s} \left( e^{-i\omega t'} - e^{-i\omega t} \right) dS
\]

where \( k \) is the proportionality coefficient,

\( f_1 \) is the obliquity factor,

\( A \) is the wave amplitude at a unit distance from the source,

\( r_s \) is the distance from the receiver point to the secondary source \( dS \),

\( r_o \) is the distance from the wave source to the secondary source \( dS \),

\( t' \) is the time for waves travelling from the source to \( dS \), and

\( t \) is the time for waves travelling from the source to the receiver point.
The terms $k f_1$ in Equation 3.16 are an unknown function $f_1$ for the obliquity factor and a proportionality coefficient $k$. These represent diffraction. The Fresnel-Kirchhoff diffraction formula, derived from the Kirchhoff approximation, gave a solution for $k f_1$, and the equation for the pressure at a receiver point (Figure 3.10) is (Elmore and Heald, 1969):

$$P = -\frac{iA}{\lambda} e^{-i\omega} \int_{S} \left( \frac{\cos \theta_s + \cos \theta_o}{2} \right) \frac{e^{i(r_s^2 + r_o^2)}}{r_s + r_o} dS$$

3.17

where $P$ is the pressure at the receiver point,(Figure 3.10)

$\theta_s$ is the angle between the wave source and the normal of the aperture,

$\theta_o$ is the angle between the receiver point and the normal of the aperture,

$r_s$ is the distance from the receiver point to the aperture,

$r_o$ is the distance from the wave source to the aperture,

$S$ is the surface around the receiver point,

$\Delta S$ is the aperture width, and

$\frac{\cos \theta_s + \cos \theta_o}{2}$ is known as the obliquity factor.
This model is included for completeness but was not used in any simulation for this thesis. A future project could compare various models of diffraction.

### 3.4 Interference

Another important area of wave simulation is the interference of waves, as this allows us to study wave beam forming by arrays of sources. In a linear medium, one can apply the principle of superposition to obtain the resultant disturbance. The principle of superposition states that the actual displacement of any part of the disturbed medium equals the algebraic sum of the displacements caused by the individual waves. This principle can be applied to many types of waves, including sound waves.

Suppose we have waves $P_1$, $P_2$ emitted from two sources,

\[ P_1 = P(r_1, t) \]  \hspace{1cm} 3.18

\[ P_2 = P(r_2, t) \]  \hspace{1cm} 3.19

where $r_1$ and $r_2$ are the distances of the wavefronts from the two sources at time $t$.

The interference wave $P_{\text{int}}$ is
\[ P_{\text{int}} = P_1 + P_2 = P(r_1, t) + P(r_2, t) \] 

For example, Figure 3.11 shows the interference between waves from two point sources.

![Wave interference from two point sources.](image)

**Figure 3.11** Wave interference from two point sources.

### 3.5 Directional characteristic of sound source

The pressure of sound waves emitted from a circular piston depends on orientation. The pressure is the strongest in the direction of propagation. The mathematical equation for the direction characteristic is (Kuttruff, 1991, p. 66):

\[ \Gamma(\theta) = \frac{2J_1(ka \sin \theta)}{ka \sin \theta} \]
where

\[ \Gamma(\theta) \] is directivity function of a piston source,

\[ J_1 \] is the Bessel function of first order,

\[ k \] is the wavenumber,

\[ a \] is the radius of the piston source, and

\[ \theta \] is the angle to the axis of the piston.

Figures 3.12 and 3.13 show a 3D picture of this directivity function using ray tracing technology. Three light sources are used. The light in front is red, right top is green and the right bottom is blue, in order to strengthen the three dimensional effect.

**Figure 3.12** Directivity function of a piston source is depicted in 3D, where \( ka = 7.5 \).

**Figure 3.13** Directivity function of a piston source is depicted in 3D, where \( ka = 15 \).

It should be well noted that Equation 3.21 only describes the characteristics of wave pressure, in the far field, emitted from a circular piston with a pulse containing a single frequency tone.

### 3.6 Conclusion

When using the Continuous models, each wave property must be modelled separately and then the results combined with superposition to produce the data for visualisation (for
example Figure 3.8). The mirror equation, which approximates the law of reflection when the incident waves strike an obstacle at points in the paraxial area, is used to simulate the reflected wavefront if the obstacle is of plane shape or of arc shape. For a more complex curved shape, the method to calculate the reflected wavefront is derived in Chapter 5. Equation 3.12 can be used to simulate the wave diffraction from a knife edge and Equation 3.17 can be used from an aperture. The principle of superposition can be used to simulate the wave interference.

The simulation work for reflection mentioned above only takes account of specular surfaces. In real environments, some surfaces are too rough for specular reflection. For rough surfaces, we can use the Lambertian model of diffusion (Nayar et al, 1990). Lambert's law states that for any isotropic surface flux phenomenon (i.e., for a flux such that the amount of property emitted by the surface per unit time per unit area perpendicular to the direction of emission is independent of the direction of emission), the amount of that property emitted per unit area of the surface must be proportional to the cosine of the angle between the direction of emission and the normal to the surface (Cunningham et al, 1980).
Chapter 4
Lattice Gas Model

4.1 Introduction

The Lattice Gas (LG) models are a special class of Cellular Automata. Cellular Automata began in about 1950, when John von Neumann set himself the task of proving the possibility of self-duplicating automata. Such a machine, given proper instructions, would build an exact duplicate of itself. Each of the two machines would then build another, the four would become eight, and so on. Von Neumann first proved his case with "kinematic" models of a machine that could roam through a warehouse of parts, select needed components and put together a copy of itself. Later, adopting an inspired suggestion by his friend Stanislaw M. Ulam, he showed the possibility of such machines in a more elegant and abstract way (Gardner, 1971).

The new proof of Von Neumann used what is now called a "uniform cellular space" equivalent to an infinite checkerboard. Each cell can have any finite number of "states", including a "quiescent" state, and a finite set of "neighbour" cells that can influence its state. The pattern of states changes in discrete time steps according to a set of "transition rules" that apply simultaneously to every cell. The cells symbolise the basic parts of a finite-state automaton and a configuration of live cells is an idealised model of such a machine. Von Neumann, by applying transition rules to a space in which each cell has 29 states and four orthogonally adjacent neighbours, proved the existence of a configuration of about 200,000 cells that would self-reproduce (Gardner, 1971).

Cellular automata consist of a lattice, each site of which can have a finite number of states; the automaton evolves in discrete steps, the sites being simultaneously updated by a deterministic or nondeterministic rule. Only a finite number of neighbours are involved in the updating of any site. It can be described formally by the 5-tuple (Tzionas et al., 1994):
\{ I, Z, Q, \delta, \omega \}

where

$I$: the set of inputs meaningful to the automaton,

$Z$: the set of outputs generated by the automaton,

$Q$: the set of discrete internal states of the automaton,

$\delta$: the function that relates every pair of elements taken from sets $I$ and $Q$, i.e., $i$ and $q_t$ respectively, to the next element of $Q$, i.e., $q_{t+1}$, and

$\omega$: the function that relates every pair of elements $i$, $q_t$ to an element of $Z$, i.e., $z_t$.

Cellular automata can be implemented in massively parallel hardware. The class of cellular automata used for the simulation of fluid dynamics is called "lattice gas models".

### 4.2 Lattice Gas model

The LG model was developed by Hardy et al. (Hardy et al., 1976) as a means of rapidly calculating the Navier-Stokes equation for large computer simulations of fluid flow. Kadanoff and Swift (Frisch et al., 1987) were the first to apply it to modelling wave motion. They used a continuous time model of particle motion between cells in a grid. Krutar et al. (1991) extended this work to develop a discrete time model of wave motion by combining groups of particles into differential pressures that propagate through the network.

The environment is modelled as a network of cells (Figure 4.1). We only consider cells connected in a grid of squares but other connection are possible (Krutar et al., 1991). At each cell a set of equations that represent the state of the environment at that cell is calculated. Possible states include sound source, media 1..n, media boundary and simulation window edge.
Krutar et al.'s LG model provides a finite difference solution (Equation 4.1) for the acoustic wave equation (Equation 4.2) to calculate the pressure at each cell at each time step.

\[ dP(x,t + dt/2) = P(x,t + dt) - P(x,t) \quad 4.1(a) \]

\[ ddP(x,t) = dP(x,t + dt/2) - dP(x,t - dt/2) \quad 4.1(b) \]

\[ ddP(x,t) = \sum_a m_a P(x + dx_a, t) \quad 4.1(c) \]

where

- \( P(x,t) \) is the pressure at time \( t \) and location \( x \), equivalent to an integer number of particles,
- \( dt \) is the time step,
- the subscript \( a \) represents one of the five directions, N, S, E, W, and at the cell (Figure 4.1),
- \( m_a \) is weighted coefficient of the pressure \( P \) at the direction \( a \), \( m_0 = -c_x^2 \),
- \( c_x \) is the sound speed at cell \( x \), and
- \( dP \) is the pressure derivative with respect to time \( t \).

The acoustic wave equation (Equation 4.2) expresses the double derivative of the pressure of a wave at an instant in time as a function of the speed of sound in the medium and the double derivative of the pressure with respect to the position of the particles that move to propagate the wave.

\[ \frac{\partial^2 P}{\partial^2 t} = c^2 \nabla^2 P. \quad 4.2 \]
Figure 4.1 Grid of cells used in LG model.

We can derive the discrete form of the wave equation (Equation 4.1(c)) from the continuous form (Equation 4.2) as follows. First, we define

\[ m_N = m_S = m_E = m_W = c^2 \quad \text{and} \quad m_0 = -4c^2, \]

where \( c \) is the speed of sound at location \((x,y)\), then the summation in Equation 4.1(c) can be rewritten as:

\[
\frac{m_E P(x + dx_E, t) + m_W P(x + dx_W, t) + m_N P(x + dx_N, t) + m_S P(x + dx_S, t) + m_0 P(x + dx_0, t)}{d^2x} + \frac{m_N P(x + dx_N, t) + m_S P(x + dx_S, t) + m_0 P(x + dx_0, t)}{d^2y} = c' \left[ \frac{P(x + dx_E, t) + P(x + dx_W, t) - 2P(x + dx_0, t)}{d^2x} + \frac{P(x + dx_N, t) + P(x + dx_S, t) - 2P(x + dx_0, t)}{d^2y} \right]
\]

From a Taylor series expansion, we find the finite difference form of the second order partial derivative of \( P \) with respect to \((x,y)\):

\[
\frac{\partial^2 P(x,y,t)}{\partial^2 x} + \frac{\partial^2 P(x,y,t)}{\partial^2 y} = \frac{P(x + dx, y, t) + P(x - dx, y, t) - 2P(x, y, t)}{d^2x} + \frac{P(x, y + dy, t) + P(x, y - dy, t) - 2P(x, y, t)}{d^2y},
\]

Therefore, Equation 4.4 can be rewritten as:
The right side of Equation 4.1(c) can be regarded as a discrete form of the wave equation. From Equation 4.1(a) we obtain,

\[ dP(x, t - dt/2) = P(x, t) - P(x, t - dt) \]  

Substituting Equations 4.1(a) and 4.7 into the right side of Equation 4.1(b), we get,

\[ ddP(x, t) = P(x, t + dt) - P(x, t) - (P(x, t) - P(x, t - dt)) = P(x, t + dt) - 2P(x, t) + P(x, t - dt) \]

Equating Equations 4.1(c) and Equation 4.8, and rewriting the resultant equation, we get,

\[ P(x, t + dt) = 2P(x, t) - P(x, t - dt) + \sum a m_t P(x + dx_a, t) \]

Replacing \( m_t \) in Equation 4.9 with the simulation speed \( c_s \), the two-dimensional form of Equation 4.9 is,

\[ P_0(t + 1) = 2P_0(t) - P_0(t - 1) - 4c_{s0}^2 P_0(t) + c_{sW}^2 P_W(t) + c_{sE}^2 P_E(t) + c_{SN}^2 P_N(t) + c_{SN}^2 P_S(t) \]  

where

\( P_0(t) \) is the wave pressure at location \( x \) and time \( t \),

\( c_{s0} \) is the simulation speed (cells propagated / simulation step) at location \( x \),
subscript $w$ refers to the all on the west side, $e$ east, $n$ north and $s$ south (Figure 4.1).

Equation 4.10 is a discrete form of the wave equation which discards higher order terms in the Taylor series expansion so it does not implement it completely. As a result some noise has been brought into the results.

In a square grid, the maximum speed of the simulated wave is 0.707 cells per time step (Krutar et al., 1991). Thus, the maximum speed of sound in any of the medium in the simulation is equivalent to this rate. In practice, the user wants to specify the frequency ($f$), spatial resolution (model scale) and the medium used in the simulation. From this, the simulator calculates the simulation rate, based on the maximum speed of sound ($c_{max}$) in any of the medium.

\[
\text{simulate rate } R = \frac{c_{max} \times S}{0.707} \text{ steps / second}, 
\]

\[
\text{model scale } S = \frac{m \times f}{c_{max}} \text{ cells / meter}, 
\]

where $m$ is the number of cells / wavelength.

The three steps (Equations 4.1(a), 4.1(b) and 4.1(c)) of Krutar's model have been combined into one iterative formula (Equation 4.10). However, the coefficients have been changed to enable the speed of sound to specified at each cell. This finite difference model of wave propagation is implemented in Algorithm 4.1.
Procedure Propagation()

(the array ‘P’ is the pressure $P_a(t)$,
the array ‘P_new’ is the pressure $P_a(t+1)$,
the array ‘P_old’ is the pressure at the previous iterative step $P_a(t-1)$, and
the array ‘Cx’ is the speed $c_{xa}$.)

begin
...
for $i := 1$ to $n$ do
  for $j := 1$ to $m$ do
    $P_{\text{new}}[i, j] := Cx[i, j+1] * P[i, j+1] + Cx[i, j-1] * P[i, j-1] + Cx[i+1, j] * P[i+1, j]$
    $+ Cx[i-1, j] * P[i-1, j] - 4 * Cx[i, j] * P[i, j] + 2 * P[i, j] - P_{\text{old}}[i, j]$;
    Update(i);
  end;
end; Propagation

Algorithm 4.1 The main part of the Propagation procedure.

The procedure ‘Update’ (Algorithm 4.2) performs the task of updating arrays $P_{\text{old}}$, $P_{\text{new}}$ and $P$. Algorithm 4.1 shows that four arrays, ‘P’, ‘P_new’, ‘P_old’, and ‘Cx’ are needed. The memory space is $4 * 4 * (m + 2) * (n + 2)$ bytes. In fact, the array ‘P_new’ has a size of [0..1, 0..m+1]. Therefore, the memory space is required is

$$4(3(m + 2)(n + 2) + 2(m + 2)) = 4(3mn + 8m + 6n + 16) \text{ bytes}$$

4.13
**Algorithm 4.2** Update pressure values in arrays.

**Algorithm 4.3** The main loop of the LG model.

The LG_Simulation is implemented with Algorithm 4.3. Source_emit models the wave source(s). Propagation (Algorithm 4.1) implements the LG model, and Display_result produces the visualisation output. Figure 4.2 was produced with Algorithm 4.3. It shows a visualisation of the output of a Polaroid transducer for a 50 kHz wave at 20°C. The transducer is modelled as a line of cells of length equal to the diameter of the transducer. The lobe pattern and spreading beam in the far field is clearly seen in this visualisation. In Figure 4.2a, each source cell has the same pressure amplitude. In Figure 4.2b, the pressure in cells at the ends of the line are tapered to zero. Notice the difference in the side lobes. By weighting the pressure in the cells the simulation of the transducer can be adjusted to match measured lobe patterns.
Figure 4.2 Visualisation of a Polaroid sensor showing beam and side lobes with linear mapping from normal pressure amplitude (-1.0 .. 1.0) to grey scale (0 .. 255). The surface of the sensor is the line on the left made up of 104 point source cells. a) all source cells have the same pressure amplitude; b) the pressure in 10 source cells at each end of the line are tapered to zero.
4.3 Boundary rules

For the LG model to simulate all wave motion properties of interest it must model wave propagation, interference, reflection and diffraction. The wave equation models propagation and interference and as shown above is implemented by Equation 4.10. Diffraction is the gradual spreading out of sound waves. It is simulated in the LG model by the fact that each cell is a weighted sum of its neighbours. Reflection occurs when the impedance of the media changes at a boundary. An impedance change is usually accompanied by a speed change.

When a sound wave impinges on a boundary between two media, the speed of sound and the acoustic impedance change. Equation 4.10 only models the change of speed. To model the change of impedance, the boundary cells must include values to calculate the reflection and transmission coefficients. These are then used to calculate the amplitudes of the reflected and transmitted waves.

The amount of energy reflected at a boundary, and hence the amount transmitted through the boundary, is a function of the relative impedances of the two media and the incident angles. As the environment is modelled as a rectangular mesh, only waves at normal incidence need to be considered. This assumes that a wave incident at any other angle will be modelled by two components of pressure. In the case of normal incidence, the reflection coefficient $R$ for the waves travel from medium-1 to medium-2 is defined as (Burdic, 1984, p. 100):

$$ R = \frac{Z_1 - Z_2}{Z_1 + Z_2} \quad 4.14 $$

where

- $Z_1$ is the impedance in the medium 1 and
- $Z_2$ is the impedance in the medium 2.

The transmission coefficient is defined as:
\[ T = \frac{2Z_2}{Z_2 + Z_1} = 1 + R \] 4.15

In the following sections, a novel set of boundary rules to carry out these relations will be discussed. They will be divided with two sorts of rules: one deals with a boundary with a smooth interface, the other with a rough interface.

4.3.1 Simulation for a smooth boundary

In this section, rules at cells near a smooth boundary are discussed, and examples follow.

4.3.1.1 Rules at cells near a smooth boundary  In the Figure 4.3, cells 2, 6, 10 and 3, 7, 11 are near a boundary.

![Figure 4.3 A boundary and cells around it.](image)

Let us consider cell 6, referring to Equation 4.10:

\[ P_6(t + 1) = 2P_6(t) - P_6(t - 1) - 4c_6^2 P_6(t) + c_2^2 P_2(t) + c_5^2 P_5(t) + c_{10}^2 P_{10}(t) + c_7^2 P_7(t). \] 4.16

The last term in Equation 4.16, \( c_7^2 P_7(t) \), is a component from the boundary. It should be composed of two parts, one transmitted from cell 7, the other reflected from cell 6 itself. This term should be replaced by
\[ c_6^2 R_{12} P_6(t) + c_7^2 T_{21} P_7(t) \]

where \( R_{12} \) is the reflection coefficient of a pressure wave travelling from medium-1 to medium-2, and \( T_{21} \) is the transmission coefficient of a pressure wave travelling from medium-2 to medium-1. The entire expression of the rule being applied to cell 6 is

\[
P_6(t+1) = 2P_6(t) - P_6(t-1) - 4c_6^2 P_6(t) + c_2^2 P_2(t) + c_5^2 P_5(t) + c_{10}^2 P_{10}(t) + c_6^2 R_{12} P_6(t) + c_7^2 T_{21} P_7(t)
\]

Similarly, the rule at cell 7 is:

\[
P_7(t+1) = 2P_7(t) - P_7(t-1) - 4c_7^2 P_7(t) + c_5^2 P_5(t) + c_{11}^2 P_{11}(t) + c_7^2 R_{21} P_7(t) + c_6^2 T_{12} P_6(t)
\]

### 4.3.1.2 Implementation results

Several simulation examples are given in Figure 4.4. (a) was produced with the normal iterative formula (Equation 4.10), and (b), (c) and (d) with the novel rules (Equations 4.18 and 4.19). The speeds are 331 m/s in both media. The impedances for medium-1 and medium-2 are (a) 4,466,000 Ns/m³, (b) 4,466,000 Ns/m³, (c) 466,000, 4 Ns/m³, and (d) 4, 4 Ns/m³ respectively. The reflection coefficient in (a) and in (b) is:

\[
R = \frac{Z_2 - Z_1}{Z_2 + Z_1} = \frac{466000 - 4}{466000 + 4} \equiv 1
\]

As the normal iterative form (Equation 4.10) does not take the impedance into account, so that no reflection can be seen in (a), it is wrong. In (b), the reflected waves are observed, with a similar amplitude as that of incident waves. In (c), the reflection coefficient approximates -1. The reversed phase of the reflected pressure waves can be seen. As the transmission coefficient approximates 0, no pressure is transmitted. In (d) the impedances are set to a same value, therefore, no reflection occurs. The picture looks
same as in (a). Thus, in this situation, the new rules produce the same results as the normal rules (Equation 4.10).

Figure 4.4 Equation 4.10 is used to produce (a). Equations 4.18 and 4.19 are used to produce (b), (c) and (d). The speeds in medium-1 and in medium-2 are the same, 331 m / s, in these four simulations.

(b) The impedance in medium-1 is 4 Ns / m³, in medium-2 is 466000 Ns / m³;

(c) The impedance in medium-1 is 466000 Ns / m³, in medium-2 is 4 Ns / m³; and

(d) The impedances in medium-1 and medium-2 are the same.
4.3.2 Simulation for a rough boundary

To simulate wave behaviour at a rough boundary, we can set the boundary in a zigzag shape (See Figure 4.5).

![Figure 4.5](image)

**Figure 4.5** A zigzag interface and cells around it. (a) One-cell zigzag; (b) Two-cell zigzag.

4.3.2.1 Rules at cells near a rough boundary In figure 4.5 (a), cells 3, 4, 6, 7, 11, 12, 14 and 15 are near the boundary. The rules for one pair (cells 6 and 7) of these cells are defined as:

\[
P_6(t+1) = 2P_6(t) - P_6(t-1) - 4c_6^2 P_6(t) + c_2^2 P_2(t) + c_5^2 P_5(t) + c_{10}^2 P_{10}(t) + c_6^2 R_{12} P_6(t) + c_7^2 T_{21} P_7(t) \quad (4.18)
\]

\[
P_7(t+1) = 2P_7(t) - P_7(t-1) - 4c_7^2 P_7(t) + 3c_7^2 R_{21} P_7(t) + T_{12} (c_3^2 P_3(t) + c_{11}^2 P_{11}(t) + c_6^2 P_6(t)) \quad 4.20
\]

More generally, we have:

\[
P_0(t+1) = 2P_0(t) - P_0(t-1) - 4c_0^2 P_0(t) + \sum_a (c_a^2 T_a P_a(t) + 4c_a^2 R_a P_0(t)) \quad 4.21
\]
where subscript $a$ notates left, right, top or bottom side of a cell. For example, $T_{\text{left}}$ is the transmission coefficient when pressure waves travel from the left side. If the medium in the left side is same as that the cell is in, $T_{\text{left}} = 1$. Similarly, $R_{\text{left}}$ notates the reflection coefficient when pressure waves go to left. If the medium in the left side is same as the cell is in, $R_{\text{left}} = 0$.

4.3.2.2 Implementation results Figure 4.6 demonstrates results from implementing these algorithms. Picture (a) shows a simulation involved a smooth boundary for the sake of comparison; (b) is a rough boundary being simulated with a one-cell zigzag interface; (c) is simulated with a two-cell zigzag interface. Diffusion in these pictures is not apparent. It should be much more apparent with a beam than with a point source, because the spread of the beam should be a function of the diffusion.
Figure 4.6 Simulation for a rough boundary.
(a) a smooth boundary for the sake of comparison;
(b) a rough boundary with a one-cell zigzag interface; and
(c) a rough boundary with a two-cell zigzag interface.

4.3.3 Data structure

As a boundary can be appear at any position to a cell, left, right, above or below, the
following data structure for implementing the above novel boundary rules is proposed:

```plaintext
type node = record
  pressure: real;
  medium: medium_type;
  coefficient[4]: coefficient_direction;
end;
```

where the fields of

- pressure: is used to store the pressure amplitude at a cell;
- medium: is to specify which medium the cell is in;
- medium_type: to define an enumeration type. For example, if there are two
  media in a simulation environment, the 'medium_type' is defined as
{medium1, medium2}. The number of elements in this enumeration set is the number of the media;

coefficient: specifies the index of an array storing reflection or transmission coefficients;

coefficient_direction: defined as an enumeration type. For example, if there are two media in a simulation environment, the ‘coefficient_direction’ is defined as {same_medium, medium1_to_medium2, medium2_to_medium1}. The number of elements in this enumeration set is \( N^2 - N + 1 \), where \( N \) is the number of media;

The array storing reflection or transmission coefficients is:

\[
\begin{array}{c|cc}
\text{Index} & 0 & 1 \\
\hline
\text{same\_medium} & R_{21} & T_{21} \\
\text{medium1\_to\_medium2} & R_{12} & T_{12} \\
\text{medium2\_to\_medium1} & \cdot & \cdot \\
\end{array}
\]

4.4 Window edge problem

The problem with any neighbour cell based simulation model is how to calculate values at the edge of the m*n array of cells. That is, what rules should apply at the edge cells.

This problem is called the window edge problem.

4.4.1 Problem

Since the number of cells is limited, for example, the size of the matrix for the cell lattice is m * n, the values of the cells just outside the window edge ([m+1, j] or [i, n+1], 1 <= i <= m, 1 <= j <= n) cannot be obtained. Yet the calculation for the value at position [i,
or \([m, j]\) requires its neighbour's values, including the value at position \([m+1, j]\) or \([i, n+1]\). This makes it difficult to calculate the value at the boundary of the array. It is different from the common boundary problem, which occurs when the medium changes. Accordingly, solutions commonly used in a boundary problem, for example, finding an initial condition, cannot be used here.

Under the assumption that no waves come in from the edge, the values at edge cells are set to zero. Unfortunately, this causes the edge to behave like a reflector. The edge is not reflective nor absorptive but is meant to be continuous with the rest of the invisible cells in our simulation. If the value at the edge cell is set to zero, it is equivalent to setting the speed at that cell to zero. As it has been pointed out in Section 4.2, the speed change will cause wave reflection. From the point of view of the macroscopic world, we may assume that no waves come in from the edges. However, we cannot assume it to be equivalent of setting the value from the edge direction of the cells to zero. This is because LG is a model of the microscopic world.

4.4.2 Solutions

Several methods have been tried to solve this problem. These are outlined below.

4.4.2.1 Symmetric Duplication method If the edge has symmetric cells inside a window, then we can set cells at the edge to the values of their symmetric cells inside the window. For example, when the source is located near an edge, (column \(n-2\) in Figure 4.7), the cells at columns \(n\) and \(n-4\) are symmetric about the source as there is the same environment at both sides of the source, uniform medium and no obstacles. Thus the cells at the boundary, column \(n\), of the lattice can be set to the values of cells at column \(n-4\).
Figure 4.7 The edge, column n, is symmetric to the column n-4 about the source at the column n-2.

In most cases the environment is asymmetric. The environment showed in Figure 4.8 is not symmetric, though columns n-3 and n-1 are in the same medium and without obstacles. The distances from the obstacle, located at column n-6, to column n-4 and to column n are different, so that the columns n and n-4 are not symmetric about column n-2, the location of the source.

Figure 4.8 The edge at column n is asymmetric to the column n-4 about the source at column n-2, as reflection from the obstacle at column n-6 will effect column n-4 and n differently.

4.4.2.2 Calculation method with the Continuous model If the environment is complex and asymmetric, or it is simple and symmetric, but the source is located at the window centre, then the symmetric part of the window edge cannot be found inside the
window. In this case, the Continuous model, introduced in Chapter 3, can be applied to the window edge. That is, the value is calculated with a mathematical formula which is a function of time and the location of the window edge, independent of the value at its neighbour cells. A simple example follows:

![The left edge is calculated with a continuous model, while the cells at the other three edges are set to zero at the outside direction. It is clearly shown that these three edges behave like reflectors, while the left edge is without reflection, as desired.](image)

**Figure 4.9** The left edge is calculated with a continuous model, while the cells at the other three edges are set to zero at the outside direction. It is clearly shown that these three edges behave like reflectors, while the left edge is without reflection, as desired.

This method used a continuous model to solve the problem in a discrete model. However, if the environment has multi-path reflection or diffraction, the calculation will become very complex, because it has to model the whole environment.

### 4.4.2.3 Speed Tapering method

As the environment is not always symmetric, and the Continuous model has great complexity in calculation time, other simple methods have been derived. Krutar suggests that the speed of sound be tapered to zero in a band of cells near each edge to halt wave motion at the edges. It has been tried to decrease the speed near edges linearly to zero over a range of 30 columns.
for i := 0 to 30 do
    for j:= 1 to n do
        Cx[m - i][j] := speed$^2$ * $i^2$ / 30$^2$;
    end;
end;

**Algorithm 4.4** Speed Tapering applied to the right edge of a simulation window.

The right side in Figure 4.10 shows this method.

![Figure 4.10](image)

**Figure 4.10** Comparison of edge methods. The left edge is simulated by pressure calculated with the Continuous equation, the right edge uses the Speed Tapering, and the top and bottom edges are set to zero.

When this method was applied to simulating the *far field*, on both the top and bottom edges of a window, the result was not satisfactory, because it produced the reflected waves shown in Figure 4.11.
Figure 4.11 The wave source is a line of transmitters, located at the left side, with width of 5.2 wavelengths. After 370 simulation steps, the reflection from both top and bottom is obvious. The speeds in the cells in 30 lines adjacent to the both top and bottom edges are tapered to zero linearly.

Changing the speed non linearly may improve the results, but the variations in speed causes some reflection. The speed at an edge is decreased to zero so no reflection comes from the edge. Reflection will occur wherever the speed is changed, that is, reflection will occur in the region near the edge.

4.4.2.4 Amplitude Tapering method As the speed change causes the reflection, alternatively, it has been tried to decrease the values at cells near edges, so that the amplitude of waves at the edges becomes zero.

For $i := 1$ to $m$ do
    For $j := n - 30$ to $n$ do
        $P_0(t+1) := 2P_0(t) - P_0(t-1) - 4c_{x_0}^2 P_0(t) + c_{xW}^2 P_W(t) + c_{xE}^2 P_E(t) + c_{xN}^2 P_N(t) + c_{xS}^2 P_S(t)$
        $P_0(t+1) := 0.95 \times P_0(t+1)$
    end
end

Algorithm 4.5 Amplitude Tapering method applied to the cells near the top edge of a simulation window.
This results in no waves being reflected.

Figure 4.12 Waves are emitted from a line source set at the left side. The amplitude of cells near the top and bottom sides are multiplied by 0.95 (30 lines nearest the top and bottom). The reflection from these edges is reduced greatly.

This method is faster and has a better result than the Speed Tapering method, although slight reflection still exists. During the iterative process, the factor 0.95 is multiplied by the amplitude of the pressure $P_0(t+1)$ after the calculation with Equation 4.10 has been done. The four speeds $c_s$ in the equation are identical. But the new value $0.95 * P_0(t+1)$ will become the values $P_w(t)$, $P_e(t)$, $P_n(t)$ and $P_s(t)$ in Equation 4.10 of its neighbour cells at the following iterative step. In the calculation formula, the term

$$c_s^2 [0.95 * P(t)]$$

can be considered as

$$[c_s^2 * 0.95] * P(t),$$

that is reducing the pressure is equivalent to changing the speed. This is why the reflection still exists with this method.
4.4.2.5 Amplitude Compensation method Here another compensation method has been provided to improve the Amplitude Tapering method. As multiplying the amplitude by 0.95 changes the speed in the next iterative step in fact, this component is divided in the next iteration with the factor 0.95. The algorithm for this method follows:

\[
\begin{align*}
\text{for } & i := 1 \text{ to } m \text{ do} \\
& \text{for } j := n - 30 \text{ to } n \text{ do} \\
& P_0(t + 1) := 2P_0(t) - P_0(t - 1) - 4c_{s_0}^2P_0(t) \\
& \quad + c_{s_w}^2P_w(t) + c_{s_e}^2P_e(t) + c_{s_n}^2P_n(t) + c_{s_s}^2P_s(t) / 0.95 \\
& P_0(t + 1) := 0.95 \times P_0(t + 1) \\
\end{align*}
\]

Algorithm 4.6 Amplitude Compensation method applied to the cells near the top edge of a simulation window.

Figure 4.13 shows the implementation result:

\[\text{Figure 4.13 Waves emitted for a line source located in the left size, with width of 5.2 wavelengths. The reflection from top and bottom is eliminated with the Amplitude Compensation method. The region near edges are set to 30 lines width.}\]
As the division operation is only applied to its lower neighbour, if the cell nears to the top edge, the amplitude after multiplying and dividing is not completely equal to its original value. The amount of the amplitude propagating to the top edge still decreases. The edge region is set to 30 lines in width, so the amplitude $P(t)$ at the edge has been multiplied with a factor $0.95^{30}$ and becomes quite small. Therefore only quite small amounts of wave will be reflected from the edge.

4.4.2.6 Post History method The previous methods either produce distortion of the wave near the edges or are computationally expensive. An alternate method that is computationally simple and produces no edge distortion but may produce low amplitude reflection at the edges is the Post History method.

In this method, the pressures of the cells on the window boundary are calculated by assuming a value for the non existent cells just outside the boundary. As the boundary is non reflective, all wave motion at the boundary is assumed to be in a direction out of the window. Thus, for a wave normal to the boundary the pressure value in the non existent cell outside the boundary is equal to the pressure value in the boundary cell at the previous time step.

The following rule is used for cells on the top boundary of the window,

$$P_0(t + 1) = 2P_0(t) - P_0(t - 1) - 4c_0^2 P_0(t) + c_0^2 P_0(t - 1) + \sum_a c_a^2 P_a(t)$$

4.22 where subscript $a$ notates W, E, or S side of a cell. Similar rules are used for the other boundaries and the corners.

The result of using this rule is compared to the other methods in Figure 4.14. While this rule is accurate for normal incidence it is only an approximation for waves incident on the boundary at other angles of incidence. However, the error is small and it only results in reflection of waves with small amplitudes. Figures 7.1 to 7.5 display simulations in the far field. The top and bottom edges are calculated with the Post
History method. From these figures, we can see that this method works well in the far field.

![Image](image_url)

**Figure 4.14** Comparison of edge methods. The left edge is simulated by pressure calculated with the Continuous equation, the right edge uses the Speed Tapering, and the top and bottom edges use the Post History method.

### 4.5 Noise removal

It has been noticed that there is a tail behind waves with the cell based simulation models. The fewer cells the wavelength is, the more serious the tails are. In Figure 4.15 (a) to (f), the images with different wavelength are illustrated. The number at the bottom of each image is the number of cells per wavelength.
Figure 4.15 One cycle of waves is emitted from a point source. The tail behind one cycle of waves is more serious in (f) than in (a).
The tail is considered as a sort of noise. One approach to overcome this sort of noise is to set the wavelength in the simulation model no less than 15 cells. Yet this results in an enormous size of the simulation array.

The signal frequency used in a transmitter is 50 kHz, that is 6.8 mm per wavelength. To simulate an obstacle with 1 meter dimension, the number of cells is:

\[
\text{cells for simulating the obstacle} = \text{cells (per wavelength)} \times \frac{\text{obstacle size (mm)}}{\text{wavelength (mm)}}
\]

\[
= 15 \times \frac{1000}{6.8} = 2206 \text{ cells.}
\]

This figure limits this simulation model being applied to simulating a small range. In order to simulate a 5 * 5 meter environment, the simulation array should be of size 11030 * 11030. In this case, we have to set the simulation wavelength less than 15 cells. A non linear colour mapping algorithm is provided to remove the noise from the display when the wavelength is set to less than 15 cells. The amplitude of tails is small comparing with that of the signals, so that if the amplitude of the noise is forced to one value, or the same colour, the oscillation of the noise will not be seen, and also it will not affect the visualisation seriously.

Figure 4.16 Noise removal with a non linear colour mapping method by mapping the amplitudes between the noise thresholds into an identical colour.
The colour mapping algorithm including noise removal method follows:

\[
\text{if (noise\_remove) then}
\]

\[
\text{colour := } (c_r + n_t) \times (p - m_p) / (M_p - m_p);
\]

\[
\text{if (colour < (c_r - n_t) / 2 then } \{ \text{nothing} \};
\]

\[
\text{else if (colour > (c_r + n_t) / 2 then}
\]

\[
\text{colour := colour - n_t;}
\]

\[
\text{else}
\]

\[
\text{colour := c_r / 2;}
\]

\[
\text{end;}
\]

\[
\text{else}
\]

\[
\text{colour := c_r \times (p - m_p) / (M_p - m_p);}
\]

\[
\text{end;}
\]

**Algorithm 4.7** Colour mapping.

where

- \(c_r\): colour range,
- \(n_t\): noise removal threshold,
- \(p\): pressure value,
- \(m_p\): minimum pressure value, and
- \(M_p\): maximum pressure value

Figure 4.17 is an example to demonstrate results from implementing this algorithm.
Figure 4.17 One cycle is emitted from a point source, and simulated with 7 cells per wavelength.
(a) There are a lot of tails behind the wave.
(b) The tail has been reduced when the noise removal method (in Algorithm 4.7) has been applied.

4.6 Conclusion

The Lattice Gas models have been shown to simulate the qualitative features of hydrodynamic flows. Most LG models adopt boolean values. They use less computing space and time, but results only have two levels, 0 and 1. This kind of model is used to show the distribution of particles (Appert et al., 1994; Ershov, 1994). The research reported here focuses on visualising the sound pressure in air. The results needed should have many levels so that it can be used to render a grey scale image. The simulation model discussed in this Chapter is based on the Krutar's model which is the discretised form of the wave equation. Coefficients in the Krutar's model in the iterative form, Equation 4.1 (c), have been modified, and the three calculation steps in the Krutar's model, Equation 4.1, have been combined into one in each iterative step.

It has been pointed out the LG model based on wave equation does show wave diffraction. Also several methods for solving the window edge problem have been
posed, and it seems that the Post History method is the best one among them. It possesses simple calculations and can be applied to complex simulation environments.

Boundary rules have been discussed. As Equation 4.10 does not take the impedance into account, it cannot be used for cells near a boundary. A novel boundary rule, modelling impedance change, has been proposed in Section 4.3. But when we simulate a boundary, around which impedance does not change, but speed does, the result is not correct, as reflection occurs (Figure 4.18). The reflection coefficient is zero, so it should have no reflection.

![Figure 4.18](image)

**Figure 4.18** The impedance is the same in both media. The speed of sound is 331 m/s in medium-1 and 662 m/s in medium-2. Simulated with the novel boundary rule, Equation 4.21.

Simulation for diffusion has been explored in Section 4.3.2. Though the results do not show apparent difference from that of specular reflection, it is a worthwhile experiment.

It has been noted that the LG model introduced some artifacts into the results. When simulating waves further into the far field, the first wavefront is not a real signal (Figures 4.11, 4.12 and 4.13), but is a sort of artifactor, noise. Another sort of noise occurs when the wavelength in the simulation is less than 15 cells. Wave tails behind real
signals will be seen. To avoid these sorts of noise, the wavelength should be set no less than 15 cells. This will increase the size of a simulation array, so that it will limit the application of this simulation model to a large area. A noise removal technology (Algorithm 4.7) has been provided in Section 4.5. To a certain extent, it reduces noise, and improves the quality of simulation results. However, if the wavelength is set much less than 15 cells, the amplitude of tails behind real signals is significant (Figure 4.15 (f),

Figure 4.19 One cycle of waves is emitted from a point source at the centre. Wavelength is 3.5 cells.
(a) An original result without being applied noise removal technology;
(b) noise removal threshold is set to 50 (the whole range is from 0 to 255);
(c) threshold is 80; and
(d) threshold 100.
3.5 cells per cycle). In this case, the Algorithm 4.7 cannot provide us with a satisfactory result (Figure 4.19).

Equation 4.21 can be used both cases of smooth or rough interfaces of a boundary. The different between these two cases is only a matter of boundary setting in their data structure.

The memory space for this model has been analysed in Section 4.2. It requires \(4(3mn + 8m + 6n + 16)\) bytes (See Equation 4.11). This figure results from implementing Equation 4.10. To execute Equation 4.21, we should replace the array of 'P' in Algorithm 4.1 with the data type 'node', defined in Section 4.3.3. If the type 'real' is supposed to be 4 bytes, the memory size for the type 'node' is

\[
4 + \left\lfloor \frac{(N + 4 \times (N^2 - N + 1))}{8} \right\rfloor \text{ bytes}
\]

where \(N\) is the number of media. If two media are in a simulated environment, the type 'node' needs

\[
4 + \left\lfloor \frac{(N + 4 \times (N^2 - N + 1))}{8} \right\rfloor = 4 + \left\lfloor \frac{(2 + 4 \times (2^2 - 2 + 1))}{8} \right\rfloor = 6 \text{ bytes.}
\]

The arrays, in Algorithm 4.1, 'P', 'P_new', and 'P_old' now need:

\[
6(m + 2)(n + 2) + 4(m + 2)(n + 2) + 4 \times 2(m + 2) = 10mn + 28m + 20n + 56 \text{ bytes}
\]

The space for the speed is \(4N\). Reflection and transmission coefficients require space of \(2 \times 4(N^2 - N + 1)\). Therefore, computing with this simulation model, two media in an simulated environment, needs

\[
10mn + 28m + 20n + 56 + 8N^2 - 4N + 8 = 10mn + 28m + 20n + 88 \text{ bytes}
\]

where \(m\) and \(n\) are the width and height of a simulation window.
The computing time complexity is $O(s \times m \times n)$, where $s$ is the number of simulation steps. The number of simulation steps is determined by the distance from a wave source to the simulated obstacles and by the simulation speed. The simulation speed varies with real speed in different media. As the maximum simulation speed is 0.707, the simulation speed in any medium is determined by the ratio of maximum real speed in any media to the minimum real speed. The formula for the simulation speed is

$$ss = 0.707 \frac{s_c}{s_m} \text{ cells / simulation step}$$

where

- $ss$ is the simulation step,
- $s_c$ is the speed in a current medium, and
- $s_m$ is the maximum speed.

The Lattice Gas model is suitable for simulation in the near field. The square images are simulated in the near field, while the rectangle figures include both the near and far field. For a square image, the size of a view window is 200 * 200 cells, the number of simulation steps is 200. It takes 5.5 minutes to run a simulation on a Macintosh Quadra 950. If we want to simulate an environment, with an obstacle 1 meter from a source, and iron material, the number of cells required for the distance from the source to the obstacle is 2206, according to Equation 4.23. The speed in iron is 5950 m / s, the speed in air is 331 m / s, the simulation speed in air, according to Equation 4.28, is:

$$ss = 0.707 \frac{s_c}{s_m} = 0.707 \frac{331}{5950} = 0.04 \text{ cells / simulation step}$$

The number of simulation steps for simulating wave propagation from the source to the obstacle is:
\[ s = \frac{n_c}{ss} = \frac{2206}{0.04} = 55150 \] 4.30

where \( n_c \) is the number of simulation cells.

If the size of the simulation window is 2206 * 2206 cells, the simulation time for simulating the wave propagation from the source to the obstacle is:

\[
\frac{55150 \times 2206 \times 2206}{100 \times 200 \times 200} \times 5.5 = 369028 \text{ minutes} = 6150 \text{ hours.} \] 4.31

This figure is for the time of flight from the source to the obstacle. If we want to simulate the echoes at the receiver, the time needed will be doubled.
Chapter 5
Algorithms for Calculating the Shape of a Wavefront Scattered by a Curved Obstacle for Specular Reflection

5.1 Introduction

When a sound wave meets an obstacle, it is scattered. The shape of the scattered wavefront depends on the geometry of the obstacle and the curvature of the wave. If the obstacle is flat, the scattered wavefront will be like that from the wave source but reversed, without distortion. If the obstacle is concave or convex, the scattered wavefront will be focused or spread accordingly. In Chapter 3, the mirror equation (Semat, 1966) and the law of reflection from geometric optics (Hecht, 1987) are used to model this process. In this Chapter, an algorithm is presented for calculating the shape of a wavefront scattered by an obstacle with an arbitrary geometric shape, based on the law of reflection from geometric optics.

In Section 5.2, an equation is derived for the pressure of a scattered wavefront at any point in space, from the source amplitude and the location of the centre of curvature of the scattered wavefront. In Section 5.3, an equation is derived for the location of the centre of curvature for any point on the scattered wavefront. In Section 5.4, algorithms are developed to calculate the sound pressure at all points in space using these equations. In Sections 5.5 and 5.6, these algorithms with examples are illustrated, and in Section 5.7, their comparison is given.
5.2 Wave pressure

If a wave source is a cylinder source and the cross section of obstacles are invariant in the third dimension, this case can be considered in two-dimensional space with a point source, as their mathematical expression is in a two-dimensional form.

In Chapter 3, the solution for the wave equation in two-dimensions is shown:

\[ f = F e^{-i\omega t} [J_0(kr) + iN_0(kr)] \]  
(3.3)

If this function is applied to the pressure \( P(r,t) \), then the real part of this solution is:

\[ P(r,t) = F [J_0(kr) \cos(\omega t) + N_0(kr) \sin(\omega t)] \]  
5.1

Equation 5.1 can be rewritten as

\[ P(r,t) = FA \sin(\omega t + \varphi) \]  
5.2

where

\[ A = A(kr) = \sqrt{J_0^2(kr) + N_0^2(kr)}, \]  
5.3

\[ \varphi = \varphi(kr) = \tan^{-1}\left(\frac{J_0(kr)}{N_0(kr)}\right) \]  
5.4

The ratio of the pressure \( P(r_2,t) \) at distances \( r_2 \) from the source to that \( P(r_1,t) \) at distance \( r_1 \) is

\[ \frac{P(r_2,t)}{P(r_1,t)} = \frac{A(kr_2)\sin[\omega t + \varphi(kr_2)]}{A(kr_1)\sin[\omega t + \varphi(kr_1)]} \]  
5.5
which can be written in the form

\[ P(r_2, t) = P(r_1, t) \frac{A(kr_2) \sin(\omega t + \varphi(kr_2))}{A(kr_1) \sin(\omega t + \varphi(kr_1))} \]

5.6

When the outgoing wave meets an obstacle (Figure 5.1), the scattered wave behaves like a new wave which is emitted from an imaginary source. For the reflected waves, the distances in Equation 5.6 from the wave source is now the distance from the imaginary source.

\[ R = \frac{Z'_o \cos \theta - Z_o \cos \theta'}{Z'_o \cos \theta + Z_o \cos \theta'} \]

\[ \theta \]

**Figure 5.1** Ray model of reflection.

The reflection coefficient \( R \) is a function of the angle of incidence and the angle of reflection, and the impedance of two mediums meeting at the boundary. According to Kuttruff, (Kuttruff, 1991, p. 40), the reflection factor \( R \) has the following form:
If the obstacle is steel, its characteristic impedance is $46.6 \times 10^6 \text{Ns} / \text{mA}$, while that in air is $4.29 \text{Ns} / \text{mA}$. Then the reflection factor $R$ is

$$R = \frac{46.6 \times 10^6 \cos \vartheta - 4.29 \cos \vartheta'}{46.6 \times 10^6 \cos \vartheta + 4.29 \cos \vartheta'} \approx 1$$

$$P(r_1, t) = RP(r_0, t) = P(r_0, t)$$  \hspace{1cm} 5.7

Therefore, for the rest of this Chapter, a reflection factor of 1 is assumed. The calculation of the reflection coefficient can easily be added in situation where the impedance difference is not so large. This research focuses on the developing the computational geometry of the problem.

The geometric shape of the obstacle determines the shape of the wavefront of the scattered wave. If the obstacle is convex, the scattered wave will expand. If the obstacle is concave, the scattered wave will be focused. But the phase angle in the sine term in Equation 5.4 will not be changed, that is, the phase terms in $P(r_0, t)$ and $P(r_1, t)$ should be identical. As $\varphi(kr_0)$ does not always equal $\varphi(kr_1)$, $\varphi_0$ is used to denote the difference between them, that is,

$$\varphi_0 = \varphi(kr_0) - \varphi(kr_1)$$  \hspace{1cm} 5.8

where $\varphi_0$ represents the phase difference between the initial angles of two sources.

Now we have

$$\omega t + \varphi(kr_0) = \omega t + \varphi(kr_1) + \varphi_0$$  \hspace{1cm} 5.9

We can apply Equation 5.6 to the wave reflected from the imaginary source,

$$P(r_2, t) = P(r_1, t) \frac{A(kr_2) \sin[\omega t + \varphi(kr_2) + \varphi_0]}{A(kr_1) \sin[\omega t + \varphi(kr_1) + \varphi_0]}$$  \hspace{1cm} 5.10
According to Equations 5.7 and 5.2,

\[ P(r_1, t) = P(r_0, t) = FA(kr_0)\sin(\omega t + \varphi(kr_0)) \]  \hspace{1cm} 5.11

Substituting Equation 5.11 into Equation 5.10 gives:

\[ P(r_2, t) = FA(kr_0)\sin(\omega t + \varphi(kr_0)) \frac{A(kr_2)\sin[\omega t + \varphi(kr_2) + \varphi_0]}{A(kr_1)\sin[\omega t + \varphi(kr_1) + \varphi_0]} \]  \hspace{1cm} 5.12

As \( \varphi_0 = \varphi(kr_0) - \varphi(kr_1) \), then

\[ \sin(\omega t + \varphi(kr_0)) = \sin[\omega t + \varphi(kr_1) + \varphi_0] \]  \hspace{1cm} 5.13

\[ \sin[\omega t + \varphi(kr_2) + \varphi_0] = \sin[\omega t + \varphi(kr_2) + \varphi(kr_0) - \varphi(kr_1)] \]  \hspace{1cm} 5.14

Substituting Equations 5.13 and 5.14 into Equation 5.12 gives:

\[ P(r_2, t) = F \frac{A(kr_0)A(kr_2)}{A(kr_1)} \sin[\omega t + \varphi(kr_2) + \varphi(kr_0) - \varphi(kr_1)] \]  \hspace{1cm} 5.15

### 5.3 Imaginary source

In Section 5.2, the relationship between the wave pressures at different positions was discussed. Also the pressure of the scattered wavefront from the imaginary source was calculated. In this section, how to find the location of the imaginary source is discussed.

The imaginary source is located at the centre of the curvature of the scattered wavefront. The formula for the centre of the curvature of any curve is:
\[ x_e = X - \frac{\dot{Y}(X^2 + Y^2)}{X\dot{Y} - XY} \]

\[ y_e = Y + \frac{\ddot{X}(X^2 + Y^2)}{X\ddot{Y} - XY} \]

where

\( X \) and \( Y \) are the coordinates of the receiver point on the curved wavefront, and are defined with respect to a parameter, 
\( \dot{X} \) and \( \dot{Y} \) denote the first derivative with respect to the parameter, and 
\( \ddot{X} \) and \( \ddot{Y} \) denote the second derivative with respect to the parameter.

\[ \text{Figure 5.2 Diagram of notations for various parameters.} \]

For modelling sound propagation, \( X \) and \( Y \) are functions with respect to the parameter \( \alpha \) (Figure 5.2), where \( \alpha \) is the angle of the incident ray to the x axis. In this case, \( X \) and \( Y \) have the forms,
\[ X(\alpha) = x_0 + r \cos \delta \]
\[ Y(\alpha) = y_0 + r \sin \delta \]

where

- \( x_0 \) and \( y_0 \) are the coordinates of the reflecting point, and are functions of \( \alpha \),
- \( r \) is the distance from reflecting point to the receiver point, and is a function of \( (x_0, y_0) \), and
- \( \delta \) is the angle of the reflected ray to the x axis. It too is a function of \( \alpha \).

If the shape of the obstacle is given in a general form, \( y = f(x) \), then \( X \) and \( Y \) are functions of \( x \),

\[ X(x) = x + r \cos \delta \]
\[ Y(x) = y(x) + r \sin \delta \]

where \( x \) and \( y \) are coordinates of the position of the reflecting point. In Figure 5.2, the reflecting point is located at \( (x_0, y_0) \).

In the following formulas, sometimes a pair of formulas are distinguished by the labels 'a' and 'b' as above. These formulae describe the same problem, but the first 'a' is given in a parametric form, and the second 'b' is given in a general form.

If we know the function describing the obstacle, we can calculate \( X, Y, \dot{X}, \dot{Y} \), and then using Equation 5.16 to find the centre of curvature \( (x_c, y_c) \). Thus we obtain the location of the imaginary source.

### 5.4 Wavefront algorithms

Having derived the necessary equations, the next step is to use these equations to calculate the sound pressure at all points in two-dimensional space at any instant in time.
In this section, two algorithms are formulated for generating the total complex sound field of the backscattered waves.

Consider each point in space ((X, Y) in Figure 5.2) to be a receiver point. The incident wavefront is modelled as a set of rays radiating from the source (0,0). Each ray reflects from the obstacle at a different reflecting point \((x_0,y_0)\). One or more reflected rays pass through each receiver point. The location of the receiver point \((X,Y)\) can be alternatively described by the angle \((\alpha)\) of the incident ray to the x axis and the distance \((r)\) along the reflected ray from the obstacle to the receiver point.

Two algorithms for calculating the pressure at every point in space are considered. In the \(\alpha - r\) algorithm, the location of the receiver point \((X,Y)\) is calculated from \(\alpha\) and \(r\). The algorithm (Algorithm 5.1) spans space with a nested loop, which increments \(r\) in the inner loop and \(\alpha\) in the outer loop. The pressure is calculated at each point specified by \((\alpha,r)\).

In the \(X-Y\) algorithm, the receiver point \((X,Y)\) is given, and \(\alpha\) and \(r\) values are calculated for that point. The algorithm (Algorithm 5.2) spans space with a nested loop, which increments \(Y\) in the inner loop and \(X\) in the outer loop. The pressure is calculated at each point \((X,Y)\).

### 5.4.1 \(\alpha - r\) algorithm

Define a 2D curve that describes the surface in 3D with invariant cross section in the third dimension, with a parametric function in terms of \(\alpha\):

\[
\begin{align*}
  x &= x(\alpha) \quad \text{5.18a} \\
  y &= y(\alpha)
\end{align*}
\]

or in a general form

\[
y = f(x) \quad \text{5.18b}
\]

The slope \((s)\) of the curve at a point \((x_0,y_0)\) is:
\[ s = \frac{dy}{dx} \frac{1}{d\alpha} \]

5.19a

\[ s = \left. \frac{dy}{dx} \right|_{x=x_0} \]

5.19b

The angle \( \beta \) of the normal to the curved surface (Figure 5.3) relative to the x axis at the point \((x_0, y_0)\) is:

\[ \beta = \tan^{-1}\left( \frac{dx/d\alpha}{dy/d\alpha} \right) \]

5.20a

\[ \beta = \tan^{-1}\left( \frac{dx}{dy} \right) \bigg|_{x=x_0} \]

5.20b

The angle \( \delta \) of the reflected ray to the x axis is:

\[ \delta = \pi - (\gamma + \beta) = \pi - (\alpha + 2\beta) \]

5.21a

\[ \delta = \pi - (\gamma + \beta) = \pi - \left( \tan^{-1}\left( \frac{y_0}{x_0} \right) + 2\beta \right) \]

5.21b

where \( \gamma = \alpha + \beta \) is the angle of the reflected ray to the surface normal (Figure 5.3).
Figure 5.3 Geometrical parameters used in the $\alpha - r$ algorithm.

The equation for the receiver point $(X,Y)$ on the ray reflected from the point $(x_0,y_0)$ is given in Equation 5.17. As the terms $x_0,y_0$ and $\delta$ in Equation 5.17 are all functions of $\alpha$ (in a parametric form) or $x$ (in a general form), the curvature centre $(x_c,y_c)$ can be found with Equation 5.16.

\begin{align*}
\textbf{Algorithm 5.1} \quad \alpha - r \text{ algorithm}
\end{align*}

\begin{align*}
\text{for } \alpha : = \alpha_{\text{min}} \text{ to } \alpha_{\text{max}} \text{ by } \Delta \alpha \text{ do} \\
\text{for } r : = r_{\text{min}} \text{ to } r_{\text{max}} \text{ by } \Delta r \text{ do} \\
\quad \text{Calculate } X \text{ and } Y \text{ from } \alpha \text{ and } r; \quad \{ \text{Equation 5.17}\} \\
\quad \text{Calculate Imaginary Point from } X, Y, \text{ and } \alpha; \quad \{ \text{Equation 5.16}\} \\
\quad \text{Calculate Pressure at } X, Y \text{ at time } t; \quad \{ \text{Equation 5.15}\}
\end{align*}

Equation 5.17 can be used to calculate every receiver point in a view window by varying the increment $\alpha$ and $r$, as long as the increment values are small enough. The problem is in deciding on a suitable increment value. If the value is too small, some receiver points may be calculated redundantly; if too large, some may be skipped. Thus,
it is impossible for this space spanning algorithm to visit every receiver point once and only once.

5.4.2 X-Y algorithm

The derivation of this algorithm also starts with a parametric form of the function describing the curve shape (Equation 5.18a), and uses the same definition for the angles (Figure 5.4) except the angle $\gamma_1$ of reflection and the angle $\gamma_2$ of incident are calculated in different ways.

![Figure 5.4 Geometric parameters used in the X-Y algorithm.](image)

The angle $\gamma_1$ of reflection, for a reflected ray passing through the receiver point $(X,Y)$, is

$$\gamma_1 = \tan^{-1}\left(\frac{Y-y_0}{x_0-X}\right) - \beta$$

5.22

The angle $\gamma_2$ of incidence, for an incident ray which hits the reflecting point $(x_0,y_0)$, is

$$\gamma_2 = \alpha + \beta$$

5.23
In geometric optics, the angle of incidence equals the angle of reflection,

\[ \gamma_1 - \gamma_2 = 0 \]  
\[ 5.24 \]

Substituting Equations 5.22 and 5.23 into Equation 5.24 gives,

\[ \tan^{-1}\left(\frac{Y - y_o}{x_o - X}\right) - \alpha - 2\beta = 0 \]  
\[ 5.25 \]

If we use \( g(\alpha) \) to denote the difference between the angle \( \gamma_2 \) of incident and the angle \( \gamma_1 \) of reflection, Equation 5.25 can be rewritten as:

\[ g(\alpha) = \tan^{-1}\left(\frac{Y - y_o}{X - x_o}\right) + \alpha + 2\beta = 0 \]  
\[ 5.26a \]

\[ g(x_o) = \tan^{-1}\left(\frac{Y - y_o}{X - x_o}\right) + \tan^{-1}\left(\frac{y_o}{x_o}\right) + 2\beta = 0 \]  
\[ 5.26b \]

The roots \( \alpha \) or \( x_o \) of these equations describe the positions of reflecting points, which are at the obstacle surface with the parameter \( \alpha \) or \( x_o \), where the angle of incidence is equal to angles of reflection, or in other words, when waves hit the obstacle at the position \((x(\alpha), y(\alpha))\) or \((x_c, f(x_c))\), it will be reflected to pass through the receiver point \((X, Y)\).

The first derivative may be used when finding the roots of Equation 5.26,

\[ \dot{g} = \frac{\dot{x}(Y - y) - \dot{y}(X - x)}{(X - x)^2 + (Y - y)^2} + 1 + 2\frac{\dot{x}Y - \dot{y}x}{x^2 + y^2} \]  
\[ 5.27 \]

Equation 5.26 is transcendental with respect to \( \alpha \) or \( x_o \). Its derivatives may be more complex than it is. A numerical method is required to iteratively solve this equation. 
The numerical method will be chosen depending on the final form of Equation 5.26 which is a function of the shape of the surface.

\begin{algorithm}
\begin{verbatim}
for \( X := X_{\text{min}} \) to \( X_{\text{max}} \) do \\
\quad for \( Y := Y_{\text{min}} \) to \( Y_{\text{max}} \) do \\
\quad\quad Calculate \( \alpha \) or \( x_0 \) from \( X \), and \( Y \); \hspace{1cm} \{ \text{Equation 5.26} \}
\quad\quad Calculate Imaginary Point from \( X \), \( Y \), and \( \alpha \); \hspace{1cm} \{ \text{Equation 5.16} \}
\quad\quad Calculate Pressure at \( X,Y \) at time \( t \); \hspace{1cm} \{ \text{Equation 5.15} \}
\end{verbatim}
\end{algorithm}

Algorithm 5.2 \( X-Y \) algorithm

5.5 Simple examples

In this section, two examples of scattering from obstacles are examined, whose shapes are defined with the general form \( x = f(y) \). The \( X-Y \) algorithm is used to calculate these space spanning pictures. As the obstacle in the first example (Figure 5.5) is flat, either Equation 5.26 or the mirror equation can be used to calculate \( y \) in the first step of the body of the \( X-Y \) algorithm. Equation 5.26 results in an equation with a single root:

\[
g(y) = \tan^{-1}\left(\frac{Y-y}{X-(ky+c)}\right) + \tan^{-1}\left(\frac{y}{ky+c}\right) + 2\tan^{-1}(k) = 0 \hspace{1cm} 5.28
\]

The shape of the obstacle in the second example (Figure 5.6) is defined by the parabolic function \( x = ky^2 + c \). The final form of Equation 5.26 for this example is:

\[
g(y) = \tan^{-1}\left(\frac{Y-y}{X-(ky^2+c)}\right) + \tan^{-1}\left(\frac{y}{ky^2+c}\right) + 2\tan^{-1}(2ky) = 0 \hspace{1cm} 5.29
\]
As this obstacle is concave in shape, the reflected wavefront spreads. This means that only one ray can reach at each receiver point or we can say this equation has only one root. The Bisection method (Faires and Burden, 1993) can be used to find the root. This method is used to determine a solution to \( f(x) = 0 \) on an interval \([a, b]\), provided that \( f \) is continuous on the interval and that \( f(a) \) and \( f(b) \) are of opposite sign.

<table>
<thead>
<tr>
<th>Select appropriate initial values for ( a ) and ( b );</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeat ( y := (a + b) / 2 );</td>
</tr>
<tr>
<td>if ( g(y) \cdot g(a) &gt; 0 ) then</td>
</tr>
<tr>
<td>( a := y );</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>( b := y );</td>
</tr>
<tr>
<td>Until ( g(y) = 0 )</td>
</tr>
</tbody>
</table>

Bisection method

The results of the X-Y algorithm for the pressure at all points in space at time \( t \) are stored in an array. This data is visualised with grey scale rendering in Figures 5.5 and 5.6 for these two examples.

![Figure 5.5 Wave scattered from the obstacle: \( x = ky + c \)](image_url)
These pictures show only the reflected wave patterns, with the black dot on the left indicating the location of the source.

### 5.6 Complicated examples

In this section, reflection from two complex obstacles is demonstrated, where the reflected sound field of a circular wave includes interference between reflections from different points on the obstacles. The $X-Y$ algorithm is used. The first step is to calculate the angle $\alpha$ in the first example, and $y$ in the second example for the rays that reflect through the receiver point $(X,Y)$.

In the first example, the obstacle is defined in a parametric form:

$$
\begin{align*}
  x &= 500 - 300 \frac{a^3}{\alpha^2 + a^2} \\
  y &= x \tan \alpha
\end{align*}
$$

(a = 0.1) \quad 5.30

The final form of the equation 5.26a is:
As the equation is complicated, the graph is used as a visualisation tool to assist us to look at it. A suitable numerical method is chosen for an approximate solution. Consider the graph of Equation 5.31 as shown in Figure 5.7. The \( \alpha \) values at points where the graph crosses the \( \alpha \)-axis are solutions of the equation \( g(\alpha) = 0 \). Equation 5.31 has different sets of solutions depending on the values of \( X \) and \( Y \). This graph shows that when \( X = 350 \), the equation has one root except when \( Y \) is around 161, where it has three roots. Physically, this means that at some receiver points (for example, \( X = 350, \ Y = 161 \)), the scattered wave comes from three reflecting points, and at others from one reflecting point. That is the number of reflecting points that generate a ray through a receiver point is equal to the number of roots.
Although the Bisection method will work for the case when more than one root is contained, it can find only one of them. If an equation has more than one root, the range is divided into subranges, each of which has no more than one root before the Bisection method is used. The subranges are separated by stationary points: local maximum or minimum where the derivatives are zero (Figure 5.8). But the derivative of Equation 5.31 (Figure 5.8) is complicated, and thus it is harder to find its monotonic region, which is required by the simple numerical methods, such as the Bisection method or Newton's method, for an approximate solution. If we can find its monotonic regions among derivative equations of any high order, we can find the roots for that derivative equation with the Bisection method. Consequently, we can find the roots for its one order lower derivative equations. Finally we can find the solution for the zero order
derivative equation, that is, the equation itself. As the higher order derivatives become more and more complicated, it is difficult to determine their monotonic regions.

A trivial approach for finding approximation solutions to an equation \( f(x) = 0 \) is to calculate the value of the equation \( f(x_1), f(x_2), ..., f(x_n) \) at points \( x_1, x_2, ..., x_n \) by a small interval \( \delta x \). The roots are found in the regions \([x_i, x_{i+1}]\) whose values at ends \( f(x_i) \) and \( f(x_{i+1}) \) are of opposite sign. The exact location of the roots can then be found by applying the Bisection method within these regions. The problem is how to choose the interval size \( \delta x \). If we know the number of roots, we can start with a bigger interval, and subdivide it repeatedly until we find the all roots. If the number of roots are unknown or flexible, we have to exhaust the whole region.

In this case for the solution to the Equation 5.31, with the help of Figure 5.11, the number of rays reflected from the obstacle is one or three. That means the number of roots for the Equation 5.31 is one or three. In most cases only one root exists. If we use the above trivial approach to find the roots, usually we will travel the whole range in the smallest intervals. As the derivative of Equation 5.31 is complex, we cannot find how many roots exist by analysing the equation. With the help of the graph (Figure 5.7), we find that there appears to be three stationary points in the argument range \( \alpha > 0 \). As the shape of the obstacle is symmetric with the y axis, the reflected wavefront will be symmetric as long as the wave source sits on the obstacle's symmetric axis. The rays through the receiver points whose y coordinates are greater than zero are reflected by the reflecting points whose parameter \( \alpha \) values are greater than zero. For the rays through the minus y value receiver points, we can calculate the reflecting point with the absolute value of y first, then we can take the opposite sign of the solution \( \alpha \) value. As the number of roots of Equation 5.31 is in most cases one and sometimes three, and the number of its stationary points, the roots of its derivative equation, is in most cases two and sometimes zero, we decide to find its derivative equation by using the trivial method.
I begin my search in a bigger interval. If the number of the stationary points found is less than two, the process is repeated with smaller intervals ($\delta \alpha$) until a predefined limit is reached, where it is assumed that no further stationary points exist. Once the stationary points are found, the curve can be divided into regions, with no more than one root per region. The Bisection method can then be used to find the root in each region.

$\alpha$

\[ \dot{g}(\alpha) \]

\[ \dot{g}(\alpha_i) \]

\[ \alpha_i + \delta \alpha \]

\[ \dot{g}(\alpha_i + \delta \alpha) \]

Figure 5.9 Identifying the location of a stationary point by the zero crossing of the derivative, i.e., it is in the region $(\alpha_i, \alpha_i + \delta \alpha)$ while $\dot{g}(\alpha_i) \cdot \dot{g}(\alpha_i + \delta \alpha) < 0$. 

Figure 5.8 Curves of $g(\alpha)$ and $\dot{g}(\alpha)$, when $X = 350$, $Y = 230$. 

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure5_8.png}
\caption{Curves of $g(\alpha)$ and $\dot{g}(\alpha)$, when $X = 350$, $Y = 230$.}
\end{figure}
\[ dg[0] := dG(a); \]
\[ dg[\text{maxdivision}] := dG(b); \]

\textbf{repeat} \hspace{1cm} \textbf{repeat}

\begin{align*}
  n &:= 2; \\
i &:= 0; \\
dga &:= dg[0]; \\
\end{align*}

\textbf{repeat}

\begin{align*}
a &:= (b - a) \times (i + 1) / n + a; \\
dg[(i + 1) \times \text{maxdivision} / n] &:= dG(a); \\
\text{if } (|dG(a)| < \varepsilon) \text{ then} & \\
  \text{root[root_num ++] := } a; \\
\text{else} \begin{align*}
  \text{if } (dga \times dG(a) < 0.) \text{ then} & \\
  \text{root[root_num] := } \text{Bisection}(a - (b - a) / n, a); \\
  \text{root_num := root_num + 1;} \\
\end{align*} \\
\text{end}; \\
dga &:= dg[(i + 2) \times \text{maxdivision} / n]; \\
\text{if } (dga \times dG(a) < 0.) \text{ then} & \\
  \text{root[root_num ++] := } \text{Bisection}(a, a + (b - a) / n); \\
  \text{root_num := root_num + 1;} \\
\text{end}; \\
i &:= i + 2; \\
\text{until } ((\text{root_num} >= 2) \text{ or } (i >= n)); \\
\text{until}(\text{root_num} >= 2) \text{ or } (n >= \text{maxdivision}); \\
\text{if } ((\text{root_num} < 2) \text{ and } (dg[\text{maxdivision} - 1] \times dg[\text{maxdivision}] < 0.)) \text{ then}& \\
  \text{root[root_num ++] := } \text{Bisection}(b - (b - a) / \text{maxdivision}, b); \\
  \text{root_num := root_num + 1;} \\
\text{end};
\end{align*}

\textbf{Algorithm 5.3} Finding the stationary points of g(a).
The second step is to calculate the location of the imaginary source for each reflected ray by substituting the parametric equations for the curve and their first and second derivatives into Equation 5.16. They are listed in Appendix C.

The third step is to calculate the pressure amplitude at every receiver point using Equation 5.15. The resulting data array can then be visualised as a colour intensity image as shown in Figure 5.10.

![Figure 5.10](image)

**Figure 5.10** Wavefront scattered from the obstacle defined in Equation 5.30. The source is indicated by a dot on the left. The colour bar indicates mapping from pressure amplitude to colour.

Figure 5.10 shows both the decrease in wave pressure due to spreading by a convex surface and the interference due to focusing by a concave surface. The sharp edges in the image are a result of this focusing. To investigate this phenomena further a simplified version of the $\alpha - r$ algorithm was developed to show the direction of the reflected rays. Figure 5.11 shows that the waves in the interference region are formed by the superposition of three wavefronts.
for $\alpha := \alpha_{\text{min}}$ to $\alpha_{\text{max}}$ by $\Delta \alpha$ do

Calculate reflecting point;

Calculate $X$ and $Y$ from $\alpha$ and $r$; /* Equation 5.12 */

Draw a line from reflecting point to $X, Y$;

end

Algorithm 5.3 $\alpha$ algorithm

![Image showing scattered rays from an obstacle](image_url)

Figure 5.11 Scattered rays from the obstacle defined in Equation 5.30, calculated with $\alpha$ algorithm.

The second example (Figure 5.12) shows the power of the $X$-$Y$ algorithm to model reflection from complex shapes including the interference patterns set up by the
superposition of multiple reflections. The calculation method is similar to that of the first example, except a general equation (Equation 5.26b) was used rather than a parametric equation, as the shape of an obstacle is given in a general form: \( x = ky^3 + c \).

\[ x = ky^3 + c \]

**Figure 5.12** Wave scattered from the obstacle: \( x = ky^3 + c \)

### 5.7 Comparison of the X-Y and \( \alpha-r \) algorithm

Comparing the steps in the two algorithms, it is found that the main calculation of the \( \alpha-r \) algorithm is Equation 5.17 and that of the X-Y algorithm is Equation 5.26. Equation 5.26 is much more complicated than Equation 5.17. But for the modelling of space, the \( \alpha-r \) algorithm executes the nested loops many more times than the X-Y algorithm in order to traverse every receiver point in a view window. To guarantee that any receiver point is covered, the loop increment in the \( \alpha-r \) algorithm may have to be very small and depends on the shape of the obstacle.
In addition, the $\alpha - r$ algorithm suffers from quantisation errors. As the results from calculating Equation 5.17, ($X$ and $Y$), are not always integers, they have to be rounded to integers to become the coordinates on a computer screen (Figure 5.13). However, the $\alpha - r$ algorithm is suitable for modelling a beam where wave pressure is only calculated in the region of space covered by the beam.

**Figure 5.13** The redundant and skipped calculation in $\alpha - r$ algorithm. The quantisation error is showed by $\bullet$ the position calculated and $\circ$ the position rounded to integers.

The results of the $\alpha - r$ and the $X - Y$ algorithms can be rendered as intensity variations for visualisation (Figure 5.5, 5.6, 5.10 and 5.12). The results of the $\alpha$ algorithm can be rendered as a ray diagram (Figure 5.11). Thus, the algorithm you choose for a particular problem is determined by whether you want to model a beam or all of space and the type of rendering you require for visualisation.

All of the previous examples were solved using the $X - Y$ algorithm because it is more efficient. The two complicated examples have been solved using the $\alpha - r$ algorithm with virtually identical results.
5.8 Conclusion

The wave pressure of a scattered wavefront varies with phase and with wavefront curvature in both time and space. The phase of the wave at a receiver point is a function of time and distance from a source. The shape of the scattered wavefront is determined by the shape of the incident wave and the curvature of the reflecting obstacle. Although other parameters can effect wave pressure, they are assumed constant during wave propagation.

In this Chapter, algorithms for calculating the shape of a wave scattered by a curved obstacle have been presented. These algorithms calculate the wave pressure at all receiver points in a region of space surrounding the obstacle by determining the pressure at imaginary sources located at the centre of the curvature of the reflected wavefronts. Solutions to scattering off four obstacles have been presented to illustrate the details and the general nature of these algorithms.

First, an equation was derived to calculate the wave pressure at a receiver point at known distances from a wave source and an imaginary source (Equation 5.15). To use this formula, an equation was derived for the location of the imaginary source for the reflected wave that passes through the receiver point (Equation 5.17). These equations are used in the $\alpha-r$ algorithm and the $X-Y$ algorithm. An alternate equation (Equation 5.26) was derived for the $X-Y$ algorithm. Both algorithms use nested loops to span space.

The nested loops in the two algorithms, (Section 5.4), use different increment variables to span space. As a consequence, the $\alpha-r$ algorithm is suitable for modelling beams, but has quantisation errors. In contrast, the $X-Y$ algorithm is a better space spanning algorithm. Both algorithms are based on the assumption that the shape of the obstacle can be described as a continuous curve.
Chapter 6
Algorithms for Calculating the Geometry Of Multi-path Reflection

6.1 Introduction

The air pressure of sound waves at a certain point in space is the superposition of the pressure of all waves from different directions that meet at this point at any instant in time. To visualise sound waves scattered in space, we should model the behaviour of waves from different directions and from different paths. In Chapter 3, wavefronts scattered from plane and arc shaped edges were discussed. In Chapter 5, the wavefront reflected from an arbitrary curved edge was discussed. These results are obtained under the assumption that no multi-path reflection occurs. That is, as sound waves travel from a transmitter to a receiver, the waves are not reflected more than once from objects. In this Chapter, the multi-path reflection will be discussed.

A well known technique in Computer Graphics is the ray tracing. With the ray tracing technique, rays are traced in a backward direction as they bounce off objects until they reach a viewer. This approach can only be applied when the initial direction of the rays is known or can be determined. When modelling ultrasound, the transmitted waves from a disc transducer have a curved wavefront and a conical curve of audition and the waves arriving at a point in space can be from any direction.

6.2 Algorithm of multi-path reflection from plane surfaces

The algorithm discussed in this section deals with multi-path reflection where a ray may reflect from many surfaces but only reflects from an individual surface once, that is, no
bounced reflection. The algorithm developed in next section can handle more complex cases of reflection paths that contact an individual surface more than once.

Consider an environment where all objects have plane surfaces. A sound transmitter and a receiver are located somewhere in this environment. The receiver may receive waves directly from the transmitter, or after one reflection, or after multi-path reflections from the surfaces. The total instantaneous amplitude at the receiver is the superposition of all arriving waves. These waves vary in phase and amplitude due to their different paths. The algorithm developed in this section calculates the position of the points where the waves have reflected from the surfaces.

Let S denote the position of the source, the transmitter, R the receiver, and L1, L2 and L3 the reflecting surfaces.

Figure 6.1 An environment with three objects, a source S and a receiver R.

6.2.1 Wave path directly from a wave source to a receiver

There is one path from the source S directly to the receiver R.
6.2.2 Wave paths for single-path reflection

In this section, the paths along which waves are reflected from only one surface are discussed. There are \( n \) paths at most, where \( n \) is the number of surfaces in the environment. In the current example, \( n \) is three.

Waves arriving at the receiver R may be reflected from the surface \( L_2 \), that is as if they were emitted from an imaginary source \( S'_2 \). The imaginary source is on the opposite side of the surface \( L_2 \), from the source S. It is a mirror point of the source S. \( S'_2 \) in Figure 6.3 shows the location of the imaginary point relative to the surface \( L_2 \). The reflecting point \( P_2 \) is found by the intersection of the surface \( L_2 \) and the line connecting the imaginary point \( S'_2 \) and the receiver R. The ray path from the source S to the receiver R reflecting from \( P_2 \) is \( S-P_2-R \) (Figure 6.3).
Now let's prove that the reflecting point $P_2$ is the intersection point of the surface $L_2$ and the line $RS'$. 

![Diagram](image)

**Figure 6.4** The reflecting point is the intersection point of the surface edge and the line connecting $S$ and $S'$. 

Let $P_2$ be the reflecting point on the surface $L_2$, $CP_2 \perp L_2$, and $S_2$ is the mirror point of $S$ about $L_2$, $B$ is on $L_2$ and $BS \perp L_2$, and $|S'B| = |SB|$, $S' \perp L_2$. According to the Law of Reflection, the angle of incidence equals the angle of reflection, that is,

\[ \angle SP_2C = \angle RP_2C. \]  

\[ \therefore \triangle SBP_2 = \triangle S'BP_2 \quad \therefore \angle SP_2B = \angle S'P_2B \]  

\[ \therefore CP_2 \perp L_2 \text{ and Equation 6.1} \quad \therefore \angle SP_2B = \angle RP_2A \]  

\[ \therefore \angle RP_2A = \angle S'P_2B \]  

Then $R$, $P_2$, and $S'$ are on a line, and $P_2$ is the crossing point of $AB$ and $RS'$ (Figure 6.4). 

We can use same approach to find other reflecting points on $L_1$ and $L_3$, if they exist. First we find the mirror point $S'$ of $L_1$, then we connect the receiver point $R$ and the mirror point $S'$, If we find that the intersection point is not on the surface $L_1$ (Figure...
6.5), then there is no ray path which starts from the source $S$ and reflects from the surface $L_1$, and arrives at the receiver point $R$.

![Diagram](image)

**Figure 6.5** The ray path $S$-$L_1$-$R$ does not exist.

Similarly, after we have found the imaginary point $S_3$ of $S$ about $L_2$, the intersection point is beyond the surface $L_3$ (Figure 6.6). There is no ray directly reflected from the surface $L_3$ to the receiver $R$.

Note, this result conforms with the arc model of ultrasonic sensing (McKerrow, 1993). A property of the arc model is that the range measured to a surface within the cone of audition is the orthogonal distance to that surface.
6.2.3 Wave paths for multi-path reflection

If \( n \) surfaces exist in an environment, the number of possible paths \( P_p(n) \), including multi-path and single-path reflection, from a source to a receiver is,

\[
P_p(n) = \sum_{i=0}^{n} A_i^n.
\]

\( A_i^n \) in Equation 6.5 denotes the number of ray paths, each of which reflects from \( i \) of the \( n \) surfaces in a certain order and \( A_i^n = \frac{n!}{(n-i)!} \). The path is considered as being different if the \( i \) surfaces reflect rays in different orders in a ray path. For example, the path S-L_1-L_2-R is different from the path S-L_2-L_1-R.

In current example, \( n = 3 \), so

\[
P_p(3) = \sum_{i=0}^{3} A_i^3 = A_0^3 + A_1^3 + A_2^3 + A_3^3
\]

\[
= \frac{3!}{3!} + \frac{3!}{2!} + \frac{3!}{1!} + \frac{3!}{0!}
\]

\[
= \frac{6}{6} + \frac{6}{2} + \frac{6}{1} + \frac{6}{1} = 16
\]
These 16 possible paths are:

S-R,
L₃-L₂-L₁-R.

When \(i = 0\), \(A'_n\) means the number of paths in which rays are reflected from 0 surfaces, from the source to the receiver directly (Figure 6.2). When \(i = 1\), the rays are reflected from one surface between the transmitter and the receiver. Figures 6.3, 6.5 and 6.6 show these possible paths. But among them, only one path, S-L₂-R, really exists (Figure 6.3) in this example.

Let us consider the path S-L₁-L₂-R (Figure 6.7). The ray from \(L_1\) to \(L_2\) is as if it were emitted from the imaginary source \(S'_1\). In general, the vector of the mirror point \(\tilde{S}'\) (Figure 6.8) for source \(\tilde{S}\) is

\[
\tilde{S}' = \tilde{S} - 2d\tilde{N}
\]

where

\(\tilde{S}'\) is the vector of the imaginary source,
\(\tilde{S}\) is the vector of the source \(S\),
\(d\) is the distance from the source point \(S\) to the surface, and
\(\tilde{N}\) is a unit vector of the normal direction to a surface.
Figure 6.7 The ray path S-L\(_1\)-L\(_2\)-R.

Figure 6.8 Vector expression of the mirror point S'.

**Procedure** MirrorPoint(source point, obstacle) : point

```
Var MPoint : point;

begin
    N := unit normal of obstacle;
    d := distance from source point to obstacle;
    MPoint := source point - 2*d*N;
    return MPoint;
end MirrorPoint.
```

**Algorithm 6.1** Mirror point calculation.
The mirror point $S'_1$ (Figure 6.9) is easy to find with Equation 6.6.

![Figure 6.9 Mirror point $S'_1$ of $S$ with respect to $L_1$.](image)

The ray from $L_2$ to $R$ is as if it were emitted from the imaginary source $S'_2$ (Figure 6.7). Now $S'_2$ is the mirror point of $S'_1$, rather than $S$, about the surface $L_2$ (Figure 6.10).

![Figure 6.10 Mirror point $S'_2$ of $S'_1$ about $L_2$.](image)
Finally we can find the reflecting point on $L_2$, which the ray will be last reflected from. It is the intersection point of surface $L_2$ and the line connecting the receiver $R$ and the imaginary point $S'_2$ (Figure 6.11).

![Figure 6.11 Mirror point $S'_2$ of $S'$ about $L_2$.](image)

The ray hitting $L_2$ is emitted from $S'$. Connecting $P_2$ and $S'$, the intersection point on $L_1$ is the reflecting point $P_1$. So, we find the whole path from the transmitter $S$, reflected from $P_1$ and $P_2$, and arriving at the receiver $R$ (Figure 6.7). In Appendix D, the process of trialing paths reflected from different surfaces is illustrated.

Among $P_p(n)$ possible paths, some of them are real paths along which waves travel from a source to a receiver. To examine whether a possible path $\{S-L_{i1}-L_{i2}-...-L_{ij}-R\}$, where $L_{ij}$ denotes one of $n$ surfaces, $L_{ij} \in \{L_1, L_2, ..., L_n\}$, is a real path, the following algorithm is used:
for a path $S - L_{i1} - L_{i2} - ... - L_{ij} - R$

$S'_{i0} := S$;

for $j' := 1$ to $j$ do

$N_{ij'}(x, y) := \text{unit normal direction of the surface } L_{ij'}$

$S'_{ij'} := S'_{ij'-1} - 2dN_{ij'}(x, y)$;

end

$S'_{i+j+1} := R$;

repeat

Connecting $S'_{i+j+1}$ and $S'_{ij'}$,

find intersection point with $L_{ij'}$.

$j' := j' - 1$;

until (intersection point doesn't exist or $j' = -1$)

(if $j'$ equals -1, this path exists)

Algorithm 6.2 Examining whether a path exists.

6.3 Algorithm for wavefront from multi-path reflection from curved surfaces

McKerrow (1989) developed a reflection model to trace a beam wavefront from a transducer or reflected from flat surfaces. In this section, an algorithm is developed for the wavefront scattered from curved surfaces after multi-path reflection. This algorithm is an extension of the Ray Tracing algorithm. Ray Tracing traces a ray from the viewer to light source, but paths of acoustic waves are not narrow ray paths, they scatter from the transmitter right from the beginning. The acoustic waves are emitted outward in a range of directions, depending on the beam angle, and they can arrive at many surfaces, and are reflected from multiple surfaces. To trace waves from the transmitter, the following algorithm gives a solution:
for surface := 1 to n do
  check whether the wavefront impinges on the surface
  if yes then
    surface_list := surface
    to find a new source { use mirror equation}
    wave tracing()
  end
end

Algorithm 6.3 Wave tracing.

To check whether waves can arrive at a surface, we can check whether the following angle ranges intersect. The angle range of a surface is defined as the range between the vector angles of two end vectors, $V_1$ and $V_2$ from the source to the surface segment (Figure 6.12). An ultrasonic transducer produces a beam. The angle range of the wave beam is defined as the beam angle for the wave source. If these two angle ranges intersect then the beam will reflect from the object.

Figure 6.12 Waves are emitted from the source in a range of directions, forming a angle range of wave beam. The angle range of a surface is defined as the range between vectors $V_1$ and $V_2$, originating at the wave source.
For the example in Figure 6.12, the angle range of the wave beam intersects with the angle range of surface L₁, but does not intersect with the angle range of surfaces L₂, L₃, and L₄. This means the waves from the source do not arrive at the surfaces L₂, L₃ and L₄ directly. Then we can trace the waves continuously. According to the Section 3.2.2, when waves are reflected from the surface L₁, they can be considered as waves emitted from an imaginary source, the location of the imaginary source can be obtained by the mirror equation (Equation 3.11). In the recursive algorithm (Algorithm 6.4), the imaginary source is considered as the wave source, and the angle range of the wave beam is considered as the intersection region, originated from the new wave source (Figure 6.13).

![Figure 6.13](image_url)

**Figure 6.13** The intersection region is formed by the angle region of wave beam intersecting with the angle region of L₁. Points P₁ and P₂, on L₁, are the end points of L₁ of the intersection region on surface L₁.

The mirror equation can be used to calculate the location of the imaginary point. In the next step of the recursion, the wave source is assigned by the imaginary point and the angle range of the wave beam is the region, limited by the points P₁ and P₂, originated at the new wave source. The angle regions of other surfaces, L₂, L₃ and L₄, also originate at the new wave source.
The data structure for the Surface_List is defined as:

```plaintext
type Surface_List = record
    n: 0..max_surface;
    other_surface: point to Surface_List;
    next_level: point to Surface_List;
end;
```

After waves are scattered from a surface, they may hit other surfaces. These surfaces are recorded in the 'next_level' field in the 'surface_list' structure (above). Surfaces hit by waves scattered from an identical surface are recorded in the 'other_surface' field. For example, waves started from a transmitter, hit surfaces L₁ and L₂; the waves scattered from L₁ hit L₃ and L₄; and waves scattered from L₂ hit L₅. The data structure will be as shown in Figure 6.15.
Figure 6.15 Data structure for recording the sequence of reflecting surfaces. In this example, the waves paths are transmitter-L1-L3, transmitter-L1-L4 and transmitter-L2-L5.

The wave tracing algorithm is shown as follows in Algorithm 6.4.

6.4 Conclusion

In this Chapter, multiple paths of multi-path reflection from a source to a receiver are discussed. For an environment with n plane surfaces, the possible paths, without bounced reflection, is $P_p(n) = \sum_{i=0}^{n} A_i^i$, where bounced reflection is defined as reflecting more than once from the same surface. For example, the path S-L1-L2-L3-L1-R is a path including bounced reflection as the waves are reflected twice from the surface L1. Algorithm 6.2 provides a method to check whether a possible path is a real path. The kernel of this algorithm is to find reflecting points on surfaces, according to the theory of ray tracing, then to find imaginary points for the sake of obtaining detailed wave structure. Algorithm 6.4 provides a solution for the wave paths in an environment where
Procedure wave_tracing (Var surface_list : Surface_List);
begin
    If surface_list = Nil then
        create a surface_list, and initialise it;
        the wave source := the transmitter;
        the angle region of wave beam := the range of transmitter;
    else begin
        current surface := surface_list↑.n;
        calculate the imaginary point of current surface with previous wave source;
        the wave source := the imaginary point;
        calculate the angle region of wave beam according to the intersecting points, P1 and P2;
    end

    this_surface_list := Nil;

    for surface := 1 to maxsurface do
        calculate the range angle of surface;
        calculate the intersection region;
        if the intersection region is not empty then
            if this_surface_list = Nil then
                create this_surface_list;
                surface_list↑.next_level := this_surface_list;
            else begin
                create other_surface_list;
                this_surface_list↑.other_surface := other_surface_list;
                this_surface_list := other_surface_list;
            end;

            this_surface_list↑.n := surface;
            this_surface_list↑.other_surface := this_surface_list↑.next_level := Nil;
            wave_tracing (this_surface_list);
        end { if }
    end { for }
end wave_tracing

Algorithm 6.4 Recursive procedure of wave tracing.
all objects have curved surfaces. The algorithm of wave tracing is derived from the ray tracing algorithm, which considers a light path as a ray. The acoustic waves from a transmitter spread in a certain angle, and expand or contract due to the surfaces that they reflect from.

Algorithm 6.2 can only be applied in an environment where there is no bounced reflection. Although Algorithm 6.4 can be applied in an environment where objects have plane surfaces and bounced reflection is possible, a more efficient algorithm can be developed for this problem.

The line drawn from a wave source to a curvature centre of a surface is called a principal axis. The mirror equation can only be applied when rays are emitted at small angles from the principle axis. If the rays are emitted at large angles from the principle axis, the reflected rays will not converge at a point. In this case, the mirror equation cannot provide an accurate solution.
Chapter 7
Application to Robotics

In robotics, we wish to visualise the reflections of common geometric features when insonified by a Polaroid ultrasonic transmitter. In this Chapter, it is indicated that how this work can be applied to robotics. The development of specific application for visualisation in robotics is beyond the scope of this thesis.

7.1 Beam pattern of a Polaroid sensor

In Figure 7.1, the output of a Polaroid sensor is visualised, showing the beam both in the near and far fields. In the far field, the beam spreads to give a conical field of audition. The lower intensity side lobes can also be seen.

Figures 7.2 to 7.3 show the results of typical sensing situations that occur in mobile robotics. Figure 7.2 shows 5 cycles reflecting from and diffracting around a flat surface. Figure 7.3 shows reflection and diffraction from a sharp edge and Figure 7.4 from a rounded edge. The intensity of the waves reflected toward the transmitter is higher in Figure 7.4. Figure 7.5 shows a time sequence of images of waves reflecting from a corner. The multi-path reflection is obvious in this figure.

Figures 7.1 to 7.5 were all produced with the Lattice Gas model with 20 cells per wavelength. They all use a simulation of the Polaroid transducer as a source (line on left side). The size of the simulation arrays for Figure 7.1 is 900 cells wide and 600 cells high. These arrays use about 6 Megabytes of memory. In Figures 7.1 to 7.5, only every 3rd cell is used in the rendering in order to reduce the size of the image.
**Figure 7.1** Visualisation of a Polaroid sensor showing beam and side lobes. The surface of the sensor is the line on the left.

**Figure 7.2** Reflection of beam from a surface.

**Figure 7.3** Reflection of beam from a corner.
7.1.1 Validation of line source visualisation

Figures 7.6 - 7.11 were produced to analyse the simulation results for a line source transducer in terms of beam pattern. Figure 7.6 visualises wave fronts produced by a line
source which is located at the bottom left with a length of 5.2 wavelengths. As the wave fronts are symmetric about the horizontal axis at the centre of the transducer, we can set the simulation window to half of the area in front of the transducer, so that we can reduce simulation time. The simulation was stopped before the wave fronts arrived at the edges of the window, to avoid the noise from reflection of the window edges. The wavelength is 21 cells, and simulation is 600 steps.

![Figure 7.6](image)

Figure 7.6 Half wave front from a line source transducer with length of 5.2 wavelengths, simulated with LG model.

The reader can observe that the shape of the simulated wave front consists of a straight line segment and an arc segment. The wave front in front of the transducer is straight, while that at the side is an arc, centred at the end point of the line source, as showed in Figure 7.7.
The shape of a wave front from a line source transducer consists of two arcs, centred at the end points of the line source, and a straight line parallel to the line source.

This assumption is verified in Figure 7.8, by drawing curves on Figure 7.6. The result shows that the simulated wave front and drawn curves match well. In addition to
this, the exact position of the simulated wave front has been identified from the pressure values and compared to the curves, with good results.

After finding the geometric shape of the wave front, we can obtain sample points in the simulation array easily, and analyse the simulation results digitally, instead of viewing a grey scale picture.

First, a beam pattern is drawn (Figure 7.9) with the following formula (Kinsler etc. 1982, p.173):

\[
b(\theta) = 20 \log \left( \frac{\sin(0.5 \ast kL \sin(\theta))}{0.5 \ast kL \sin(\theta)} \right)
\]

where \( b(\theta) \) is a function for beam pattern,

\( \theta \) is the angle with axis at centre of the line source,

\( k \) is wave number, \( k = \frac{2\pi}{\lambda} \), where \( \lambda \) is wavelength, and

\( L \) is length of the line source.
Second, the pressure values on two wave fronts at different distances from the source are plotted for comparison. Figure 7.10 shows the simulated beam pattern at 16 wavelengths from the source, and Figure 7.11 shows the simulated beam pattern at 12 wavelengths from the source.
Figure 7.10 Beam pattern plotted from sample values in a simulation array. The wave front is at 16 wavelengths from a line source with $kL = 10.4\pi$. 
We notice that the LG model can simulate the directional features and beam patterns of an acoustic wave from a line source. Side lobes and minima are easy to observe. Also, by comparison of Figures 7.10 and 7.11 to Figure 7.9, we see that the side lobes occur at the same angles, although the minima are not as low. The discrete simulation model loses some detailed information unavoidably due to the limited number of the points per wavelength.
7.2 One transmitter and two receivers

An environment with one transmitter and two receivers has been simulated in both the near field and the far field. The Lattice Gas model was chosen for simulations in the near field, and the Continuous model for simulations in the far field.

7.2.1 Simulation in the near field

In this section, the echo signals from an obstacle with a flat surface in the near field are discussed. These echo signals are been displayed in Figures 7.12 (a) and (b). The time gap between when the two receivers receive the first echo varies with the surface position and the slope of the simulated obstacle. The obstacle is located at 70 mm from the transmitter. The receivers are 17 mm from the transmitter separately. The simulated waves travel with a speed of 0.7 mm per simulation step. The echo signals received in the two receivers are drawn at the lower part of the figures. The vertical axes are the amplitude of the received signals and horizontal ones are time, starting at 170 simulation steps and ending at 250 simulation steps. The surface slope of the obstacle is 80 degrees with respect to the x axis in (a), and 70 degrees in (b).
Figure 7.12 An environment with a transmitter, two receivers and an obstacle with a flat surface is simulated in near field with the Lattice Gas model using 3.5 cells / wavelength.

7.2.2 Simulation in the far field

The Lattice Gas model is limited by the size of the simulation array. In addition, it will not cease its simulation step by step until the echo returns to the receivers from an obstacle far away from receivers (and / or the source). In this case, the Continuous model is more suitable. The Continuous model is a time consuming model to run, however, this does not result in a serious problem when we are simulating a simple environment with one obstacle, and a few points, such as the two points where receivers are sitting, instead of calculating values at points spanning a whole environment.

The simulation application developed in this work allows for simulation of one transmitter and two receivers with the Continuous model. The user interface is as shown in Figure 7.13.
### Figure 7.13
The user interface for the option of simulating one transmitter and two receivers with the Continuous model.

Some parameters in the above user interface are specified as in Figure 7.14.

### Figure 7.14
Construction of one transmitter and two receivers. The obstacle pivot is specified at the centre of an obstacle.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Transmitter</th>
<th>Obstacle</th>
<th>Receiver</th>
<th>Ruler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position x</td>
<td>30</td>
<td>350</td>
<td>28</td>
<td>30</td>
</tr>
<tr>
<td>Position y</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>230</td>
</tr>
<tr>
<td>Beam width (degrees)</td>
<td>30</td>
<td>90</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>length</td>
<td>10</td>
<td>100</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Cancel
OK
In Figures 7.15 to 7.19, simplified symbols are used to represent a transmitter, denoted T, two receivers, R1 and R2, and an obstacle, located at the right side of the pictures. Different wave paths are drawn in different colours, and their corresponding times of flight are drawn on the time axis, located at the bottom of the pictures, in identical colours. The obstacle can be a plane shape, a corner shape or a stair shape. Algorithm 6.1 for multi-path reflection off plane surfaces is used.

Figure 7.15 The obstacle is specified to be a plane shape in a vertical direction. The echoes arrive at receivers simultaneously.
Figure 7.16 The obstacle is defined with a small slope. The echo arriving at receiver R1 is earlier than that at receiver R2. However the echo arriving at receiver R2 is outside out of the receiver beam angle (default value of 30 degrees).

Figure 7.17 A corner shape is chosen for the obstacle. Two wave paths exist for waves to travel from the transmitter to each receiver.
Figure 7.18 A stair shape is chosen for the obstacle and it has been rotated around pivot by 100 degrees. There is only one path for waves to travel to each receiver.

Figure 7.19 The obstacle is rotated about the pivot by 110 degrees. Multi-path reflections exist for waves travelling from the transmitter to the two receivers.
Chapter 8
Conclusion

The visualisation of sound waves is important in the study of acoustics in natural systems, such as bat echolocation, and artificial systems, such as ultrasonic aids for blind people and ultrasonic sensing for mobile robots. Visualisation helps us to understand a physical phenomena that we can not see. However, the results of visualisation are only as useful as the accuracy of the underlying models.

Sound waves have been simulated in this thesis with three models: two discrete and one continuous. They have been reviewed and extended. The theoretical basis for these simulations has been discussed. In Chapter 2 the details of the Transmission Line Matrix model are given, and its features are discussed. The continuous model is discussed in Chapters 3, 5 and 6. The Lattice Gas model is discussed in Chapter 4. In Chapter 7, visualisations of several common robotic sensing situations are presented.

8.1 Transmission Line Matrix model

The advantages of this model are computational simplicity and inherent parallelism. As the model is based on circuit components which are easy to understand, the nature and significance of errors can be intuitively analysed. The algorithm for this model is simple, and can be used to simulate a complicated environment. The program for the main parts of this model is only a few lines long, so it is easy to develop and maintain.

However, the ability of this model to simulate wave propagation with different speeds is an open question. This is because the velocity of electronic waves is a constant when current flows through a uniform network. Also, it is a uniform network where all nodes have the same impedance. Obstacles are simulated by open circuits in the mesh in the simulation reported here.
In the two-dimensional case, as its time complexity is $O(s \cdot m \cdot n)$, and the space complexity is $16mn + 16(m + 2)(n + 2) + \lceil mn / 8 \rceil$, this model is limited to simulations in the near field.

### 8.2 Lattice Gas model

In Chapter 4, the Lattice Gas model has been discussed in detail. The Lattice Gas model is a discrete simulation model. It has often conveniently been used to model physical systems containing many discrete elements with local interactions. It has the advantage over TLM that it can model changes of speed or impedance.

The iterative form, Equation 4.10, has been derived from Krutar's model. It is a general rule at cells where the impedance is constant. The rules at cells near a boundary where the impedance varies are presented, in Equation 4.21. It takes the speed of wave travel and the impedance of the media into account. The window edge problem has been posed, and several solutions have been given.

Equation 4.21 is the kernel part in this LG model. The simulation of various environments is realised by initialising the simulation array to model the configuration of the environment. This process is not embedded in nested loops of the main procedure Algorithm 4.1. Therefore, the algorithms presented in Chapter 4 are highly efficient. A corresponding data structure has been described in Section 4.3.3. It makes use of indices to specify the configuration of the environment, instead of storing the values for speed and impedance in each cell, so that the data structure occupies memory space economically.

An attempt to simulate diffuse reflection has been explored with a one-cell or two-cell zigzag interface in Sections 4.3.1 and 4.3.2. Although the results do not show apparent difference from that of specular reflection, it is a worthwhile trial.

The window edge problem restricts the application of cell based simulation models to small regions of space unless very large arrays are used. The calculation at window
edges with the Continuous model, introduced in Section 4.4.2.2, is a good solution when the simulated environment is simple. The Amplitude Compensation method, introduced in Section 4.4.2.5, is another solution. It shows a good result when being applied to the far field. The Post History method, introduced in Section 4.4.2.6 is the best solution. It performs well in both the near field (Figure 4.14) and the far field (Figures 7.1 - 7.5). Though it cannot avoid reflection from the window edge completely, it reduces it significantly as compared with other methods.

In Section 4.5, a sort of noise, wave tails, has been observed. It has also been observed that the noise due to tails is greater when the simulation wavelength is very short. To avoid this sort of noise, the wavelength should be set to no less than 15 cells. Alternatively, the colour mapping technology, Algorithm 4.7, introduced in Section 4.5, can be adopted.

The memory space for this LG model has been analysed in Section 4.2. It is:

\[ (4 + \left\lfloor \frac{(4 \times N^2 - 3N + 4)}{8} \right\rfloor) (m + 2)(n + 2) + 4mn + 16m + 8n + 8N^2 - 4N + 40 \text{ bytes} \]

where

- \( N \) is the number of media, and
- \( m \) and \( n \) are the width and height of a simulation window.

If two media are simulated, \( N=2 \), the memory space required is:

\[ 10mn + 28m + 20n + 56 \text{ bytes} \]

The computing time complexity is \( O(s \times m \times n) \), where \( s \) is the number of simulation steps.
8.3 Continuous model

In Chapter 3, simulation models for modelling wave propagation, reflection, diffraction, and interference have been built separately. Practical examples are given to simulate waves emitted from a cylindrical wave source. In fact, a cylindrical wave source is a source in three-dimensions. Yet as the wave pressure in the third dimension is a constant, its mathematical expression is in a two-dimensional form. The solution for the wave equation is more complex in two-dimensions than in one-dimension or in three-dimensions, as discussed in Section 3.1. An approximate form, Equation 3.8, has been used in this visualisation.

Diffraction from a knife edge has been visualised in Section 3.3, with Equation 3.12. The desired result is shown in Figure 3.8. Wave interference is visualised with Equation 3.20. Figure 3.11 shows an example of wave interference from two point sources.

In Section 3.5, the directional characteristic of a transmitter has been modelled. Ray tracing technology has been applied to visualise the beam pattern in three-dimensions. Figures 3.12 and 3.13 show the results with three light sources, red, green and blue, in order to strengthen the three-dimensional effect.

Reflection from a flat surface has been discussed in Section 3.2. The mirror equation provides us with a solution for finding an imaginary source. Reflection from an arbitrary geometric shape is discussed in Chapter 5.

In Chapter 5, algorithms have been presented for calculating the shape of a wave scattered from a curved obstacle. These algorithms calculate the wave pressure at all observation points in a region of space surrounding the obstacle by determining the pressure at imaginary sources located at the centres of the curvature of the reflected wavefronts. Solutions to waves scattering from four obstacles have been presented to illustrate the details and the general nature of these algorithms.

First, an equation was derived to calculate the wave pressure at an observation point at known distances from the wave source and imaginary source. To use this formula, an
equation was derived for the location of the imaginary source for the reflected wave that passes through the observation point. These equations are used in the $\alpha - r$ algorithm and the $X-Y$ algorithm. Both algorithms use nested loops to span space.

The nested loops in the two algorithms use different increment variables to span space. As a consequence, the $\alpha - r$ algorithm is suitable for modelling beams, but has quantisation errors. In contrast, the $X-Y$ algorithm is a better space spanning algorithm. Both algorithms are based on the assumption that the shape of the object can be described as a continuous curve.

In Chapter 6, algorithms for calculating multi-path reflection are given. An algorithm for calculating multi-path reflections from plane surfaces is discussed in Section 6.2. It has been used to produce examples of application to robotics in Section 7.2 (Figures 7.15 to 7.19). Algorithm 6.3, Wave Tracing, is presented to calculate multi-path reflection from curved surfaces.

The Continuous model can obtain a good result when simulating a limited number of points. For example, if we are only interested in the wave behaviour near the receivers (one or two), we only need calculate values at a few points. But their calculation times dramatically increase when the simulated environment becomes more complicated.

### 8.4 Comparison of models and conclusion

Both discrete models use a square grid and hence are suitable for real-time execution on a parallel computer. Also, they include all wave motion properties of interest in their equations. Due to the discretisation of space, the execution of the simulation is independent of the complexity of the environment. However, a number of problems arise as a result of the discretisation. First, special processing has to be performed at the boundaries. Second, quantisation errors in the discrete calculations appear as small noise waves. Third, wire frame rendering shows this noise as a zigzag on all the lines.
The continuous model requires separate models for each wave motion property, which must be combined using superposition to produce the final result. Also, the complexity of the calculation increases rapidly with the number of obstacles in the environment. As pressure is calculated at every point in space without reference to neighbouring points, continuous models do not have the window edge problem.

The wave pressure of a scattered wavefront varies with phase and with wavefront curvature in both time and space. The phase of the wave at an observation point is a function of time and distance from a source. The shape of the scattered wavefront is determined by the shape of the incident wave and the curvature of a reflecting obstacle.

The discrete models are simple to iterate and automatically show wave properties, such as propagation, reflection and diffraction. They are suitable for use in the near field, due to the memory requirements and simulation steps. Simulation in the far field requires many simulation steps, large amount of memory, and the signal becomes very weak. Also, the signal starts to mix with the noise from window edges.

Both TLM model and LG model are discrete models and have many similar aspects. However the LG is more flexible than TLM as its iterative form contains the values of impedance and speed. Hence the LG model is more suitable to simulate wave propagation with various speeds and with impedance changes affecting various ratios of reflection and transmission at a boundary. They have a same order of computing time complexity, but LG uses less memory space. Because TLM needs to store the values for four directions, this model can be used to visualise the direction of wave propagation, as shown in Figure 2.16.

The simulation work is mainly focused on two-dimensions in this thesis. It is not difficult to extend the models of TLM (Equations 2.64-2.68) and LG (Equation 4.21) to three-dimensions. However, the space complexity increases to \( O(m^2n^2) \). Consequently, the time complexity increases, as the cells to be calculated increase, to \( O(s \times m^2n^2) \). A three-dimensions version of the TLM model has been implemented, but the complete array did not fit into main memory so it had to be overlaid from a file, which considerably increased execution time.
The solution for the wave equation in three-dimensions used in the Continuous model has been given in Equation 3.9. We notice this solution is not much different from the solution for the wave equation in two-dimensions (Equation 3.8). The time complexity is a function of the number of view planes, that is, how many two-dimensions slices or layers are to be visualised. The Continuous model is not a cell based model, and the time complexity also depends on the size of a view window and the complexity of the calculation for each receiver point.

The other problem is that the equations describing the geometry are more complex in three-dimensions. For example, to calculate diffraction, the Fresnel-Kirchhoff diffraction formula (Equation 3.17) requires a surface integral when applied to three-dimensions, instead of the curve integral used in two-dimensions. This results in increasing time complexity.

Increased geometric complexity and calculation complexity considerably increases the calculation times, but when correct models are used they should not change the visualisation. However, differences in the physics in three dimensions will cause different visualisation results. For example, the attenuation of the wave front due to spreading loss is different for a spherical wave front from a point source to a cylindrical wave from a line source. The 2D visualisation correctly models the 3D cylindrical wave but not the 3D spherical wave.

The echo reaching a receiver is the superposition of echoes that propagate along multiple paths from the transmitter to the receiver. In three-dimensions there are many more possible paths. As a result, both the time complexity and the visualisation results may differ from a two-dimensions visualisation.

Finally, a very powerful computer is required for animation. For this reason, discrete models will be used in educational tools which must execute on personal computers. Researchers who are interested in the greater accuracy of continuous models will either require access to a powerful machine or be content with snapshots. With all models, it is difficult to achieve real-time animation, so the computer equivalent of a ripple tank will calculate and store the images and then play them back in real time.
8.5 Future work

So far the wave propagation has been rendered with 2D techniques, and has used colour intensity to show the magnitude of element motion up or down at every x-y position. The future work to this topic is to model wave motion in three dimensional space, to enable visualisation of sound motion in air.

Visualisation is useful for understanding ultrasonic sensing for robotics. Discrete models have been developed for generating time sequences of images, and a continuous model for snap shot images. Scientific visualisation can visualise either the real time sampled data or simulation data. The work for this thesis has been to visualise simulation data. To build a correct simulation model has been the first step in this case. To this end, several simulation models have been explored, compared, and refined. The next step is to visualise them with more novel visualisation technology. Other flow visualisation technology, such as surface particles, may be used to produce more meaningful images, in order to provide researchers with more useful tools.
Chapter 9

References


Ancona, M. G., 1990, 'Lattice Gas approach to semiconductor device simulation', Solid

Appert, C., d'Humieres, D., Pot, V. and Zaleski, S., 1994, 'Three-dimensional Lattice
Gas with Minimal Interactions', Transport Theory and Statistical Physics, 23(1-3),
pp. 107-122.

to Electromagnetic Problems', IEEE Transactions on Magnetics, Vol. 25, No. 4,
pp. 2916-2918.


Chernoff, H., 1973, 'The Use of Faces to Represent Points in k-Dimensional Space
Graphically', Journal of the American Statistical Association, Vol. 68, No. 342,
pp. 361-368.

Christopoulos, C., Johns, P. B., Mallik, A. and Naylor, P., 1987, 'The Use of
Automotive Electronics, pp. 159-163.

Modelling Applied to Problems in Underwater Acoustics', Oceans, Conference
Proceedings - Oceans'90, pp. 216-220.

Modelling', IEE Colloquium, Electronics Division Tutorial Colloquium on


Eagleman, J. R., 1976, 'The Visualisation of Climate', Lexington, Mass. [etc.].


Skudrzyk, ge shape as they propagate. *oustics*, Springer-Verlag.


Appendix

Appendix A Notation

\( B \) compressibility of medium
\( C \) capacitance in the electronic network of the TLM model
\( c \) speed of sound
\( c_s \) speed of light
\( c_s \) simulation speed
\( D \) distance between the wires in transmission lines
\( HPP \) a Lattice Gas model and is acronym on authors’ name: Hardy, de Pazzis and Pomeau
\( i \) current in the electronic network of the TLM model; intensity of sound wave
\( I \) the set of inputs meaningful to the automaton intensity of sound wave
\( J_0 \) Bessel function of first kind of zero order
\( k \) angular wavenumber
\( l \) distance from the source to an obstacle
\( l_f \) distance from the focus point from an obstacle
\( L \) Inductance in the electronic network of the TLM model
\( m \) width of a simulation window
\( m_a \) is weighted coefficient of the pressure \( P \) at the direction \( a \)
\( n \) height of a simulation window
\( N \) number of simulated media
\( N_0 \) Bessel function of second kind of zero order
\( P \) sound pressure in the air
\( P_p(n) \) the number of possible paths with \( n \) surfaces in an environment
\( P_m \) pressure amplitude of sound

\( P_{inc} \) incident pressure of sound waves

\( P_{ref} \) reflected pressure of sound waves

\( P_r \) reflecting point

\( R \) reflected coefficient

radius of the obstacle in mirror equation

\( r \) radius of wire in transmission line;

radius of a transducer;

distance

\( s \) displacement of element from its equilibrium

the number of simulation steps

\( S' \) imaginary point

\( S_m \) displacement amplitude

\( s_r \) space rate of simulation model to real world

\( t_r \) time rate of simulation model to real world

\( T \) period;

transmission coefficient

\( T_s \) period in simulation model

\( u \) particle speed in the waves of sound

\( V \) voltage in the electronic network of the TLM model

\( v \) voltage in the electronic network of the TLM model

\( V_{\phi} \) velocity of wave travelling

\( Q \) the set of discrete internal states of the automaton

\( Z \) the set of outputs generated by the automaton;

sound characteristic impedance

\( \delta \) the function that relates every pair of elements taken from sets I and Q;

angle of the reflected ray to the x axis

\( \omega \) the function that relates every pair of elements \( i, q \) to an element of \( Z \)

angular frequency
\( \rho \)  
density of medium

\( \lambda \)  
wavelength

\( \theta_i \)  
angle of incidence

\( \theta_r \)  
angle of reflection

\( \phi \)  
angle of the direction of incident waves to the direction of the observation point

\( \varphi_0 \)  
phase difference between the initial angles of two sources

\( \Psi \)  
angle of the obstacle plane with the orthogonal direction of incident waves
### Appendix B Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bounced reflection</strong></td>
<td>reflection from the same surface more than once</td>
</tr>
<tr>
<td><strong>connection process</strong></td>
<td>waves propagate toward its neighbours after waves scatter from a cell</td>
</tr>
<tr>
<td><strong>cutoff frequency</strong></td>
<td>the frequency at which waves cannot propagate. TLM model can only simulate well in the range from zero frequency to cutoff frequency</td>
</tr>
<tr>
<td><strong>dispersive</strong></td>
<td>waves at different frequencies travel at different speeds causing wave forms to change shape as they propagate</td>
</tr>
<tr>
<td><strong>far field</strong></td>
<td>the field further than the distance from the source: ( \frac{r^2}{\lambda} - \frac{\lambda}{4} ), where ( r ) is the radius of a transducer and ( \lambda ) is wavelength</td>
</tr>
<tr>
<td><strong>hyperstreamlines</strong></td>
<td>a generalisation of vector field streamlines</td>
</tr>
<tr>
<td><strong>instantaneous amplitude</strong></td>
<td>amplitude at instantaneous time</td>
</tr>
<tr>
<td><strong>LG</strong></td>
<td>Lattice Gas</td>
</tr>
<tr>
<td><strong>longitudinal waves</strong></td>
<td>vibrations are along the wave propagation direction</td>
</tr>
<tr>
<td><strong>multi-path reflection</strong></td>
<td>As sound waves travel from a transmitter to a receiver, the waves are not reflected more than once from objects</td>
</tr>
<tr>
<td><strong>near field</strong></td>
<td>the field within the distance: ( \frac{r^2}{\lambda} - \frac{\lambda}{4} ), where ( r ) is the radius of a transducer and ( \lambda ) is wavelength</td>
</tr>
<tr>
<td><strong>normal incidence</strong></td>
<td>the incident waves travel in a direction normal to an interface</td>
</tr>
<tr>
<td><strong>peak to peak amplitude</strong></td>
<td>amplitude range in one period from the minimum to maximum</td>
</tr>
<tr>
<td><strong>principal axis</strong></td>
<td>the line drawn from a wave source to a curvature centre of a surface</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>reflection coefficient</strong></td>
<td>the ratio of the reflected pressure amplitude to incident pressure amplitude</td>
</tr>
<tr>
<td><strong>scattering process</strong></td>
<td>waves scatter from a cell after impulses incident on the cell</td>
</tr>
<tr>
<td><strong>transmission coefficient</strong></td>
<td>the ratio of the transmitted pressure amplitude to incident pressure amplitude</td>
</tr>
<tr>
<td><strong>TLM</strong></td>
<td>Transmission Line Matrix</td>
</tr>
<tr>
<td><strong>volume visualisation</strong></td>
<td>projects a multidimensional data set onto a 2D image plane</td>
</tr>
<tr>
<td><strong>wavelet transforms</strong></td>
<td>wavelet transforms decompose a signal with a family of functions, which have local properties in both the time and frequency domains</td>
</tr>
<tr>
<td><strong>window edge</strong></td>
<td>a boundary where the visible vision space becomes invisible</td>
</tr>
</tbody>
</table>
Appendix C Equation for the example in Section 5.6 of Chapter 5

\[ X(\alpha) = x + (R - \sqrt{x^2 + y^2})\cos \delta \]

\[ \dot{X} = \dot{x} - \frac{x \dot{x} + y \dot{y}}{\sqrt{x^2 + y^2}} \cos \delta - (R - \sqrt{x^2 + y^2})\sin \delta \dot{\delta} \]

\[ \ddot{X} = \ddot{x} + \left( \frac{(x \ddot{x} + y \ddot{y})^2}{(\sqrt{x^2 + y^2})^3} - \frac{x^2 + x \dot{x} + 1}{\sqrt{x^2 + y^2}} \right) \cos \delta + 2 \frac{x \dot{x} + y \dot{y}}{\sqrt{x^2 + y^2}} \sin \delta \dot{\delta} \]

\[ -(R - \sqrt{x^2 + y^2})(\cos \delta \dot{\delta}^2 + \sin \delta \ddot{\delta}) \]

\[ Y(\alpha) = y + (R - \sqrt{x^2 + y^2})\sin \delta \]

\[ \dot{Y} = \dot{y} - \frac{x \dot{x} + y \dot{y}}{\sqrt{x^2 + y^2}} \sin \delta + (R - \sqrt{x^2 + y^2})\cos \delta \dot{\delta} \]

\[ \ddot{Y} = \ddot{y} + \left( \frac{(x \ddot{x} + y \ddot{y})^2}{(\sqrt{x^2 + y^2})^3} - \frac{x^2 + x \dot{x} + y^2 + y \dot{y}}{\sqrt{x^2 + y^2}} \right) \sin \delta - 2 \frac{x \dot{x} + y \dot{y}}{\sqrt{x^2 + y^2}} \cos \delta \dot{\delta} \]

\[ +(R - \sqrt{x^2 + y^2})(-\sin \delta \dot{\delta}^2 + \cos \delta \ddot{\delta}) \]

\[ \delta = \begin{cases} 
\pi - (\alpha + 2\beta) & \alpha + 2\beta \geq 0 \\
-\pi - (\alpha + 2\beta) & \alpha + 2\beta < 0 
\end{cases} \]

\[ \dot{\delta} = -1 - 2\beta, \]

\[ \ddot{\delta} = -2\ddot{\beta} \]

\[ \beta = \tan^{-1} \left( \frac{\dot{x}}{\dot{y}} \right) \]

\[ \dot{\beta} = \frac{x \ddot{y} - \dot{x} \ddot{y}}{x^2 + y^2} \]
\[
\ddot{\beta} = \frac{-2(\dddot{x} + \dddot{y})(\dddot{x} - \dddot{y})}{(x^2 + y^2)^2} + \frac{\dddot{x} + \dddot{y} - \dddot{x} - \dddot{y}}{x^2 + y^2} = \frac{-2(\dddot{x} + \dddot{y})}{(x^2 + y^2)} \dot{\beta} + \frac{\dddot{x} + \dddot{y}}{x^2 + y^2}
\]

\[
s = \frac{\dot{y}}{\dot{x}},
\]

\[
\dot{s} = -\frac{\dot{x}}{x} s + \frac{\dot{y}}{\dot{x}}.
\]

\[
\ddot{s} = \frac{-2\dot{s}\dddot{x} - \dddot{x} s + \dddot{y}}{x}
\]

\[
\dot{x} = 600 \frac{a^3 \alpha}{(\alpha^2 + a^2)^2},
\]

\[
\ddot{x} = \frac{600a^3}{(\alpha^2 + a^2)^2} (1 - \frac{4\alpha^2}{\alpha^2 + a^2}),
\]

\[
\dddot{x} = \frac{7200a^3(\alpha^2 - a^2)}{(\alpha^2 + a^2)^4}
\]

\[
\dot{y} = \frac{x}{\cos^2 \alpha} + \dot{x} \tan \alpha,
\]

\[
\ddot{y} = 2\dot{y} \tan \alpha + 2\dot{x} + \dddot{x} \tan \alpha,
\]

\[
\dddot{y} = \frac{2\ddot{y}}{\cos^2 \alpha} + 2\ddot{y} \tan \alpha + 2\dot{x} + \dot{x} + \dddot{x} \tan \alpha
\]
Appendix D Probing ray paths of multipath reflection for the example in Section 6.2.3

Following pictures show the process of trying paths reflected from different objects.

**Figure A.1** Trying path of S-L2-L1-R.

**Figure A.2** Trying path of S-L2-L3-R.
Figure A.3 Trying path of S-L1-L3-R.
Figure A.4 Trying path of S-L3-L1-R.
Figure A.5 Trying path of S-L3-L2-R.

Figure A.6 Trying path of S-L2-L1-L3-R.
Figure A.7 Trying path of S-L₁-L₃-L₂-R.
Figure A.8 Trying path of S-L₂-L₃-L₁-R.
Figure A.9 Trying path of S-L3-L2-L1-R, and find it.
Figure A.10 Trying path of S-L1-L2-L3-O, and find it.
Figure A.11 Trying path of S-L₃-L₁-L₂-R.
Appendix E Code

A.E.1 TLM

// The following codes are for implementation of TLM simulation model in two-dimension, introduced in
// Section 2.2.2
void TTLM::Simulation() // the main loop for TLM model
{
    for (short i = 0; i < fSimulationStep; i++)
    {
        this->Source_emit(i);
        this->Connection();
        this->Scattering();
    }
    this->Display();
};

void TTLM::Scattering() // Implementation of Equation 2.64
{
    short i, j;
    for (i = 0; i < fScreenSize.width+1; i++)
    {
        for (j = 0; j < fScreenSize.height+1; j++)
        {
            if (transmission[i][j] != 0.) {
                for(short k = 0; k < 4; k++)
                fReflection[k][i][j] = 0.5 * (fIncidence[0][i][j] + fIncidence[1][i][j] + fIncidence[2][i][j] - fIncidence[k][i][j]);
            } else {
                for(short k = 0; k < 4; k++)
                fReflection[k][i][j] = -fIncidence[k][i][j];
            }
    }
};

void TTLM::Connection() // Implementation of Equations 2.65-2.68
{
    short i, j;
    for (i = 0; i < fScreenSize.width+1; i++)
    for (j = 0; j < fScreenSize.height+1; j++)
    {
        if (j)
        fIncidence[0][i][j] = fReflection[2][i][j-1];
        if (i)
        fIncidence[1][i][j] = fReflection[3][i-1][j];
        if (j != fScreenSize.height)
        fIncidence[2][i][j] = fReflection[0][i][j+1];
        if (i != fScreenSize.width)
        fIncidence[3][i][j] = fReflection[1][i+1][j];
    }
}

A.E.2 LG

// The codes following are in relate to boundary rules discussed in Section 4.3.
void TNewRuleCa::Go() // The main loop for LG simulation model
{
    fObstacle->initMedium();
    for (short i = 0; i < fSimulationStep; i++)
    {
        fSource->initWave(i);
        this->Propagation();
    }
    this->Display();
};

void TNewRuleCaObstacle::initMedium()
{
    for (short i = 0; i < fWidth + 1; i++)

for (short j = 0; j < fHeight + 1; j++)
{
    fCa->fScreen[i][j].medium = speed2;
    for (short k = 0; k < 4; k++)
        fCa->fScreen[i][j].coef[k] = one2two;
}

this->SetObstacle();

void TNewRuleFlatObstacle::SetObstacle()
{ // Boundary setting of a flat obstacle
    short i, j;
    if (fSurfText == zigzag) { // simulate diffusion
        if (Two_cell_zigzag) { // simulate a two-cell interface
            for (j = 1; j <= (short)fHeight - 2; j += 4)
            {
                i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4);
                fCa->fScreen[i][j].coef[0] = one2two;
                for (short k = 1; k <= 3; k++)
                    fCa->fScreen[i][j].coef[k] = two2one;
                fCa->fScreen[i+1][j].coef[0] = one2two;
                fCa->fScreen[i-1][j].coef[0] = one2two;
                fCa->fScreen[i+2][j].coef[0] = one2two;
                fCa->fScreen[i+2][j-1].coef[0] = one2two;
                fCa->fScreen[i+2][j+1].coef[0] = one2two;
                fCa->fScreen[i+2][j+2].coef[0] = one2two;
                fCa->fScreen[i+2][j].coef[1] = two2one;
                fCa->fScreen[i+2][j].coef[3] = two2one;
            }
            for (i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4 + 1); i < fWidth; i++)
                fCa->fScreen[i][j].medium = speed2;
        }
        else if (One_cell_zigzag) { // simulate one-cell interface
            for (j = 0; j <= (short)fHeight; j += 2)
            {
                i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4);
                for (short k = 1; k <= 3; k++)
                    fCa->fScreen[i][j].coef[k] = one2two;
                fCa->fScreen[i-1][j].coef[0] = one2two;
                for (i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4 + 1); i < fWidth; i++)
                    fCa->fScreen[i][j].medium = speed2;
            }
        } else if (fSurfText == smooth) { // simulate specular reflection
            for (j = 0; j <= (short)fHeight; j++)
            {
                i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4);
                for (short k = 0; k < 4; k++)
                    fCa->fScreen[i][j].coef[k] = one2two;
                fCa->fScreen[i][j].coef[0] = one2two;
                for (i = (short) (j * tan(fSlopDegree) + fWidth * 3 / 4 + 1); i < fWidth; i++)
                    fCa->fScreen[i][j].medium = speed2;
            }
        }
    }
    else if (fBoundary == old_rule) { // Equation 4.11
        for (i = 1; i < fScreenSize.width; i++)
        {
            fCa->fScreen[i][j].medium = speed2;
        }
    }
}
void TNewRuleCa::Propagation() // simulate propagation and reflection
{ short i, j;
if (fBoundary == old_rule)
for (i = 1; i < fScreenSize.width; i++)
{...
for ( j = 1; j < fScreenSize.height; j ++)
{
    temp[i][j] =
        speed[fScreen[i][j+1].medium] * fScreen[i][j+1].pressure -
        speed[fScreen[i+1][j].medium] * fScreen[i+1][j].pressure -
        speed[fScreen[i-1][j].medium] * fScreen[i-1][j].pressure -
        4 * speed[fScreen[i][j].medium] * fScreen[i][j].pressure -
        2 * fScreen[i][j].pressure - fScreenOld[i][j];
};
    this->UpdateScreen(i);
}
} // if

for ( j = 0; j < fScreenSize.height + 1; j ++) // handle the cells at an edge
{
    fScreenOld[fScreenSize.width - 1][j] = fScreen[fScreenSize.width - 1][j].pressure;
    fScreen[fScreenSize.width - 1][j].pressure = temp[i][j];
    fScreenOld[fScreenSize.width][j] = fScreen[fScreenSize.width][j].pressure;
} // for j
}
} // for i

A.E.3 Continuous model

PROCEDURE Diffraction(waveSource, obstPoint, screPoint : realPoint;
    wavelength, t, i : REAL) : REAL; // Diffraction (Morse. 1968)
PROCEDURE Pressures, fai, psai, k, t, velocity : REAL) : REAL; // Equation 3.12
VAR
    X, z : REAL;
    f : REAL;
BEGIN
    TimeRange(t, -r * cos(fai) / velocity) THEN // at current time and location
        x := k * r * cos(fai);
        z := sqrt(2. * k * r) * cos(0.5 * fai);
        f := cos(x) * E1(z) - sin(x) * E2(z);
    ELSE
        f := 0.;
    END;
    IF TimeRange(t, r * cos(fai + 2. * psai) / velocity) THEN
        x := k * r * cos(fai + 2. * psai);
        z := sqrt(2. * k * r) * cos(1.5 * PI - psai - 0.5 * fai);
        RETURN cos(x) * E1(z) - sin(x) * E2(z) + f;
    ELSE
        RETURN f;
    END;
END Pressure;

VAR
    r, t, f, fa, psai, k, w : REAL;
BEGIN
    r := sqrt(TwoPointDistance(obstPoint, screPoint));
    IF screPoint.x # obstPoint.x THEN
        fa := -arctan((screPoint.y - obstPoint.y) / (screPoint.x - obstPoint.x));
    ELSE
        IF screPoint.y > obstPoint.y THEN
            fa := PI / 2.;
        ELSE
            fa := - PI / 2.;
        END;
    END;
    psai := 0.;
END.
k := 2. * PI / wavelength;
w := velocity * k;
RETURN sin(w * t - k * (screePoint.x - waveSource.x)) * Pressure(r, psi, psi, k, t, velocity);
END Diffraction;

PROCEDURE E1(z : REAL) : REAL;
BEGIN
IF z > 0. THEN
  RETURN 1. - 0.5 * (C(z * z) - S(z * z));
ELSE
  RETURN 0.5 * (1. - C(z * z) - S(z * z));
END;
END E1;

PROCEDURE E2(z : REAL) : REAL;
BEGIN
IF z > 0. THEN
  RETURN - 0.5 * (C(z * z) - S(z * z));
ELSE
  RETURN 0.5 * (C(z * z) - S(z * z));
END;
END E2;

PROCEDURE C(z : REAL) : REAL;
VAR
  a, b, c : REAL;
BEGIN
  sine := sin(PI * Squar(z) / 2.);
  cosin := cos(PI * Squar(z) / 2.);
  IF ABS(z) > 2.45 THEN
    a := FresnelA1(w);
    b := FresnelB1(w);
    c := 0.5 - 1./ PI / ABS(z) * (b * cosin - a * sine);
    IF z < 0. THEN
      c := -c;
    END;
  ELSE
    a := FresnelA(z);
    b := FresnelB(z);
    c := z * (a * cosin + b * sine);
  END;
  RETURN c;
END C;

PROCEDURE S(z : REAL) : REAL;
VAR
  a, b, s : REAL;
BEGIN
  sine := sin(PI * Squar(z) / 2.);
  cosin := cos(PI * Squar(z) / 2.);
  IF ABS(z) > 2.45 THEN
    a := FresnelA1(w);
    b := FresnelB1(w);
    s := 0.5 - 1./ PI / ABS(z) * (a * cosin + b * sine);
    IF z < 0. THEN
      s := -s;
    END;
  ELSE
    a := FresnelA(z);
    b := FresnelB(z);
    s := z * (a * sine - b * cosin);
  END;
  RETURN s;
END S;

PROCEDURE FresnelA1(w : REAL) : REAL;
VAR
  k, ak, sum, coe : REAL;
BEGIN
  k := 0.;
  ak := -1.;
  sum := 0.;
  coe := 1.;
  REPEAT
    ak := -ak * coe;
    sum := sum + ak;
    k := k + 1.;
    coe := (4. * k - 1.) * (4. * k - 3.) / (4. * PI) / Squar(Squar(w));
  UNTIL (ABS(ak) < 0.000001) OR (coe > 0.9);
  RETURN sum;
END FresnelA1;

PROCEDURE FresnelB1(w : REAL) : REAL;
VAR
  k, ak, sum, coe : REAL;
BEGIN
  k := 0.;
ak := -1. / PI / w / w;
sum := 0.;
coe := 1.;
REPEAT
   ak := -ak * coe;
   sum := sum + ak;
   k := k + 1.;
   coe := (4. * k + 1.) * (4. * k - 1.) / (4. * k + 1.);
UNTIL (ABS(ak) < 0.000001) OR (coe > 0.9);
RETURN sum;
END FresnelB;

PROCEDURE FresnelA(w : REAL) : REAL; // when w approaches zero
VAR
   k : REAL;
   ak : REAL;
   sum : REAL;
BEGIN
   k := 0.;
   ak := 1.;
   sum := ak;
   REPEAT
      k := k + 1.;
      ak := -ak * PI * PI / (4. * k + 1.) * PI / (4. * k - 1.);
      sum := sum + ak;
   UNTIL (ABS(ak) < 0.000001) OR (k > 100.)
   RETURN sum;
END FresnelA;

PROCEDURE FresnelB(w : REAL) : REAL; // when w approaches zero
VAR
   k : REAL;
   ak : REAL;
   sum : REAL;
BEGIN
   k := 0.;
   ak := PI - w - w / 3.;
   sum := ak;
   REPEAT
      k := k + 1.;
      ak := -ak * PI / (4. * k + 3.) * PI / (4. * k + 1.);
      sum := sum + ak;
   UNTIL (ABS(ak) < 0.000001) OR (k > 100.)
   RETURN sum;
END FresnelB;

/* The codes following are used to produce Figures 3.10 and 3.11, directional characteristic of sound source in Section 3.5. Ray Tracing technology has been used. The member function TRayTrace::ReadScene() is not included, which handles an input file. The input file specifies various parameters, such as wavenumber, the radius of the piston source, viewer position, number, color and position of light sources, etc. */

void TRayTrace::Screen() { // The output goes to the 'picfile' file.
   // named by a user, which is in PICT Macintosh format, can be viewed by Picture Compressor
   Ray ray;
   CVec base;
   int xrange, yrange;
   char fname[41];
   scanf("%s", fname); // enter the name for the output file
   picfile = fopen(fname, "wb");
   unsigned short i = view.resx;
   fwrite(&i, 2, 1, picfile);
   i = view.resy;
   fwrite(&i, 2, 1, picfile);
   xrange = view.resx / 2;
   yrange = view.resy / 2;
   ray.P = view.eye;
   for (int j = yrange; j > -yrange; j --)
      for (int i = -xrange; i < xrange; i ++)
         {
            CColor buff;
            char col[3000];
            double t = t * view.vpx + view.origin;
            int k = 0;
            for (int i = -xrange; i < xrange; i ++)
               {
                  printf(datfile, "%d, %d
", j, i);
                  t = i;
                  ray.D = t * view.vpx + base;
                  ray.D = VecUnit(ray.D);
                  buff = CColor(0, 0, 0);
                  this->Trace(0, 1., ray, buff);
                  buff *= 255;
                  unsigned short red, green, blue;
                  red = (unsigned short)(buff._red);
                  green = (unsigned short)(buff._green);
                  blue = (unsigned short)(buff._blue);
                  col[k * 3] = (char)(red);
                  col[k * 3 + 1] = (char)(green);
                  col[k * 3 + 2] = (char)(blue);
               }
         }
}
col[k * 3 + 2] = (char)(blue);
k++;

buff *= 256;

C RGBColor color;                                  // convert data in buff to color

color.red = (unsigned short)(buff._red);
color.green = (unsigned short)(buff._green);
color.blue = (unsigned short)(buff._blue);

SetPixel(1 * 250, -j + 250, color);

fwrite(col, 3, view.resx, picfile);

fclose(picfile);

// convert data in buff to color
// Write to the output file

// convert data in buff to color
// Write to the output file

void TRayTrace::Trace(int level, float weight, Ray *ray, CColor *col)   // Ray tracing
{

CVec P, N;

Isect hit;

n = this->NearestIntersect(ray, modelroot, &hit);             // find a nearest intersection point with

if (n != 0)
{

Prim *prim = hit.prim;

P = ray->P + hit.t * ray->D;

N = prim->geometry->normal(P);                                // compute normal vector

if (VecDot(ray->D, N) > 0.) N = -N;

Shade(level, weight, P, N, ray->D, &hit, col);

}

else
{                                                     // shadow

*col = background;

}

int TRayTrace::NearestIntersect(Ray *ray, Comp *solid, Isect *hit) // find the nearest intersection

{                                                       // point

int na = 0, nhit = 0;

Comp *csg = solid;

Isect isect[4];

hit->t = 1000;

while (csg != NULL)
{

if (csg->compflag == 0)
{                                                // find the nearest intersection point of

Prim *prim = csg->info.prim;

na = prim->geometry->nearestinters(ray, prim, isect, rayeps);

// find the nearest intersection point

for (short i = 0; i < na; i++)
{

if (hit->t > isect[i].t)
{

*hit = isect[i];

}

}

else if (csg->compflag == 1)
{

Item *item = csg->info.item;

//

}

}

hit->t += na;

csg = csg->next;

return nhit;

}

void TRayTrace::Shade (int U, fl* weight, CVec P, CVec N, CVec I, CColor *col)   // shading

{                                                       // use Phong algorithm to calculate the

CColor diffuse = ambient;

CColor highlight = Color(0., 0., 0.);

Ray tray;

tray.P = P;

CVec Eye = -I;

float kdiff = hit->prim->surf->kdiff;

float kspec = hit->prim->surf->kspec;

float ktran = hit->prim->surf->ktran;

float refr = hit->prim->surf->refr;

float shine = hit->prim->surf->shine;

for (int i = 0; i < nligh; i++)
{

CVec L = light[i].loc - P;

CColor lcol = light[i].col;

float Llen = VecLen(L);

L = VecUnit(L);

tray.D = L;

float NdotL = VecDot(L, I);

if (z > 0) && Shadow(kray, Llen, &lcol)
{

diffuse = MixColor(NdotL, lcol, 1., diffuse);

}

if (kspec != 0)
{

CVec H = L + Eye;

H = VecUnit(H);

float Phong = kspec * pow(VecDot(D, H), shine);

}
if (Phong < rayeps) Phong = 0.;
highlight = MixColorfPhong, lcol, 1., highlight);

for (*color = ((double)diff) * diffuse;
*color = (*color + (hit->prim->surf->col);
*color *= (1. - ktran);
*color += highlight;
float krefl = (1 - ktran) * kspec;
if((level + 1) < maxlevel)
{
CColor tcol;
CColor bcolor = hit->prim->surf->col;
tray.P = P;
if(krefl * weight > minweight)
{
tray.D = SpecularDirection(I, N);
Trace(level + 1, krefl * weight, &tray, &tcol); // According to the reflection direction.
// ray tracing
CColor temcol = krefl * bcolor;
temcol = temcol * tcol;
*color = *color + temcol;
}
if (ktran * weight > minweight)
{
if (TransmissionDirection(hit->enter, refin, I, N, tray.D))
{
Trace(level + 1, ktran * weight, &tray, &tcol); // According to the transmission direction.
// ray tracing
*color = MixColor(ktran, tcol, 1., *color);
}
}
}

tR = SpecularDirectiond, N);
return R;

int TRayTrace::Shadow(Ray *ray, float tmax, CColor *lcol) // calculate shadow
{
Isect hit[100];
int nhit = Intersect(ray, modelroot, hit);
for (int i = 0; i < nhit; i++)
{
if (hit[i].t > (tmax + rayeps))
{
else{
float ktran = hit[i].prim->surf->ktran;
if (ktran == 0.) return 0;
CColor col = hit[i].prim->surf->col;
*lcol = MixColor(ktran, col, 1., *lcol);
}
return 1;
}
}
CVec TRayTrace::SpecularDirection(CVec I, CVec N) // calculate specular direction
{
CVec R = -2. * VecDotd, N) / VecLen(N) * VecUnit(N) + I;
return R;
}
int TRayTrace::TransmissionDirection(int enter, float refrin, CVec I, CVec N, CVec &t)
{
float eta, cl, cs2;
if (enter == 1) eta = (1./ refrin); else eta = refrin;
cl = -VecDotd, I, N) / VecLen(I) / VecLen(N));
// cos(theta1)
// cos(theta2) = cos(theta1) * cos(theta2)
if (cs2 < 0.) return 0;
T = (eta * I) + ((eta * cl - sqrt(cs2)) * VecLen(I) * VecUnit(N));
return 1;
}
int TBox::nearestinters(Ray* ray, Prim* prim, Isect hit{}); // find the nearest intersection point with
// the object defined by Equation 3.19.. the
// direction characteristic
{double t1, t2;}
int nroots = 0;
CVec D = ray->D;
CVec P = ray->P;
CVec C = fBeam.CEN;
CVec PminC = P - C;

CVec d = D;
d._x = D._x * 3;
d._y = D._y * 3;
PminC._x *= 3;
PminC._y *= 3;

double a = VecDot(d, d);
double b = 2 * VecDot(d, PminC);
double c = VecDot(PminC, PminC) - 1.2 * fBeam.size * fBeam.size;
if (!SolutToTwoTimes(a, b, c, tl, t2)) return 0;
if (tl > t2) (double t = tl, tl = t2; t2 = t;)
if (t2 < 0) return 0;

if (tl < 0) tl = rayeps;
double t = tl;
double solut;
nroots = 0;

do{
    // find intersection point
    if (BinaryDivision(&P, &D, &C, t, t + 0.05, solut))
        short enter;
        if (InBeam(&P, solut + rayeps, &D, fC)) enter = 1; else enter = -1;
        IsectAddl&hit[nroots ++], solut, prim, enter, 0);
    t += 0.05;
}while((t < t2) £.£. (nroots < 1));
return nroots;

CVec TBeam::normal(CVec P) // calculate normal of beam pattern
{
    CVec vec;
    vec = P - fBeam.CEN;

double vecLen = VecLen(vec);
double x = vec._x;
double y = vec._y;
double z = vec._z;
double theta = acos(z / vecLen);
double phi = acos(fabs(x) / sqrt(x * x + y * y));
if ((x < 0) && (y < 0)) phi = pi() + phi;
else if ((x < 0) && (y > 0)) phi = pi() - phi;
else if ((x > 0) && (y > 0)) phi = pi() - phi;

    double v = fBeam.k * fBeam.l * sin(theta);
    double besjl = BESJl(v);
    double signbesjl;
    if (besjl > 0.) signbesjl = 1; else signbesjl = -1.;
    double h, dh;
    if (v != 0){
        h = 2. * fabs(besjl / v) * fBeam.size;
        dh = 2. * fBeam.size * signbesjl * ( BESJ0(v) - 2 * besjl/v) / v
        * fBeam.k * fBeam.l * cos(theta);
    }else{
        h = fBeam.size;
        dh = 0;
    }

    vec._x = (-dh * cos(theta) + h * sin(theta)) * cos(phi);
    vec._y = (-dh * cos(theta) + h * sin(theta)) * sin(phi);
    vec._z = dh * sin(theta) + h * cos(theta);
    vec = VecUnit(vec);
    return vec;
}
for (i = 0; i < n; i++)
{
    this->PossiblePath(fObstacle->fObstacleTable.obstacle_number,
    aTable.obstacle_number = sufNo;
    for (short j = 0; j < sufNo; j++)
    aTable.obstacle_list[j] = fObstacle->fObstacleTable.obstacle_list[aSet[j]];
    if (this->ExamPath(fReceiver1, aTable)) // check possible paths one by one
    {
        short length = this->CalculatePathLength(fReceiver1,
        aTable, &aTable); // if it is a real path, calculate length of
        // the path
        ::ForeColor(ColorTable[(++colorIndex) % 8]);
        fReceiver1->DrawWave(length, fSpeed); // draw a mark on the time axis
        ::LineToPt(aTable.obstacle_list[j]->fIntersectionPt.CRealPtToPtO);
        ::LineToPt(fSource->position);
    }
}
if (this->ExamPath(fReceiver2, aTable)) // check receiver 2
{
    short length = this->CalculatePathLength(fReceiver2,
    aSource, &aTable); // draw the multipath
    ::ForeColor(ColorTable[(++colorIndex) % 8]);
    fReceiver2->DrawWave(length, fSpeed); // draw a mark on the time axis
    ::LineToPt(aTable.obstacle_list[j]->fIntersectionPt.CRealPtToPtO);
    ::LineToPt(fSource->position);
    if (true) // for
    delete aTable.obstacle_list;
}
}

void TEquTwoReceiver::PossiblePath(short SurfNum, short PathNo,
short aSet[], short &sufNo) // find the obstacle orders of possible paths
{
    short ElementSet[] = {0, 0, 0}, ElementOrder[], Enum = 0;
    short i = 0;
    do{
        PathNo -= ArangNum(SurfNum, ++i);
    } while (PathNo >= 0);
    PathNo += ArangNum(SurfNum, i);
    WriteOrder(ElementSet, i, SurfNum, 1, ElementOrder, Enum);
    surNo = i;
    for (short k = 0; k < i; k++)
    aSet[k] = ElementOrder[k + i * PathNo];
}
void TEquTwoReceiver::WriteOrder(short ElementSet[], short i, short n,
short level, short ElementOrder[], short &Enum) // give a series of ordered figures, the
// possible figures are in ElementSet, the
// results are in ElementOrder, the number
// of ordered figures is Enum
{
    if (i == 0)
    {
        short k = 1;
        do{
            for (short j = 0; j < n; j++)
            {
                if (ElementSet[j] == k)
                {
                    ElementOrder[Enum ++] = j;
                    break;
                }
            }
            k ++;
        } while (k <= n);
        return;
    } else{
        short j = 0;
        do{
            if (ElementSet[j] == 0)
            {
                ElementSet[j] = level;
                WriteOrder(ElementSet, i - 1, n, level + 1, ElementOrder, Enum);
                ElementSet[j] = 0;
                j ++;
            }
        } while (j < n);
    }
}

Boolean TEquTwoReceiver::ExamPath(CReceiver* aReceiver,
ObstacleTable* aTable) // check whether a path exists
{
    CRealPoint aRealPoint;
    aRealPoint.CPTToRealPt(aReceiver->position);
    short i;
    aTable->obstacle_list[0]->FindMirrorPoint(aRealPoint);
for (i = 1; i < aTable->obstacle_number; i++)
{
    aRealPoint = aTable->obstacle_list[i-1]->fMirrorPt;
    aTable->obstacle_list[i]->FindMirrorPoint(aRealPoint);
    // find the mirror points of obstacles one by one
}

i = aTable->obstacle_number - 1;
aRealPoint.CPtToRealPt(fSource->position);
Boolean sucess = aTable->obstacle_list[i]->FindIntersctPt_MirroPt(aRealPoint);
for (i = aTable->obstacle_number - 2; i > 0; i--) // find the intersection point, break, if the intersection point out of the obstacle.
{
    if (!sucess) break;
    aRealPoint = aTable->obstacle_list[i+1]->fIntersectionPt;
    sucess = aTable->obstacle_list[i]->FindIntersctPt_MirroPt(aRealPoint);
}
return sucess;

short TEquTwoReceiver::CalculatePathLength(CReceiver* aReceiver, CRealPoint aSource, ObstacleTable* aTable) // calculate the path length from the source to this receiver
{
    CRealPoint aPoint;
aPoint.CPtToRealPt(aReceiver->position);
    float distance = 0;
    for (short i = 0; i < aTable->obstacle_number; i++)
    {
        distance += TwoPointDistance(aPoint, aTable->obstacle_list[i]->fIntersectionPt);
        aPoint = aTable->obstacle_list[i]->fIntersectionPt;
    }
    distance += TwoPointDistance(aPoint, aSource);
    return (short)distance;
}