1993

Echolocation: evaluation of finite element analysis of ultrasonic transducers

Robert Pierre D'Souza
University of Wollongong

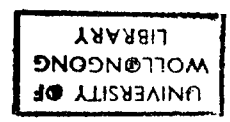
Echolocation : Evaluation of Finite Element Analysis of Ultrasonic Transducers

A thesis submitted in partial fulfilment of the

requirements for the award of the degree of

Honours Master of Science (Computer Science)

from



UNIVERSITY OF WOLLONGONG

by

Robert Pierre C. D'Souza, BSc, BSc(Tech) Bombay.

Department of Computer Science

1993

This is to certify that the work detailed by this thesis was done by the author, unless specified otherwise in the text, and that no part of it has been submitted in a thesis to any University.

R.P.C. D'Souza

Abstract

Echolocation is the process of building an acoustic image of the environment by sensing the ultrasonic echoes that are bounced off objects in the environment. It is naturally observed in bats and dolphins, who use it for navigation and orientation.

The Finite Element Method is one of several numerical techniques that simplify the abstract equations of calculus and obtain approximate solutions for real-world physical problems.

This thesis evaluates the suitability of various numerical 'Element Methods' for the modelling and design of ultrasonic transducers and the study of their radiated and scattered sound fields. To achieve this, it determines the selection criteria to choose between the alternate methods and identifies the essential set of elements that are required to model the problem. Further, it studies various commercial FEM software packages and identifies software language features necessary to implement the software package.

It arrives at conclusions regarding the basic requirements to develop a minimal software package capable of modelling ultrasonic transducers. On the basis of this thesis, software can be developed to provide a structure upon which future researchers can build and develop more complex models, such as those involving transducers and their interaction with the environment.

Acknowledgments

This thesis could NOT have been completed without a LOT of help and encouragement received from others, such as:

My supervisor, Dr. Phillip McKerrow, who always seemed to have (make?) spare time to answer my silly questions, and answered them with a straight face; and who gave me every opportunity to increase my knowledge regarding the subject (such as arranging for my attendance at a state-of-the-art seminar presented by Jean-Louis Migeot of Numerical Integration Technologies at the Acoustics and Vibration Centre of the Australian Defence Force Academy). It was invaluable!

My co-supervisor, Associate Professor Greg Doherty, who promised, if nothing else, to show me where to put the iteration variables in a 'for' loop; and ended up (giving me invaluable advice) holding my hand through much of the numerical maze.

My Head of Department, Professor Fergus O'Brien, for granting permission (and funds) for my attendance at the above seminar.

David Burnett, for the lucid explanations in his book, and the letters he wrote me in reply to my queries. John Denkmann, for the UNAFEM program, around which my software, FAME, has been modelled and based.

Maxine Lacey, who convinced me, (when both my hands were in Plaster-of-Paris casts), that I should take advantage of the Disabilities Services offered in the Union Building, to get some, if not all, of my software typed in; Greg Hampton and Gayle Ford, who, between them, ferreted out someone to do the typing; Helen Williams, who did the typing, and her husband, who read to her while she typed.

The members of our research team, who used to stoically sit through my seminars, nodding from time to time! (The nods usually increased towards the end of my seminars, though I never had the courage to ask if it was due to sleep, boredom or complete acceptance of my lucid explanations! Silence seemed infinitely preferable!) In particular, Ms. Shao Min Zhu, for her help with software headaches.

Jean-Louis Migeot, for his patience with my persistent queries after the seminar.

Thanks to the people with whom I stayed, for putting up with my comings and goings at odd hours, often wondering which thief was stealthily creeping about in the wee hours of the morning.

On a more personal note, a special word of thanks to my relatives and friends for their unceasing support, both financial and otherwise. When the road seemed difficult, their faith in my (assumed?) capabilities seemed to make it that much easier to take just another small step, and another, and .....

Thanks, all of you (including any others who I may have forgotten to mention). No doubt the list is long, but my memory is short and I've got to get to print!

Table of Contents

Chapter 1 - Introduction

1.1 The Focus of Our Research Group

I am a member of a research group that is studying and modelling echolocation. Echolocation is the process of building an acoustic image of the environment by sensing the ultrasonic echoes that are bounced off objects in the environment. It is naturally observed in bats and dolphins, who use it for navigation and orientation.

While we can hear audio frequency sounds (16 Hz to 20 KHz approx.), we can neither see nor hear ultrasonic frequency sounds. They exist in a frequency range that is beyond the threshold of human hearing capabilities. The goal of our project is to try to 'see' the ultrasonic echoes and interpret them to identify objects and learn about the environment. This requires a combination of physical observations, measurements, and computer imaging techniques.

This problem has diverse aspects, one of which is the study of ultrasonic transducers and their interaction with the environment.

1.2 The Aim of this Thesis

This thesis evaluates methods for the modelling of ultrasonic transducers, especially piezoelectric block, piezoelectric film and electrostatic. Two types of facets are to be modelled; the vibration of the transducer itself (both resonant and forced) and, through coupling to air, the resultant acoustic wave motion.

Different approaches are possible, including experimental observations to build an empirical model for the evaluation of echoes and the prediction of object shapes. The **aim of this thesis** is to examine/evaluate the suitability of numerical techniques for

this type of acoustics analysis, particularly the various 'element' methods like FEM[1], BEM[2], WEEM[3], etc.

Consequently, this thesis looks at a set of sub-problems, to arrive at some conclusions regarding how to model the transducers and their sound fields, and to ascertain the difficulty of achieving these goals. The sub-problems are :-

1) Can the techniques used and the models and equations developed for analysing structural vibration be directly transferred to this problem?
2) Is a special formulation of the Finite Element Method (FEM) required for acoustics problems?
3) Is the Boundary Element Method (BEM) a better approach? What about alternate formulations?
4) Are various kinds of mixed approaches, involving both the FEM and the BEM, more suitable?
5) What are the selection criteria for choosing between alternate approaches?
6) What elements are needed? What is their complexity?
7) What kind of software is needed? Would buying a commercial package be preferable to writing it oneself?
8) Which language is most suitable for 'Element Method' (EM) software programs? Are there any peculiarities that favour Fortran, the traditionally-used language? Are there any 'better' languages that can be used? What are the restrictions, if any, that preclude their use? What is the time frame for program development?
9) Are there any advantages in selecting a 'special' acoustics package (eg. SYSNOISE) rather than a general purpose problem-solver (eg. NASTRAN, ANSYS)?
10) Is the inverse problem possible? ie. given the sound field, can the cause of the sound field be recovered?

1. FEM ≡ FINITE ELEMENT METHOD
2. BEM ≡ BOUNDARY ELEMENT METHOD
3. WEEM ≡ WAVE ENVELOPE ELEMENT METHOD

1.3 The Approach

1.3.1 Diversity in Nature

Nature is fascinating in its various diverse life forms, each of which has specific biological systems that aid the organism in its day to day functioning. These biological systems are often 'tailor-made' to suit the organism's habitat and its preferred feeding pattern and mating rituals.

One very important biological system is the sense of hearing, which serves varied purposes in different organisms, ranging from locating, hunting and capturing of prey to a means of communication. Different organisms can perceive a different range of the sound spectrum, which varies from slightly more than 0 Hz to an upper limit of around 150 KHz. In Chapter 2, we examine the bat's hearing mechanism and study how echolocation helps in its 'night-to-night' activities.

1.3.2 Sound

The sense of hearing is the ability of living organisms to sense sound waves, which are mechanical disturbances that are propagated through solids, liquids and gases by longitudinal waves of alternate compression and decompression of the medium. The frequency of these waves is measured in cycles per second or Hertz (Hz).

Physical acoustics is the observation of sound in various physical systems, eg. strings vibrating, tuning forks, etc. Physiological acoustics is sound as observed by some sort of vibration sensing apparatus of biological origin. eg. bat hearing, in particular (Chapter 2).

I define 'Artificial' acoustics as the development of any man-made device that is used to study hearing, to prevent its deterioration (eg. ear-muffs, [Ciskowski91]) and to make it more acute (enhance sensitivity in the case of hearing loss).

When I talk about acoustics, I use it in a loose sense to include all of the above categories.

1.3.3 The Difficulty in Modelling the Ear

The physical principles that have been developed over the years for physical acoustics can not be directly applied to the mechanism of hearing. Physical concepts like elasticity, plasticity, mass, viscosity, fluidity, etc, have no simple equivalents applicable in the case of the ear. Hence, it becomes difficult to develop mathematical physical models of the ear. Additionally, the ear is buried deep in the temporal bone, making measurements of equivalent physical properties tedious. Specialised surgical techniques have to be devised to overcome this hurdle, and, quite often, these still result in damage to the ear, hence making corresponding measurements imprecise [Khanna80]. Besides, the measuring instruments themselves can affect the quantity being measured and hence the accuracy of measurements.

Hearing is dynamic in nature. The use of cadavers permits static characteristics to be measured. Bekesy [60] states that, "the elasticity of the basilar membrane of a guinea pig remained within a few percent of its normal value for several hours after death", while the eardrum's stiffness does increase if suitable precautions are not taken.

In spite of the difficulties involved, various experiments have been performed [Bekesy60] and different mathematical and computational models have been proposed to explain the observed phenomena. Geisler [76] gives an overview of the maths models that developed for the cochlea, and discusses their limitations.

1.3.4 Artificial Hearing?

It is evident that naturally occurring biological mechanisms (eg. echolocation) provide a high degree of functionality and versatility. Hence, humans try to replicate these mechanisms for use in daily activities, and, to facilitate this duplication, they

utilise the most convenient means, be it mechanical, electronic, etc. In particular, ultrasonic sensors have been used in diverse applications by ingenuously modifying a basic technique to suit the specific needs of the application. In Chapter 3, we examine the concept of the ideal transducer and study two basic transducer types, the electrostatic transducer and the piezoelectric transducer, used in the study of echolocation.

1.3.5 Modelling Views

In the study of any problem (study of transducers, in particular), we would like to know what is happening, understand how it happens and predict what would happen if various parameters are modified. There are two main approaches to the problem - the use of experimental observations or the use of numerical methods coupled with computer-aided-design. Their relative merits in tackling the problems stated above are depicted in Figure 1.1.

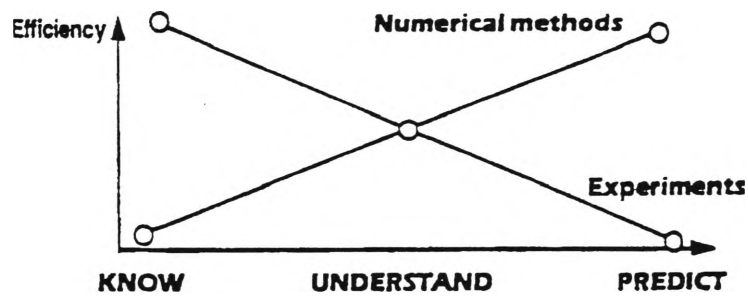


Figure 1.1: Comparison of Numerical Methods and Experiments [Migeot93].

The design, testing and debugging of physical systems by use of experimental observations is always an expensive process because it is very difficult to predict, necessitating manufacture of newer trial systems for subsequent observations. Instead, it is efficient to simulate the model of the process on a computer, permitting 'costly' mistakes to be made much faster, but without the cost (and possible damage to human life), resulting in overall savings in time and money. In addition, various parameters in

the model can be 'tweaked' to predict the optimal physical model that will satisfy all design criteria.

It seems most appropriate to integrate both of the above methods; to combine accurate physical measurements with computer-aided design and numerical modelling methods. This potentially will give us the best of both worlds; the observations taken providing us with useful real-life data so that we can increase our knowledge quickly, while the numerical methods help us to predict optimal design solutions much faster.

1.3.6 Modelling Methods

Acoustic problems require one to be able to model the geometry of the problem accurately, and to provide various kinds of boundary conditions like damping and admittance, fluid-structure interaction and coupling, and radiation to infinity. Besides, it should be possible to simulate different materials.

Hence, we examine modelling methods like the FEM, the BEM and their coupled approaches to see if they are suitable for the problem. The specific derivations required for acoustics are reproduced in Chapter 4. We note that the latter will tend to be biased by the views of Numerical Integration Technologies (NIT), who have implemented SYSNOISE, a package that provides an acoustic implementation of both FEM and BEM.

We further discuss the advantages and disadvantages of the various methods and the criteria required to select a method for solving a given problem.

1.3.7 'Element Method' Software?

With so many commercial 'Element' Method packages available, is it really necessary to develop one of our own?. Chapter 5 discusses the various elements used in formulating a problem and gives the rationalisation and justification for our views.

Chapter 6 then discusses the software design details of our package FAME (**F**inite element **A**nalysis in **M**odula-2 for **E**cholocation) and considers possible hooks for future enhancements and upgradation.

The FEM is a numerical analytical method for solving problems and the traditional approach would have been to use time-honoured FORTRAN to code it. However, we have implemented it in Modula-2 (M2) and Chapter 7 discusses the various language features required to solve numerical problems, and evaluates whether M2 is apt for the task. It also summarises debugging and porting problems, and the results obtained so far.

1.3.8 Is it worthwhile?

Gain of knowledge for immediate practical application and commercial exploitation is preferable to the pure gain of knowledge for its own sake. The development of better (more sensitive) ultrasonic transducers will open up applications in air that are not achievable with current technology.

For example, the ability to recognise an object's shape from its echoes could permit an 'intelligent' mobile robot to operate in environments that are dangerous to humans. Of course, in this respect, the machine's cost could be an inhibitory factor, but that is a different issue!

Object recognition could also be used to design a 'seeing' eye for blind people (assuming that their hearing is not impaired!) It could be in the form of a head-phone, thus freeing up the person's hands. An ultrasonic pulse/chirp could be emitted and the echoes processed and translated into audible frequencies, one for each of 4 or 8 directions. Loudness would indicate nearness.

Once the skeleton of the software has been developed, it provides a structure which future researchers can utilise to develop more complex models, such as those involving transducers and their interaction with the environment. Chapter 8 suggests future enhancements and summarises the results of this research.

Chapter 2 - Bats and Echolocation

2.1 Introduction

In this chapter, we will take an in depth view of hearing peculiarities in bats. We will then go on to study their specialised development and functionality in the particular case of the moustached bat and hence consider the relevance of developing mathematical models to mimic the performance of bats.

2.2 The Mechanism of Hearing

The fundamental structure of the bat's ear can be broken up into 3 parts:- the external ear, the middle ear and the inner ear. The external ear serves a dual purpose. Firstly, it is a protection to the ear drum and the inner ear. Secondly, it behaves as a resonator and amplifies and focuses the sound energy of certain frequencies that are critical to the life cycle of the organism. The tympanic membrane or eardrum lies at or near the body surface. Sound waves impinge on it and are transmitted through to the fluid filled chambers of the inner ear (cochlea) via the bony structures of the middle ear (Figure 2.1). When sound crosses the interface between two different media, there is usually loss of sound energy if one of the media is liquid while the other is gaseous. The middle ear amplifies the sound signal and ensures that information content is not lost in the process of transmission of sound to the fluid of the inner ear.

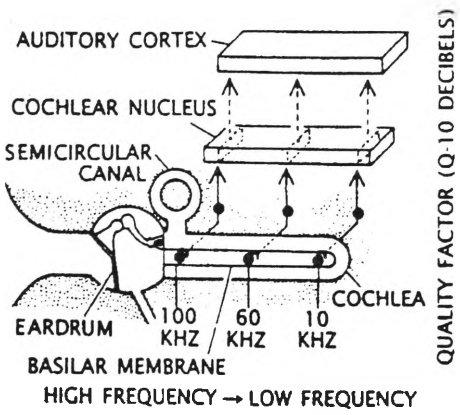


Figure 2.1: Block Diagram of the Bat's ear [Suga90].

Within the cochlea is the basilar membrane, which is vibrated at the outer part by high frequencies, and at the inner part by lower frequencies (Figure 2.1). Frequency coding in the cochlea is done via two mechanisms. At low frequencies (up to 3 KHz), *the periodicity principle* operates. Nerve fibres fire in phase with the acoustic stimulus and directly reflect its periodicity. At middle to high frequencies, *the place principle* operates. Here the acoustic frequency is directly linked to a position in the cochlea, and the presence of the given stimulus activates the nerve fibre present at that location.

The mechanism of sound localisation is quite tricky. Heads act as a sound shadow at certain frequencies. Hence, the sound is more intense in the ear nearer the source. In addition, there will be a slight delay in the arrival of sound in the far ear. These two differences, one in loudness and the other in arrival time at the two ears, function as cues to enable localisation of the position of the sound. The techniques involved require and attain a high degree of precision.

2.3 Bats and Echolocation - An Introduction

Bats are nocturnal mammals and have to rely almost exclusively on their sense of hearing to derive any information about their environment. As a result they have evolved the system of echolocation to facilitate them in navigation and in locating prey.

Echolocation is a means of imaging the environment by the vocal emission of sound pulses which are reflected from surrounding objects.

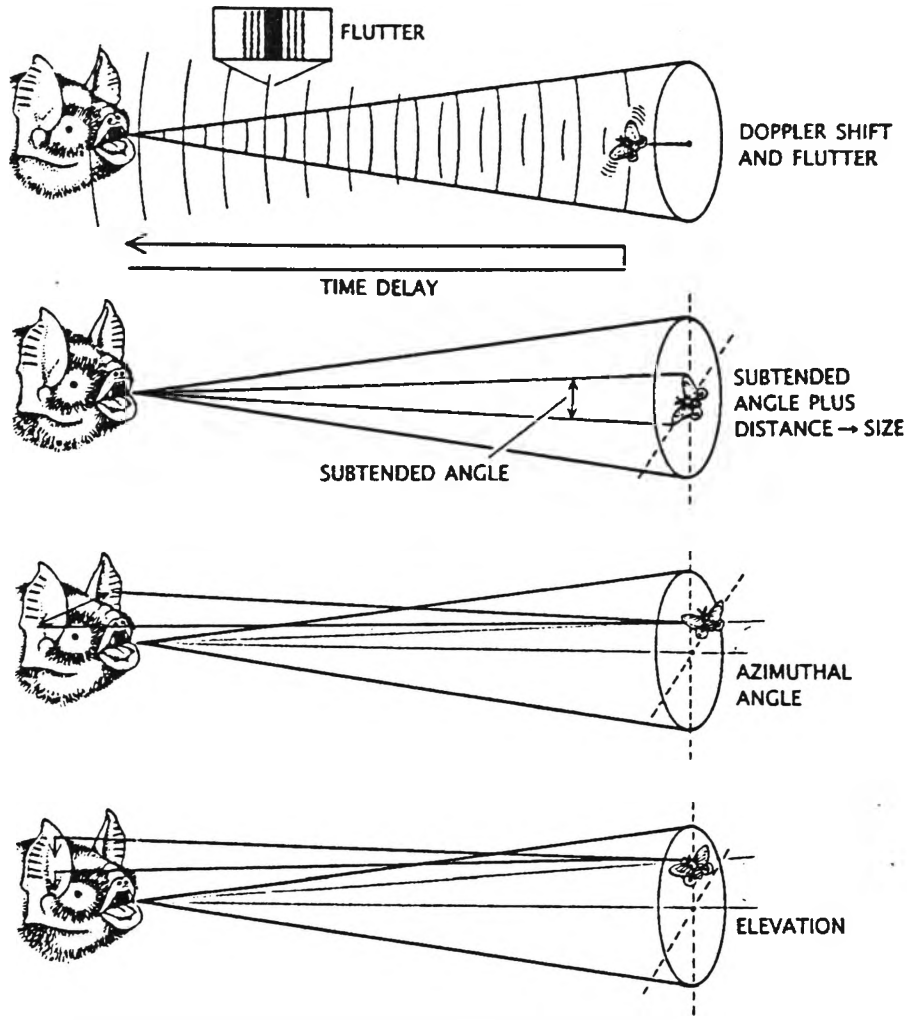


Figure 2.2: Information obtained by bats through echolocation [Suga90].

The bats process the echoes and manage to extract a wealth of information from them (Figure 2.2). The time delay of the echo indicates the distance of the target. The amplitude of the echo, combined with the time delay, indicates the size of the target. The amplitudes of the component frequencies correspond to the size of various features of the target. Differences between the ears in intensity and arrival time of the sound echoes give the azimuth (angular deviation in the horizontal plane) of the target, while the interference pattern of sound waves reflected in the structure of the outer ear gives the elevation (angular deviation in the vertical plane). Doppler shifts (changes in the

frequency of the echo relative to the original signal) inform about the relative velocity of the flying insect and its wing beat.

Leaves and ground and other stationary objects may produce stronger echoes than smaller insects and prey. These misleading signals are known as echo clutter. Bats solve this problem by use of fluttering target detection, changing echo colours or by listening to prey generated noises [Neuweiler89].

2.3.1 Pulse Types used by Bats

The Bat calls can be classified into three main types; constant frequency (CF), frequency modulated (FM) and combined CF/FM (Figure 2.3). CF pulses consist of a single tone. FM pulses start at a high frequency and sweep downwards to a lower frequency passing through all intermediate frequencies. They sound like chirps. CF/FM pulses are a long tone followed by a downward chirp. Tones may usually also include overtones (multiples of the fundamental frequency).

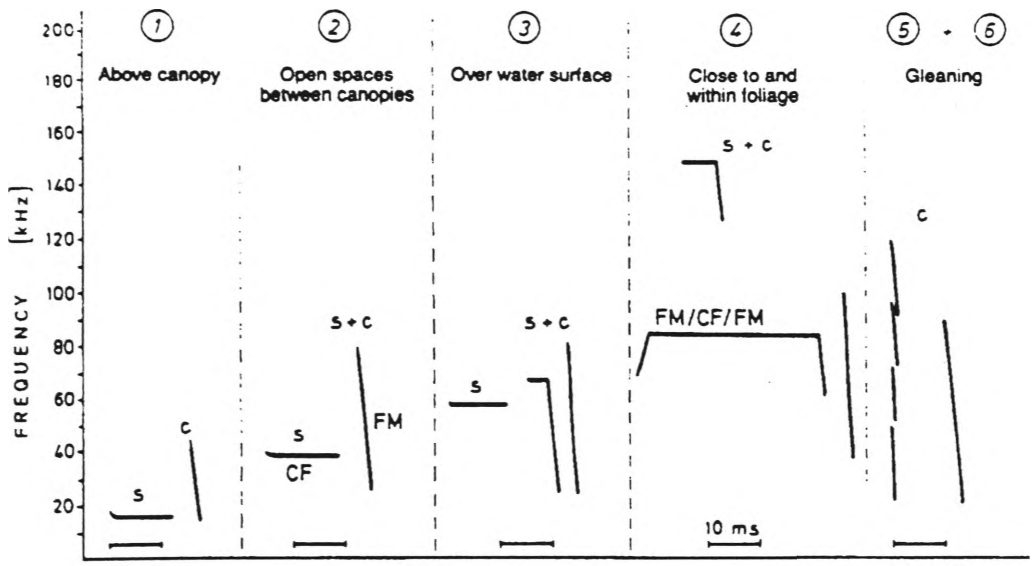


Figure 2.3: Echolocation signals specific to different foraging areas [Neuweiler89].

These pulse types are strongly dependent on the specific foraging areas in which the bat feeds [Neuweiler89]. Those species foraging high above vegetation and in the open spaces between vegetation emit pure tones while searching for insects and

then switch to FM sounds when actually hunting prey (Figure 2.3). Species hunting around and within dense foliage usually use a CF/FM signal. Species searching for prey on leaves and on the ground rely exclusively on FM signals to hunt.

2.3.2 Efficacy of the Different Pulse Types

Sound waves are reflected by an object if their wavelengths are shorter than the object's dimensions. Lower frequencies will only give an outline of the entire object with no details of its configuration or texture. Hence, high frequencies and shorter wavelengths are ideal for detecting echoes from insects or other small objects.

Unfortunately, high frequencies propagate ~~for only~~ short distances in air before significantly being attenuated. So high frequency signals must be emitted at very high pressure levels if they are to be transmitted over long distances.

To solve the above problems, certain bat species control the energy in each harmonic they emit. If the target is far, they amplify the lower harmonics which are less attenuated in air. If the target is near, they amplify the higher harmonics to obtain finer details of the target.

CF pulses are ideal for detecting Doppler shifts and targets larger than the wavelength of the signal. However, they can not locate a target precisely nor provide fine details. FM pulses can provide fine details about the target. They also contain more temporal information and are used to compute echo delays and distance to the target.

As they close in on the prey, FM bats (like the little brown bat) increase the rate of pulse emission and simultaneously reduce the duration of the emitted pulse. This serves to obtain finer details of the prey and also to track it more accurately.

2.4 The Moustached Bat

The moustached bat is adept in the art of fluttering target detection. It easily detects the ripples from insect wings against echoes of stationary backgrounds. Nobuo Suga [90] has done an in depth study of its auditory system. The following is a summary of his work.

2.4.1 Detection of Acoustic Signals

The moustached bat is a CF/FM type emitter. At rest, it emits a fundamental tone (CF1) of 30.5 KHz with three higher harmonics. The resting frequency of the second harmonic (CF2) is 61 KHz while that of the third harmonic (CF3) is 92 KHz.

If the bat detects an echo Doppler-shifted to 63 KHz, it reduces its emitted frequency by about 1.8 KHz, so that the echoes are stabilised at around 61.2 KHz (its reference frequency). This is known as Doppler Shift Compensation (DSC).

Nature has enabled these bats to be able to detect very small frequency shifts (0.01%) near the reference frequency (within the range 61-61.5 KHz). In addition, the echo of the emitted frequency is masked out due to high insensitivity of the neurons of the inner ear to frequencies around 59 KHz.

The neurons of the basilar membrane of the inner ear are sharply tuned to single frequencies within the reference frequency range and are able to detect the signal even if it is embedded in background noise.

2.4.2 Processing of Acoustic Signals

Subsequent to coding of the acoustic signal into nerve signals, further processing is done in the central auditory cortex, in which function specific regions are present.

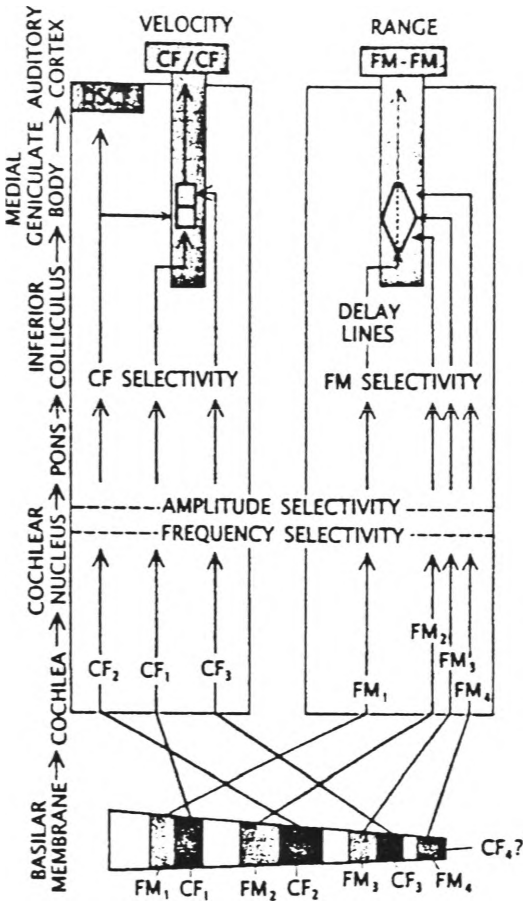


Figure 2.4: Parallel Processing of the various CF and FM harmonics to obtain the relevant echo information [Suga90].

2.4.2.1 The DSCF Region

This region is responsive to the frequencies and the amplitudes of the echoes (Figure 2.4). It consists of a neuron matrix, where each neuron is tuned to a specific frequency (around CF2) and a specific amplitude. Consequently, small variations around the reference frequency can be accurately detected. This region is thought to be responsible for the precision of DSC. Any damage to it results in much longer times to perform DSC and reduces performance.

2.4.2.2 The CF/CF Region

This region is responsive to the frequency differences between the pulses and the echoes. This type of response has been observed in two areas. The first has a

neuron matrix of CF1 versus CF2 while the second has a neuron matrix of CF1 versus CF3. Each location in the matrix specifies a particular relative target velocity, varying from minus 2 to plus 9 m/s. Velocities from 0 to 4 m/s have been emphasised, as these speeds arise during critical activities. A neuron here can become sensitive to a paired signal 6,300 times weaker than the unpaired signal alone.

2.4.2.3 The FM/FM Region

This region is sensitive to the time intervals between the pulses and the echoes. It comprises a matrix of FM pulses and echoes. There are separate areas for emitted pulses FM1 versus the delayed echoes FM2, FM3, FM4 respectively. Each neuron here is specifically tuned to a particular pulse-echo delay and most also prefer a particular echo amplitude. This translates to a target at a particular distance having a specific size. Delays ranging from 0.4 to 18 milliseconds can be sensed and this corresponds to distances ranging from 70 to 3100 mm with a resolution of about 10 mm. A neuron here can become sensitive to a paired signal 28,000 times weaker than either signal alone.

2.4.3 Need for the First Harmonic?

The first harmonic is the weakest component of the pulse, containing less than 1% of the total energy in the pulse. Hence it cannot be heard by any of the other bats nearby. However, it is conducted to the ears of the emitting bat through its own body tissues. It thus acts as a signature and helps in identifying all the other vocal emissions of the bat. It also provides a reference frequency so that it can filter and reject the sounds of all the other bats.

2.5 Modelling of the Vocal and Audio Systems of Bats

How do bats manage to emit such complicated frequencies?

Roderick Suthers [88] describes developments in the laryngeal physiology and vocal tract acoustics of three families of echolocating bats. From his description, it is clear that there is no generalised vocal tract model that will apply to all species of echolocating bats.

In spite of this limitation, researchers continue to develop more sophisticated mathematical models of the acoustic system of bats.

Patrick Flandrin [88] used time-frequency distributions for signal analysis and signal processing. Signal analysis helps describe the time varying portion of bat signals, while signal processing helps formulate receiver models for bats via a time-frequency modulation.

Richard Altes [88] conducted behavioural experiments to examine time-frequency signal representation. Below is a quote from his conclusions.

> "Recent behavioural experiments seem to indicate that detectors designed for random signals are the most likely candidates for echolocation. Random echoes are obtained when nonrandom (or random) signals are passed through filters with randomly time-varying coefficients. Such filters can be used to model scattering of sound from a fluttering insect."

2.6 Conclusion

Echolocation is a biological marvel. It consists of a customised closed loop control system. This customization occurs at both the macro and the micro levels. At the macro level, it operates on a regional basis, providing each different species of bats

vocal and audio characteristics to suit their hunting habitat and habits. At the micro level, it operates within a species, 'voice-printing' each bat with its individual vocal signature.

The 'how' of a bat's acoustic system is known, but the 'why' is still an unknown factor [Oxford64]. In other words, why DOES the bat's acoustic system work? What are the individual components that comprise the entire control system? How do they link together? These are relevant questions to which there are no answers at present!

Carver A. Mead [91] is attempting to throw some 'sound' on the situation by designing his artificial cochlea.

I will evaluate the FEM and other 'Element Methods' to see whether, through their use, we can gain any new insights into this phenomenon.

Chapter 3 - Ultrasonic Transducers

3.1 Introduction

Ultrasonic sensors have been in use for many years in mediums other than air. They have been used extensively for underwater purposes such as exploration, range measurement and for the study of dolphins, whales and bats. In addition, in the medical field, they have been used for non-invasive diagnosis of various ailments.

More recently, improved procedures have been developed that permit a larger energy transfer from an ultrasonic transmitter to air. This has led to a spurt in research relating to the use of ultrasonic devices in two dimensional (2-D) and three dimensional (3-D) mapping of the environment using audio 'images' [McKerrow93] and in the related study of bats and echolocation.

Other regions of interest include determining parameters such as position, distance, thickness, slope and orientation.

The common underlying theme appears to be to apply the knowledge gained to improve the ability of 'semi-intelligent' machines to sense and identify their environment and subsequently interact with it [RoboScrub93].

3.2 Basic Detection Techniques

All ultrasonic systems transmit high frequency sounds (>20 KHz) through the air to the receiver which converts the acoustic energy into an electrical signal.

The system can use either one or two transducers. In the former case, duplexing ensures that the single transducer is used for both transmitting and receiving. In the latter, one is used for transmitting and the other for receiving.

Duplexing is used quite often because it uses only one transducer, hence saving cost and space, and ensures that both the transmitter and the receiver are pointed along the same acoustic axis, avoiding parallax errors. Its major disadvantage is that it creates a blind zone. This is a region close to the transducer in which no object can be detected. Duplexing is usually used for the detection of contours and small targets.

On the other hand, the use of a two transducer system eliminates the blind zone, at the cost of parallax. It also permits the transducer to be optimised for efficiency. Transmitters are designed for maximum mechanical power transfer while receivers are optimised for maximum voltage sensitivity [Lamancusa88].

There are four main types of systems [Monchaud87].

1) Continuous wave (CW) ultrasonic systems send out a signal continuously. Echoes are continually picked up from the surrounding objects. An amplitude sensitive detector responds to changes in the echo signal magnitude caused by a moving target.

2) In the impulse echo mode, the transmitter is pulsed and the time interval between the transmitted pulse and the received echo is used to determine the range. To aid in linking the echo to its parent pulse, usually only one pulse is in the air at a given instant.

3) Sophisticated ultrasonic systems use frequency modulated, continuous wave (FM/CW) chirps to measure both distance and relative velocity.

4) A system developed as a blind aid [Kay86] uses a continuously transmitted frequency modulated (CTFM) signal to permit continuous sensing in air.

3.3 The Ideal Transducer

Before we examine the two main types of ultrasonic transducers, we require to have some reference model. Hence, it is instructive to consider an ideal transducer as applied to ultrasonics.

A transducer may be defined as any device that is capable of converting some form of energy to another. Ideally, it should be [Hunt82]:-

1) linear, where the output variables are linear functions of the input variables,

2) passive, where all the energy delivered to the output load is only obtained from the input source, and

3) reversible, where energy can be converted in either direction.

One further requirement is,

4) efficiency, where all the power should be transferred from input to output, resulting in 100% efficiency. This implies impedance matching between the transducer and air.

For ultrasonic transducers, the radiated pressure field, p, must satisfy the Sommerfeld radiation condition at a large distance, r, from the body.

ie. $$\lim_{r\to\infty} r^{\alpha}\left(\frac{\partial p}{\partial r}+ikp\right)=0 \qquad \text{.. .. ..} \qquad 3.1$$

where $\alpha = 1$ for 3-D and $\alpha = \frac{1}{2}$ for 2-D.

This condition implies that the infinite regions do not contribute to the radiation field, and hence require no special attention.

3.4 Flat Circular Piston in an Infinite Baffle

Consider an infinite plane that is at rest in all regions except for a small circular region, of radius a, that vibrates harmonically, with velocity v_0.

The total sound pressure at some arbitrary point x can be given by [Kuttruff91]

$$p(x,t) = v_0 Z_0\left(e^{-ikx} - e^{-ik\sqrt{x^2+a^2}}\right)e^{i\omega t} \qquad \text{.. .. ..} \qquad 3.2$$

where

Z_0 is the characteristic impedance of the medium

ω is the angular frequency of the oscillation

$k = \frac{\omega}{c} = \frac{2\pi f}{c}$ is the wave number

where f is the frequency

c is the speed of sound in the medium

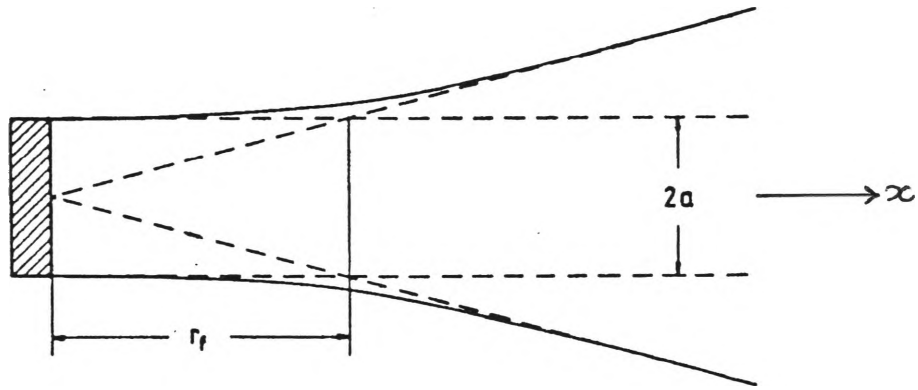


Figure 3.1: Approximate extension of the sound field in front of a circular piston with $ka >> 1$ [Kuttruff91].

As shown in Figure 3.1 [Kuttruff91], equation 3.2 describes the sound intensity in front of the piston as a 'near field', which is approximately a circular tube of radius a and length r_f, beyond which exists the 'far field', which is a conical region, whose top lies at the piston's centre and whose radius is a at r_f, extending to infinity.

In reality, the equation does result in a slightly more complex near field, whose outer envelope can be approximated as specified above, and exhibits the inverse square law in the far field [Bjorno86].

The directivity or radiation pattern of the circular piston varies with ka, which equals the circumference of the piston divided by the wavelength. Typical plots are shown in Figure 3.2 [Kuttruff91].

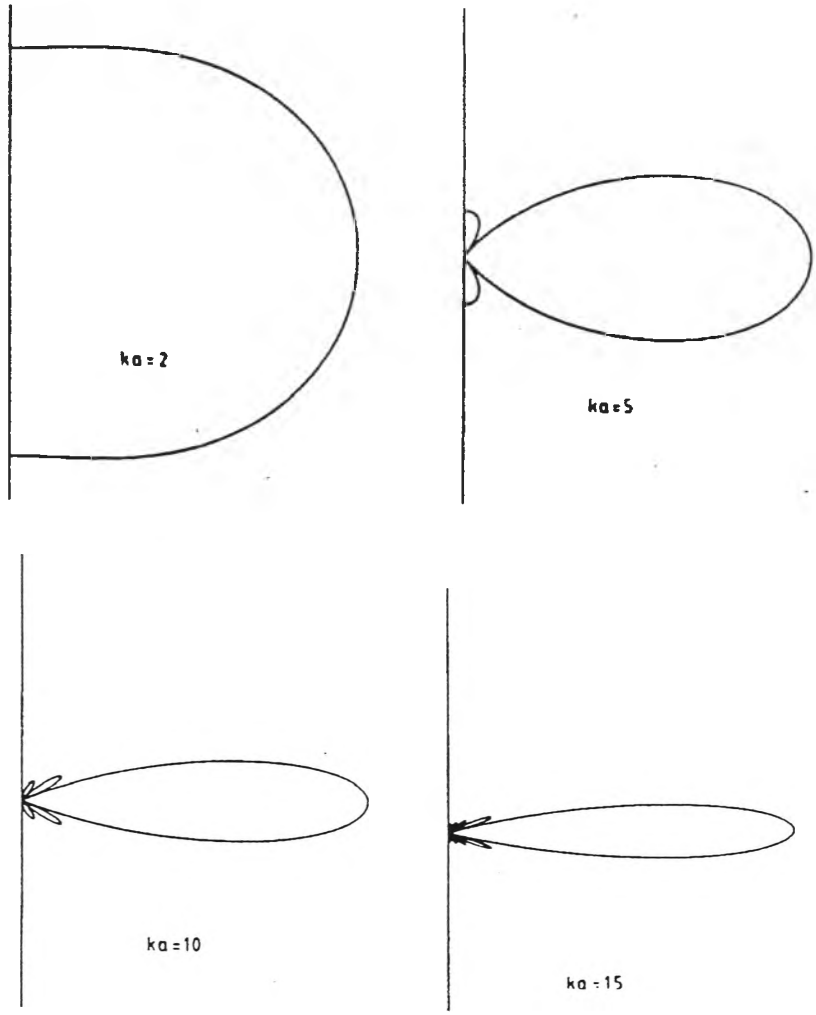


Figure 3.2: Directional pattern of the sound pressure amplitude of a circular piston for various values of *ka* [Kuttruff91].

As is obvious, for small ka (ka=2), the piston acts like a point source with a circular wavefront. But as ka increases, the piston acts as a multiple point source and the ensuing interaction between the individual wavefronts results in maxima and minima, creating one main lobe and 2 or more side lobes. Also obvious, beamwidth decreases with increasing ka.

For maximum power transfer, we require that the impedance of the piston be matched to that of the medium. For efficient transfer of sound energy to air, we need the entire transducer surface to oscillate uniformly with equal amplitude and phase. Hence the forcing function should be capable of extracting such performance. If this is not directly possible, then some additional means for coupling should be used, such as a mechanical membrane [Babic91] or acoustic impedance matching materials [Magori87]

Fluid loading only plays an important part when the effects of the fluid on the structure cannot be neglected. Usually fluid loading by air can be neglected due to its small resistance. However, for very light structures, it becomes necessary to consider its effects. eg. in loudspeaker analysis, the speaker membrane has very little stiffness and is very light. When water is the medium, the coupling effects cannot be isolated, due to the large mass of water.

3.5 The Polaroid (Electrostatic) Transducer

The principle of the electrostatic transducer has been used to develop the polaroid transducer which performs precise automatic focusing of the camera lens of the Polaroid SX-70 Sonar Camera.

The transducer assembly is shown in Figure 3.3 [Biber80]. It consists of a plastic (Kapton) foil plated with gold on the front side and stretched over an aluminium backplate, which has a series of concentric grooves. The backplate and the foil together represent an electrical capacitor.

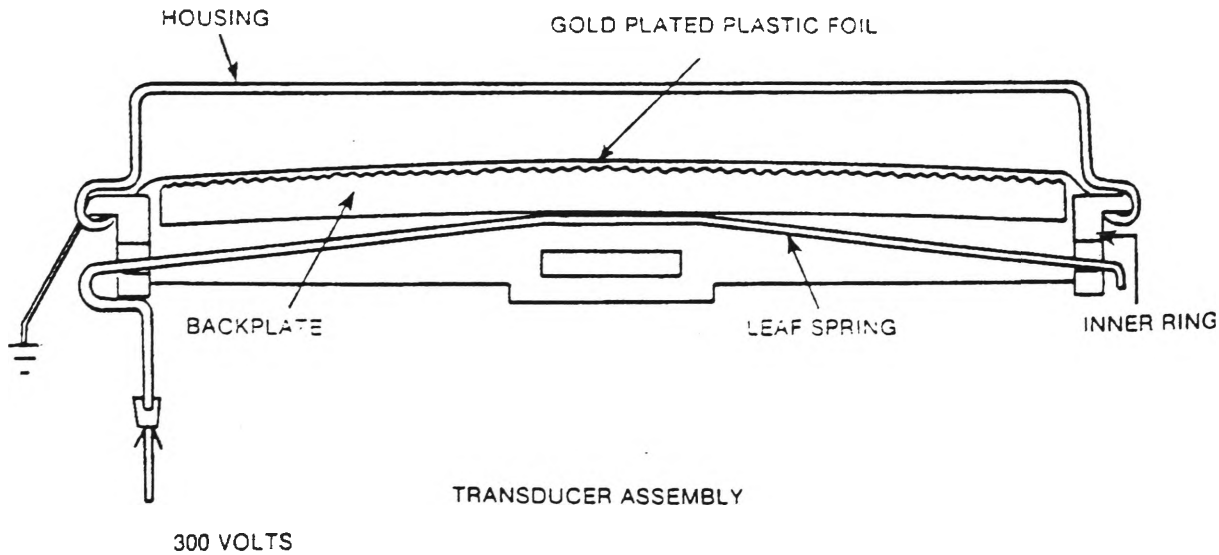


Figure 3.3: Block diagram of Polaroid Electrostatic Transducer [Biber80].

The operating principle of this transducer is explained below [Biber80]. A steady bias voltage (300V) is applied across the backplate and the foil, generating an attractive force between them. An alternating voltage is superimposed on this bias voltage. It changes the force of attraction in a manner proportional to the alternating voltage and at the same frequency. The foil vibrates in response to this force and hence transmits an acoustic signal into the air.

The force exerted on the diaphragm [Kuhl54], and consequently the sound pressure, is proportional to: $\left(U + \tilde{U}\ sin\ \omega t\right)^2$.

ie. $f_d \propto \left(U + \tilde{U}\ sin\ \omega t\right)^2$ 3.3

where

U is the DC voltage

$\tilde{U}$ is the AC voltage

The device can also be used as a receiver. The acoustic signal causes the foil to vibrate and hence there is a periodic change in capacitance between the foil and the backplate. This varying capacitance is converted to voltage via suitable electronics.

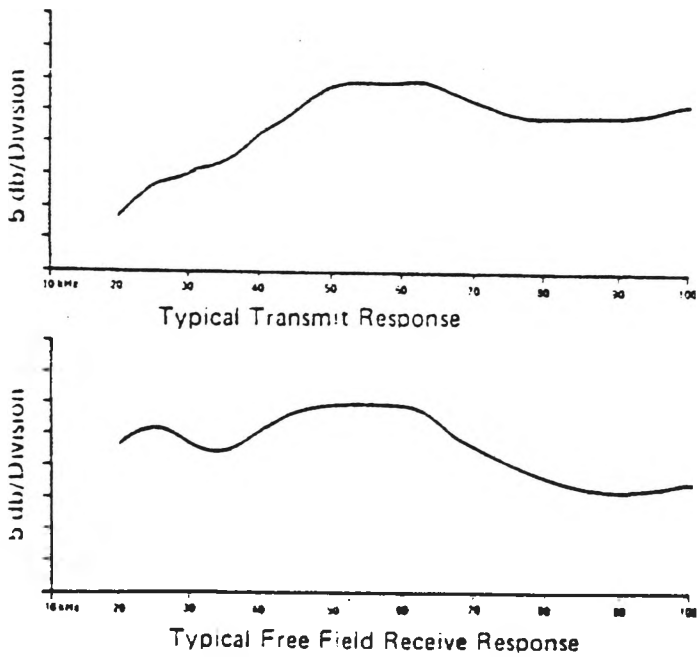


Figure 3.4a: Polaroid Electrostatic Transducer response plots [Polaroid80].

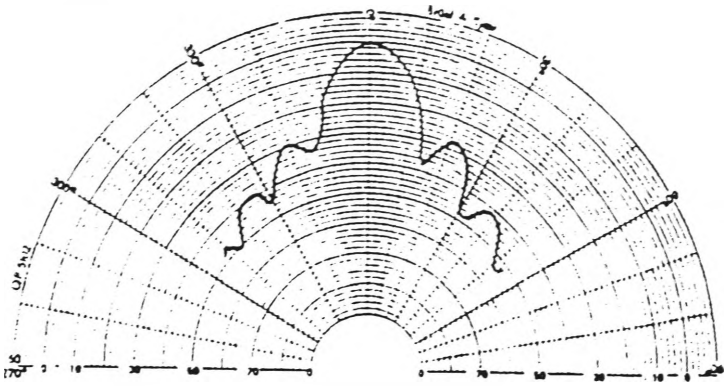


Figure 3.4b: Polaroid Electrostatic Transducer Directivity plot [Polaroid80].

The resulting frequency plot and beam width are given in Figures 3.4a and 3.4b respectively [Polaroid80]. We note that with a diameter of 4.013 cm and a frequency of 50 KHz, ka = 19, which accounts for the high directivity of the beam pattern. Also note the large differences in sensitivity between transmission and reception. Its minimum transmitting and receiving sensitivities at 50 KHz are 107 dB and -45 dB respectively [Polaroid80].

3.6 The Piezoelectric Transducer

3.6.1 The Principle

If a static or dynamic pressure is applied to certain natural crystalline substances like quartz or Rochelle salt, a DC or AC voltage is generated respectively. Conversely, if a voltage is applied to these crystals, they exhibit a mechanical vibration. This relationship between the electrical and mechanical effects is known as the piezoelectric effect.

Certain ceramics, like PZT (Section 3.6.2), require to be poled before they exhibit the piezoelectric effect. Poling is the momentary application of a strong direct current, which converts an isotropic ceramic into an anisotropic piezoceramic.

These piezoelectric substances usually have a natural frequency of vibration. If the applied frequency matches this natural frequency, then the substance will resonate. This frequency selectivity is exploited for sensing applications.

The piezoelectric transducer can either be a transmitter or a receiver or both. If the applied voltage varies at an ultrasonic frequency, the transducer vibrates at the same frequency and transmits these vibrations via a diaphragm to the surrounding air and thus produces ultrasonic waves. As a receiver, the received input frequency generates a proportional AC voltage. This voltage is amplified and processed.

3.6.2 Materials

Ferroelectric materials like barium titanate ($BaTiO_2$) and lead zirconate titanate ($Pb(Zr,Tc)O_3$) (PZT) are in powder form and so are used in piezoceramics.

There are also piezoelectric high polymers like polyvinylidene fluoride (PVDF or PVF2).

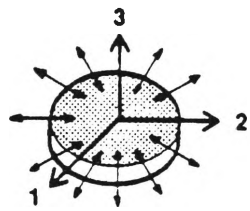
3.6.3 Design is Critical

The design of piezoelectric transducers is critical. The natural frequency of these resonators is inversely proportional to the thickness. However, no matter the shape of the element, a number of natural modes of vibration exist. Hence, all dimensions should be precisely specified to prevent coupling between various modes of vibration when the driving frequency is close to one of the other modal frequencies. Even minor coupling can lead to degradation of the overall transducer's performance. [Smith84] So accurate modelling of piezoelectric transducers must include this cross-coupling.

3.6.4 Modes of Operation

Piezoelectric substances have various modes of operation. In piezoelectric crystals, these are controlled by the placement of electrodes (the direction of the electric field) and the alignment of the crystallographic axes [Berlincourt64]. Piezoceramics additionally require appropriate poling to ensure the best possible alignment of the dipoles.

1. Thin Disc
Radial Mode

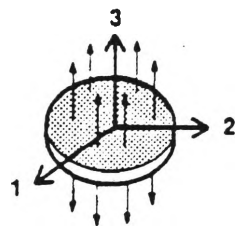


Electroded on flat surfaces.
Poled through thickness.
Frequency constant N = dia X fr

$$\text{Capacitance} = \frac{\text{dia X } \varepsilon r}{5.664 \text{ X th}}$$

$$\varepsilon r = \frac{(5.664)\,(Co)\,(th)}{dia}$$

2. Thin Disc or Plate
Thickness Mode

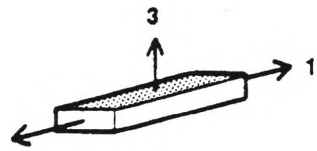


Electroded on flat surfaces.
Poled through thickness.
Frequency constant N = th X fr

$$\text{Capacitance} = \frac{\text{dia X } \varepsilon r}{5.664 \text{ X th}}$$

$$\varepsilon r = \frac{(5.664)\,(Co)\,(th)}{dia}$$

3. Long thin Bar
Length Mode

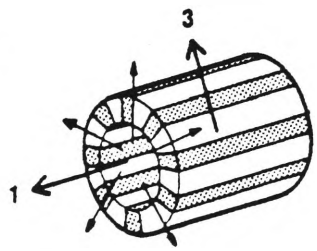


Electroded on shaded surfaces.
Poled through thickness.
Frequency constant N = L X fr

$$\text{Capacitance} = \frac{\varepsilon r \text{ X area}}{4.448 \text{ X th}}$$

$$\varepsilon r = \frac{(4.448)\,(Co)\,(th)}{area}$$

4. Thin wall Tube
Radial (hoop) Mode



Electroded on shaded stripes.
Poled between stripes.
Frequency constant N = mean dia X fr

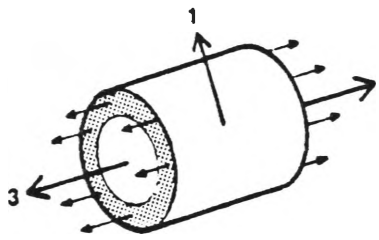
$$\text{Capacitance} = \frac{\varepsilon r \text{ X area}}{4.448 \text{ X th X } .8}$$

$$\varepsilon r = \frac{4.448\,(Co)\,(\text{ave. space between stripe})}{(\text{wall})\,(L)\,(N)}$$

N = number of stripes

Figure 3.5a: Modes of vibration of the Piezoelectric resonator [Seippel83].

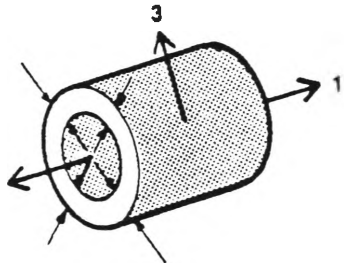
5. Thin wall Tube
Length Mode



Electroded on ends
Poled through length.
Frequency constant N = L X fr

$$\text{Capacitance} = \frac{(OD - ID) \times \varepsilon r}{5.664 \times th}$$

6. Thin wall Tube
Thickness Mode

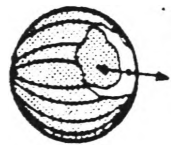


Electroded on curved surfaces.
Poled through wall thickness.
Frequency constant N = th X fr

$$\text{Capacitance} = \frac{\varepsilon r \times L}{1.628 \times (\log 10 \frac{OD}{ID})}$$

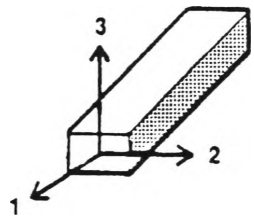
$$\varepsilon r = \frac{(1.628)\,(Co)\,(\log 10 \frac{OD}{ID})}{L}$$

7. Thin wall Sphere
Radial Mode



Electroded on curved surfaces.
Poled through wall thickness.
Frequency constant N_t = mean dia X fr

8. Shear Plate



Electroded on shaded surfaces.
Poled through thickness. (3 axis)
Frequency constant N_t = th X fr

$$\text{Capacitance} = \frac{\varepsilon r \times \text{area}}{4.448 \times th}$$

$$\varepsilon r = \frac{(4.448)\,(Co)\,(th)}{\text{area}}$$

Figure 3.5b: Modes of vibration of the Piezoelectric resonator [Seippel83].

The various modes of vibration achieved with different shapes and poling characteristics are summarised in Figures 3.5a and 3.5b.

Let us consider some simple modes applied to the quartz crystal in Figure 3.6. Assuming the international convention for crystallographic axes is applied, we have the

z-axis as shown. Any one of the three x-axes can be selected (all are equivalent). Then, the y-axis is chosen so that x, y and z-axes form a right-handed co-ordinate system.

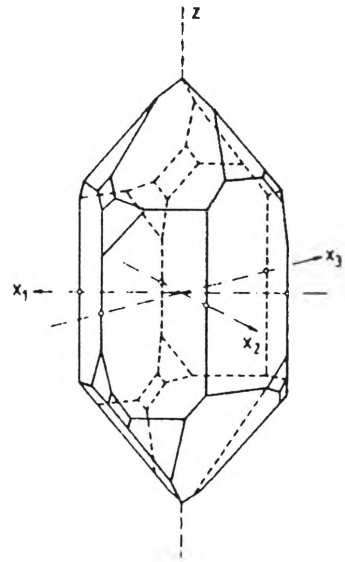


Figure 3.6: Quartz crystal. [Kuttruff91].

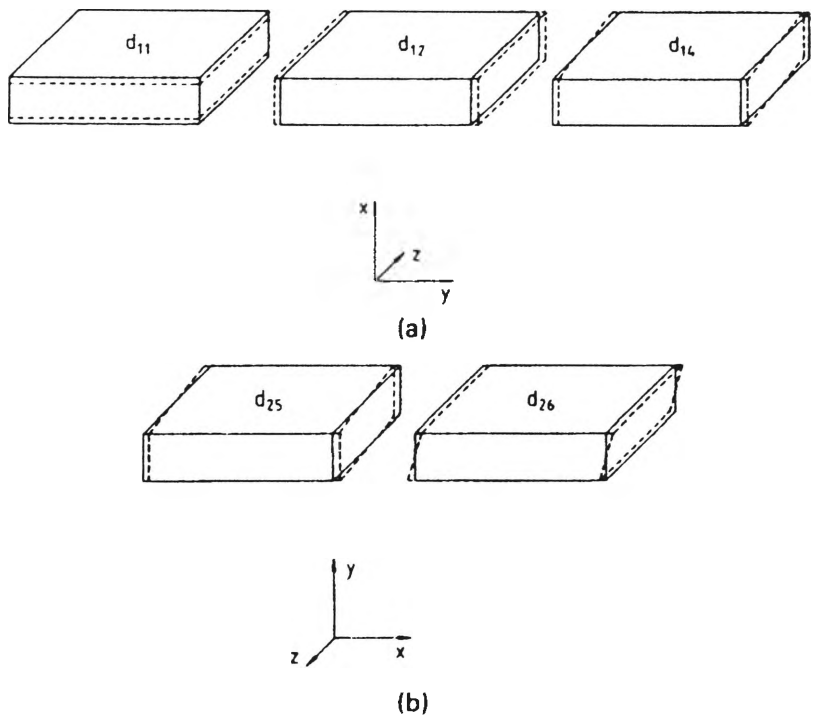


Figure 3.7: Relation between the piezoelectric moduli and the deformation of a quartz plate; (a) x-cut, (b) y-cut [Kuttruff91].

Now, the x-cut is cut out of a quartz crystal perpendicular to its crystallographic x-axis (Figure 3.7). The electric field is applied parallel to the x-axis. This results in a

tensile stress σ_{xx} that tends to increase/decrease the thickness of the plate (longitudinal effect) and a corresponding 'opposite' tensile stress σ_{yy} that tends to decrease/increase a lateral dimension (transverse effect). A shear stress σ_{zy} (associated with shear deformation) is also produced.

Similarly, the y-cut (perpendicular to the y-axis) has two shear stresses, σ_{xy} and σ_{zx}, while the z-cut does not exhibit piezoelectricity as all the corresponding constants are zero.

In similar fashion, all other piezo-resonators have different modes of operation. The interested reader is referred to Seippel [83] or Berlincourt [64]. Figure 3.5 illustrates some of the principal shapes and direction of polarisation and modes of vibration of the more commonly used driving elements.

3.6.5 Bimorph sensors

The change in dimensions of a disc of piezoelectric material under voltages below the breakdown level is small. The discs are also quite stiff and this limits the amplitudes of their vibrations. Bimorph structures are used to overcome these limitations (Figure 3.8).

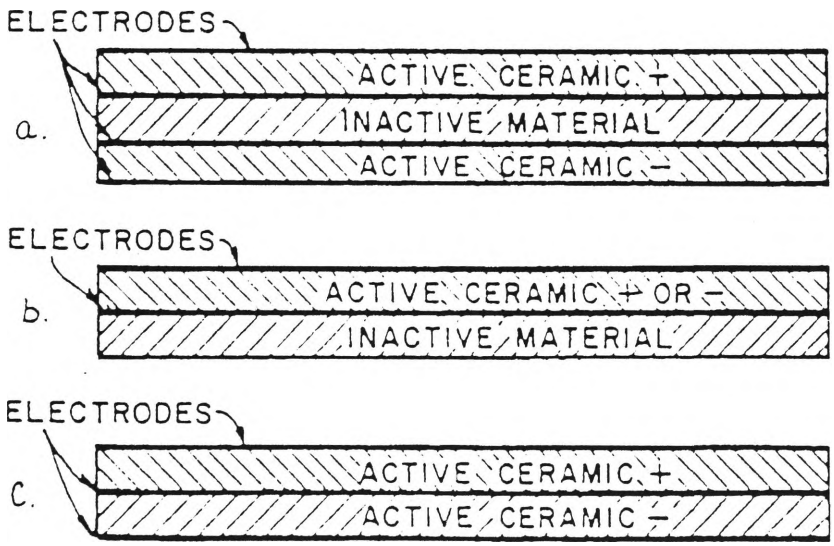


Figure 3.8: Bimorph composite structures for ultrasonic sensors; (a) trilaminar, (b) bilaminar and (c) bimorph [Bjorno86].

Structure (a) consists of two similar PZT ceramic discs attached to a central metallic disc. Electrodes are connected to both ceramic discs, which are poled in the same sense if the output terminals are to be the central metal disc and either of the outer electrodes. If the output terminals are to be the outer electrodes, then the ceramic discs are poled in the opposite sense. Alternately, a single ceramic disc could be attached to the metallic disc to give structure (b). The central metal disc in (a) may be so thin as to be negligible and this gives rise to structure (c). These composite structures (a), (b) and (c) are known as trilaminar, bilaminar and bimorph structures respectively.

When an excitation voltage is applied to the electrodes of the discs, the piezoelectric ceramic composite flexes in and out in a motion perpendicular to the plane of the disc. This flexural vibration can be transferred to the fluid around the composites. The transfer of acoustic energy may occur either via a coupling structure such as a diaphragm [Babic91] or by direct coupling to the fluid medium [Magori87].

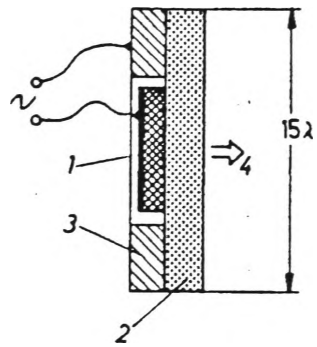


Figure 3.9a: Schematic of RU type high-directivity ultrasonic transducer;
(1) piezoceramic disk, (2) matching element, (3) metal ring, (4) maximum radiation direction.
[Magori87]

This latter phenomenon is exploited in the Siemen's RU-x series of transducers (Figure 3.9a), where the outer periphery of the radial transducer is clamped. The radial resonance mode is preferred because all spurious resonance modes are far separated from it, and, compared to the thickness mode vibrator, it requires less material for a larger surface resonance excitation, other parameters being the same [Magori87]. It requires a special matching element to improve signal transmission to the medium. Figures. 3.9b and 3.9c show its typical motion and high directivity.

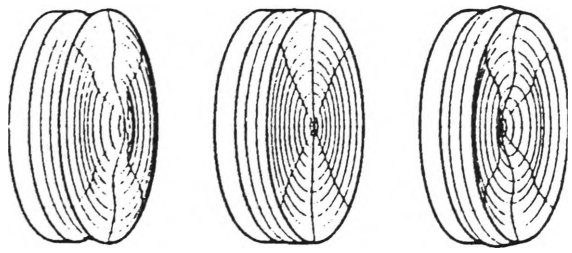


Figure 3.9b: Motion of radiating surface of RU type high-directivity ultrasonic transducer in 3 successive phases [Magori87].

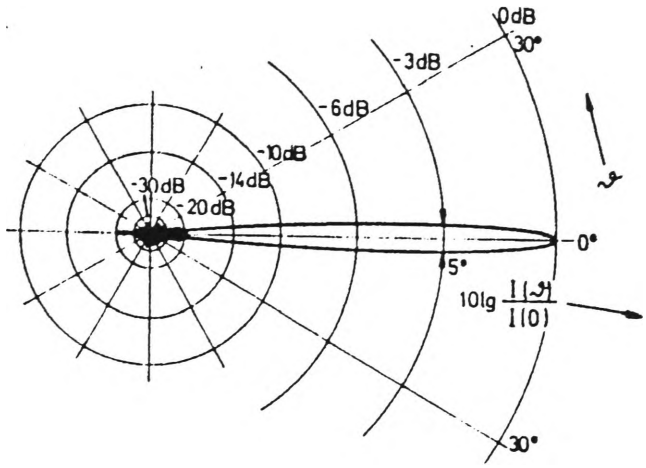


Figure 3.9c: Radiation pattern of RU type high-directivity ultrasonic transducer [Magori87].

3.6.6 PVF2

PVF2 is a piezoelectric transducer available in sheet form. It has excellent linearity and low hysteresis. Due to its thin film format, it can be used as shown in Figure 3.10.

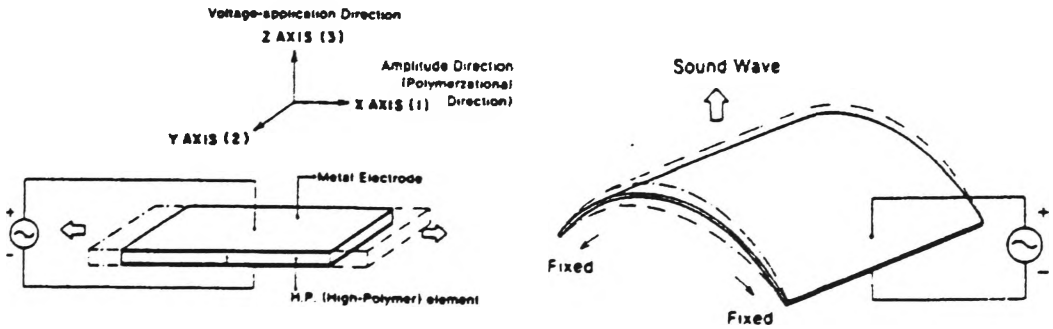


Figure 3.10: Piezoelectric high-polymer and its transmitting mode; left (unclamped) and right (clamped) [Schoenwald87].

Referring to the left side of the figure, when a sinusoidal field is applied to the PVF2 sheet along the normal axis (z), the sheet vibrates along the transverse axis (x).

When the sheet is firmly clamped at the two ends (right side of figure), compressional waves (basically, sound) can be excited or detected [Schoenwald87].

A transducer based on this principle has been developed by CSIRO Applied Physics. It has a 12 dB better response than the Polaroid transducer.

3.6.7 Piezoelectric Equations

Piezoelectric materials are intrinsically anisotropic and involve the coupling of elastic and dielectric phenomena.

The coefficients are typically arranged in a 9x9 matrix, where each column describes an independent stress variable (eg. elastic stress component or electric field component) and each row describes a dependent strain variable (eg. geometric strain component or electric dipole component).

The piezoelectric constants are third-order tensors relating second order symmetric tensors (stress or strain) to vectors (electric field or charge density). The elastic constants are fourth-order tensors relating two second-order symmetric tensors. The dielectric constants are second-order tensors relating two vectors.

Hence, for any direction of propagation, there are three possible acoustic waves with mutually perpendicular directions of propagation, and, usually, different velocities. The corresponding physical constants and variables have components along all three axes. These components interact with each other, resulting in tensors.

Let us consider a very simplistic derivation to obtain the basic piezoelectric equations.

3.6.7.1 Simple derivation

Let E, D, T and S correspond to the electric field, electric displacement, shear stress and strain components respectively.

We get four different sets of relations, by selecting any two of the above to be independent variables, and making the others dependent.

In particular, if T and D are functions of S and E, we have:

$$T = T(S,E) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 3.4$$

and $$D = D(S,E) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 3.5$$

By Taylor's expansion for small fields, we get the following constitutive equations:-

$$T = \left(\frac{\partial T}{\partial S}\right)_E S + \left(\frac{\partial T}{\partial E}\right)_S E \quad .. \quad .. \quad .. \quad .. \quad 3.6$$

and $$D = \left(\frac{\partial D}{\partial S}\right)_E S + \left(\frac{\partial D}{\partial E}\right)_S E \quad .. \quad .. \quad .. \quad .. \quad 3.7$$

In matrix representation, equations 3.6 and 3.7 become

$$\{T\} = [c]^E \{S\} - [e]^t \{E\} \quad .. \quad .. \quad .. \quad .. \quad 3.8$$

and $$\{D\} = [e]\{S\} + [\varepsilon]^S \{E\} \quad .. \quad .. \quad .. \quad .. \quad 3.9$$

where

$\{T\}$ is a 6-by-1 stress vector

$\{S\}$ is a 6-by-1 strain vector

$\{D\}$ is a 3-by-1 electric displacement vector

$\{E\}$ is a 3-by-1 electric field vector

$[c]^E$ is 6-by-6 elastic stiffness matrix for constant field E

$[e]$ is 3-by-6 piezoelectric coefficient matrix

$[\varepsilon]^S$ is 3-by-3 dielectric coefficient matrix for constant strain S

We note that $[c]^E$, $[e]$ and $[\varepsilon]^S$ are material property constants. These are affected by various parameters like the type and composition of the materials and the cut used.

The exact equations have been derived from thermodynamic potentials [Berlincourt64], and the interested reader is referred there.

Table 3.1: Material data of the piezoelectric material VIBRIT 420 [Lerch90]

Density: $\rho = 7600 kg/m^3$.

Elastic Stiffness constants:
$$[c]^E = \begin{bmatrix} 14.9 & 10.1 & 9.8 & 0 & 0 & 0 \\ & 14.9 & 9.8 & 0 & 0 & 0 \\ & & 14.3 & 0 & 0 & 0 \\ & & & 2.2 & 0 & 0 \\ & & & & 2.2 & 0 \\ & & & & & 2.4 \end{bmatrix} 10^{10} N/m^2$$

Piezoelectric constants:
$$[e] = \begin{bmatrix} 0 & 0 & 0 & 0 & 11.7 & 0 \\ 0 & 0 & 0 & 11.7 & 0 & 0 \\ -5.4 & -5.4 & 13.5 & 0 & 0 & 0 \end{bmatrix} As/m^2$$

Dielectric constants:
$$[\varepsilon]^S = \begin{bmatrix} 8.0 & 0 & 0 \\ 0 & 8.0 & 0 \\ 0 & 0 & 7.2 \end{bmatrix} 10^{-9} As/Vm$$

Chapter 4 - Element Methods

4.1 The Finite Element Method

> "The Finite Element Method is a computer-aided mathematical technique for obtaining approximate numerical solutions to the abstract equations of calculus that predict the response of physical systems subjected to external influences."
>
> David S. Burnett [Burnett88]

The total problem domain is modelled by subdividing it into regions called finite elements, each of which is fully specified by a set of functions that describe all the variables in that region. These specially chosen functions ensure continuity of behaviour through the entire domain, and with their help an approximate solution is interpolated. A description of the intricacies of the FEM for the solution of 1-D general problems is detailed in Appendix E.

The FEM can be used to solve:-

Boundary Value problems (BDVP), eg. equilibrium or steady state problems

Eigenproblems (EIGP), eg. resonance or stability problems

Initial Boundary Value problems (IBVP), eg. vibration or diffusion problems

Dynamics problems (DYNP), eg. wave propagation problems.

The partial differential equations (PDEs) for the above types of problems are listed in Appendix C.

4.1.1 Derivation of Acoustics FEM

When solving acoustic problems, the Helmholtz equation is utilised most frequently. So we state the necessary boundary conditions (BCs) and derive the equations for acoustic FEM.

In some arbitrary domain V, we have the Helmholtz equation (Appendix B)

$$\nabla^2 p + k^2 p = 0 \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.1$$

The surface of the domain can be split into 3 regions, S_1, S_2, and S_3, on each of which a different boundary condition (BC) can be applied.

$$S_1 : p = \bar{p}, \text{ (a specified pressure)} \quad .. \quad .. \quad .. \quad .. \quad 4.2$$

$$S_2 : v_n = \bar{v}_n \Leftrightarrow \partial_n p = -i\rho\omega\bar{v}_n, \text{ (a specified normal velocity)} \quad 4.3$$

$$S_3 : A_n = \frac{v_n}{p} \Leftrightarrow \partial_n p = -i\rho\omega A_n p, \text{ (a normal admittance)} \quad 4.4$$

Now, note that at all points on the surface, one and only one of the above conditions must hold.

In a FEM calculation, S_1 is usually satisfied exactly by the approximate solution p^*, and hence generates no residual. Substitution of p^* in equations 4.1, 4.3 and 4.4 would usually result in some remainder left over due to the inexact nature of p^*. This gives us the corresponding residuals.

$$R_V = \nabla^2 p^* + k^2 p^* \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.5$$

$$R_{S_2} = -\partial_n p^* - i\rho\omega\bar{v}_n \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.6$$

$$R_{S_3} = -\partial_n p^* - i\rho\omega A_n p^* \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.7$$

To obtain the best solution, we would prefer all BCs to be satisfied and hence all residuals (or the sum of their averages over their region of relevance) should be zero. So application of the Method of Weighted Residuals, for some specified weighting function W, gives us

$$\int_V W.R_V dV + \int_{S_2} W.R_{S_2} dS + \int_{S_3} W.R_{S_3} dS = 0 \quad .. \quad .. \quad 4.8$$

Substituting for the residuals, we obtain,

$$\int_V W\left(\nabla^2 p^* + k^2 p^*\right)dV - \int_{S_2} W\left(\partial_n p^* + i\rho\omega\bar{v}_n\right)dS - \int_{S_3} W\left(\partial_n p^* + i\rho\omega A_n p^*\right)dS = 0$$

$$.. \quad .. \quad .. \quad .. \quad .. \quad 4.9$$

Integrating the first term of equation 4.9 by parts gives us

$$\int\left(-\partial_k W.\partial_k p^* + k^2 W p^*\right)dV + \oint_S W.\partial_n p^* dS - \int_{S_2} W\left(\partial_n p^* + i\rho\omega\bar{v}_n\right)dS - \int_{S_3} W\left(\partial_n p^* + i\rho\omega A_n p^*\right)dS = 0$$

$$.. \quad .. \quad .. \quad .. \quad .. \quad 4.10$$

The second term of equation 4.10 is a positive integral over the entire surface, S. Its positive value over S_2 and S_3 would cancel out with the corresponding negative integrals in the third and fourth terms, resulting in

$$\int \left(-\partial_k W.\partial_k p^* + k^2 W p^*\right) dV + \int_{S_1} W.\partial_n p^* dS$$
$$-i\rho\omega \int_{S_2} W.\bar{v}_n dS - i\rho\omega \int_{S_3} W.A_n p^* dS = 0$$

.. .. 4.11

Since we have fixed pressure conditions on the surface, we select W such that $W = 0$ on S_1. This eliminates the integral over S_1 in equation 4.11, giving us

$$\int \left(-\partial_k W.\partial_k p^* + k^2 W p^*\right) dV - i\rho\omega \int_{S_2} W.\bar{v}_n dS - i\rho\omega \int_{S_3} W.A_n p^* dS = 0$$

.. .. 4.12

Define the shape function to be

$$p^* = \sum_i p_i N_i \text{ (away from } S_1) \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.13$$

Using the Galerkin Method, we let the weighting function $W = N_i$, giving us

$$\left(\int \left(-\partial_k N_i.\partial_k N_j + k^2 N_i N_j\right) dV\right).p_j - i\rho\omega \int_{S_2} \bar{v}_n N_i dS$$
$$-\left(i\rho\omega \int_{S_3} A_n N_i N_j dS\right).p_j = 0$$

.. .. 4.14

Collecting terms of p_j on the left hand side (LHS) and taking all other terms to the right hand side (RHS), and writing in matrix notation, we get

$$\left(\left[\int \partial_k N_i . \partial_k N_j dV\right] + i\rho\omega\left[\int_{S_3} A_n N_i N_j dS\right] - \omega^2\left[\int \frac{N_i N_j}{c^2} dV\right]\right).\{p\} = -i\rho\omega\left\{\int_{S_2} \bar{v}_n N_i dS\right\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.15$$

Equation 4.15 has the form

$$\left([K] + i\rho\omega[C] - \omega^2[M]\right)\{p\} = -i\rho\omega\{F\} \quad .. \quad .. \quad 4.16$$

where

$$[K] = \left[k_{ij}\right] = \left[\int \partial_k N_i . \partial_k N_j dV\right] \text{ is 'acoustic stiffness' matrix} \quad .. \quad .. \quad .. \quad .. \quad 4.17$$

$$[C] = \left[c_{ij}\right] = \left[\int_{S_3} A_n N_i N_j dS\right] \text{ is 'acoustic damping' matrix} \quad .. \quad .. \quad .. \quad .. \quad 4.18$$

$$[M] = \left[m_{ij}\right] = \left[\int \frac{N_i N_j}{c^2} dV\right] \text{ is 'acoustic mass' matrix} \quad .. \quad .. \quad .. \quad .. \quad 4.19$$

$$\{F\} = \{f_i\} = \left\{\int_{S_2} \bar{v}_n N_i dS\right\} \text{ is 'acoustic forces' matrix} \quad .. \quad .. \quad .. \quad .. \quad 4.20$$

Equation 4.16 is the standard equation for acoustics problems. It bears a close resemblance to the equations for structural dynamics problems (Appendix C). The corresponding matrix terms have identical meanings while the unknown is pressure.

4.1.2 Solution of Acoustics FEM

There are three possible solution schemes.

1) Harmonic response - direct method

Here we wish to find the unknown pressures at some arbitrary fixed frequency. So we substitute the value of the fixed frequency into 4.16 and hence,

$$\omega = \overline{\omega} \Rightarrow \{p\} = -i\rho\overline{\omega}\left([K] + i\rho\overline{\omega}[C] - \overline{\omega}^2[M]\right)^{-1}\{F\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.21$$

Equation 4.21 is then solved to determine the pressures.

2) Acoustic modes extraction

Here we assume that no forces act on the system and that damping is negligible. Substituting $F = C = 0$ into 4.16, we get 4.22, which is a standard eigenproblem formulation. Hence we get the free oscillating modes of the system.

$$\omega_i, \{\phi_i\} \neq 0 : \left([K] - \overline{\omega}_i^2[M]\right)\{\phi_i\} = 0 \quad .. \quad .. \quad .. \quad 4.22$$

3) Harmonic response - modal superposition

Here we express the unknown pressures as a linear combination of the modal frequencies (obtained by acoustic mode extraction), and solve the resulting

system of equations to determine the participation factor, a_i of each mode ϕ_i towards the final solution. We note that these matrices are all complex.

$$\{p\} = a_i \{\phi_i\} = [\phi]\{a\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.23$$

$$\omega = \overline{\omega} \Rightarrow \left(\left[\hat{K}\right] + i\rho\overline{\omega}\left[\hat{C}\right] - \overline{\omega}^2 \left[\hat{M}\right] \right)\{a\} = -i\rho\overline{\omega}\left\{\hat{F}\right\} \quad 4.24$$

4.1.3 Disadvantages

One of the major problems of the FEM is its limitation in solving acoustics problems. These require a uniform mesh with a minimum of 6 elements per wavelength to obtain a reasonably accurate solution. Hence a finite cube of side 2 metres would require 6000 elements at 500 Hz. At 20 KHz, it would contain 380 million elements! Obviously, ultrasonic problems would only exacerbate the situation!

Its second major problem is that it does not satisfy the Sommerfeld condition (Equation 3.1) and hence has to depend on other formulations for solution of infinite domain problems. The latter invariably results in a very large number of degrees of freedom, with the corresponding increase in computation time.

4.2 Alternate Formulations

Higher frequencies and infinite domains require alternate formulations. Dr. McKerrow suggested considering each ultrasonic chirp to be a single pulse of energy. At present, the author is not cognisant of this approach. Whether it falls under the category of FEM or some other high frequency analysis (like statistical energy analysis) is a question that requires further study.

4.2.1 Approximate Formulation

One method for modelling radiation (exterior) problems is to surround the object at some 'large' finite distance by a perfectly acoustic absorbing surface to prevent reflection. However, this never represents the real problem and seldom gets good results.

4.2.2 Parallel Processing

To handle the large number of DOF, efforts have been made by various authors to speed up computing by use of vector processors, like CRAY1 [Kratz83] and Cyber 205 [Diekkamper83], and processor arrays, like ICL DAP [Duller83]. However, in all cases, it was necessary to coerce the problem into a form suitable for solution on these machines. In fact, Kratz [83] states that the use of commercial packages may not give the fastest results, as they are not fine-tuned to the corresponding CPU architecture. He suggested that better CPU utility can be obtained by writing your own pipelined programs. His recommendation is suitable for those who wish to fully utilise the CPU time they buy.

Unfortunately, these large computers are not readily available to all users. There is always a premium on time, cost and availability. So it seems appropriate to consider any alternate method that reduces the number of computations required. The Wave Envelope Element Method (WEEM) and the Boundary Element Method (BEM) fill this slot.

4.2.3 Wave Envelope Element Method

This method has only recently been developed and is still in the experimental stages [Coyette92-2]. It uses a special weighting function that involves the product of

the complex conjugate of the shape function and a factor that tends to zero as distance increases (tends to infinity). This effectively permits wave envelope elements to extend to infinity. Its complex conjugate nature results in cancellation of all exponential terms. This permits evaluation by Gauss-Legendre schemes rather than the more complex Gauss-Laguerre schemes [Abramowitz64] needed by the infinite element [Bettess77].

The latter has not gained a large following because its complex nature made its implementation tedious.

4.3 The Boundary Element Method

Green's third identity [Stakgold68] forms the basis of the BEM. It states that, for any two sufficiently continuous functions, u and v, we have the relation

$$\int_V \left(u\nabla^2 v - v\nabla^2 u\right)dV = \oint_S \left(u\partial_n v - v\partial_n u\right)dS \qquad .. \qquad .. \qquad 4.25$$

The importance of this relation lies in the fact that volume integrals and second-order derivatives on the left hand side are replaced respectively by surface integrals and first order normal derivatives on the right hand side.

Instead of the volume, the surface is discretised, resulting in the matrix formulations yielding full matrices of lower dimension than the sparse matrices of the FEM. For typical problems, it reduces the number of degrees of freedom required to model the problem, while still maintaining similar accuracy in the solution. Being a surface method makes it easier to model infinite domain problems and quicker to generate and refine the mesh, simultaneously reducing computational costs. So it was quickly accepted as a powerful alternative to the FEM.

Shaw [91] states, "There are essentially an infinite number of Boundary Integral Equation (BIE) formulations". These may be based on source distributions or Green's functions or Betti's reciprocal theorem or virtual work, etc.

Brebbia [80] developed the Galerkin Method of Mean Weighted Residuals for BEM. We examine it to try to provide common features between the FEM and the BEM formulations (Appendix D), to speed up software development time.

Wu [91] showed how the Helmholtz equation could be used for acoustic radiation and scattering problems. Bernhard [91] used it to discuss the sensitivity of BE acoustics response to boundary condition changes, to geometry shape changes and to frequency changes. This would be useful in transducer analysis and design.

Ciskowski et al [91] claim that they are aware of only two BEM applications in bioacoustics, of the vocal tract and of the hearing system. They further develop it to model ear-muffs (Figure 4.1).

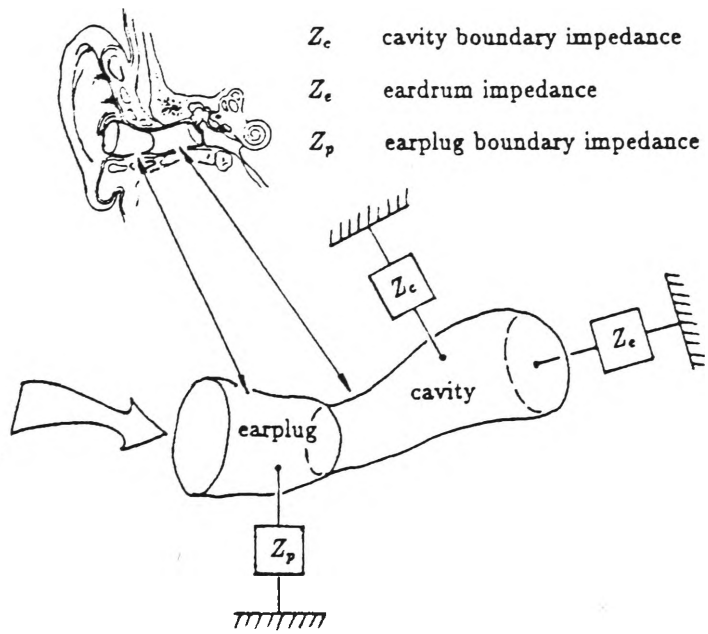


Figure 4.1: The plugged ear canal and its assumed equivalent model [Ciskowski91].

4.3.1 Theory of the Acoustics Direct BEM

The field at point Q due to the point source at point P is given by the Green function [Stakgold68]

$$G_P(Q) = \frac{e^{-ikr}}{4\pi r} \qquad 4.26$$

where r is the distance between P and Q.

We note the choice of Green function incorporates the Sommerfeld radiation condition in its formulation, thus permitting the BEM to be used for infinite domain problems.

Let p and G_P be solutions to the Helmholtz equation. Hence they satisfy it.

$$\nabla^2 G_P = \delta_P - k^2 G_P \Leftrightarrow p\nabla^2 G_P = p\delta_P - pk^2 G_P \quad .. \quad 4.27$$

$$\nabla^2 p = -k^2 p \Leftrightarrow -G_P \nabla^2 p = \left(-G_P\right)\left(-k^2 p\right) \quad .. \quad .. \quad 4.28$$

where $\delta_P = \delta(P)$ is the Dirac Delta function at P.

Setting $u = p$ and $v = G_P$ in Green's 3rd identity, we get

$$\int_V \left(p\nabla^2 G_P - G_P \nabla^2 p\right) dV = \oint_S \left(p\partial_n G_P - G_P \partial_n p\right) dS \quad .. \quad 4.29$$

Substituting from 4.27 and 4.28 into 4.29, we get

$$\int_V p\delta_P dV = \oint_S \left(p\partial_n G_P - G_P \partial_n p\right) dS \quad .. \quad .. \quad .. \quad 4.30$$

Depending on the location of the point of evaluation, the LHS of equation 4.30 takes different values. In particular, for a point on the boundary surface, we have the value $\frac{1}{2} p_P$, since only half the delta function lies inside the boundary.

Let us assume that the surface is discretised into n parts.

ie. $S = \bigcup_{i=1,n} S_i$ 4.31

Let the nodal pressure on S_i be

$$p = \sum N_j p_j \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.32$$

Let the nodal normal derivative of pressure on S_i be

$$\partial_n p = \sum N_j \partial_n p_j \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.33$$

Substituting 4.31, 4.32 and 4.33 into 4.30, we find that the pressure at any node on the surface is given by

$$\frac{1}{2} p_P = \sum_i \left[\int_{S_i} \left(\left(\sum_j N_j p_j \right) \partial_n G_P - G_P \left(\sum_j N_j \partial_n p_j \right) \right) dS \right]$$

.. .. 4.34

Since p and $\partial_n p$ are constants, they are removed from within the integration to give

$$\frac{1}{2} p_P = \sum_i \left[\sum_j p_j \int_{S_i} \left(N_j \partial_n G_P \right) dS - \sum_j \partial_n p_j \int_{S_i} \left(G_P N_j \right) dS \right]$$

.. .. 4.35

Simplifying 4.35 gives us

$$\frac{1}{2}p_P = \sum_k [p_k h_k(p) - \partial_n p_k g_k(p)] \quad .. \quad .. \quad .. \quad 4.36$$

where

p_k and $\partial_n p_k$ are unknown constants

$h_k(p)$ and $g_k(p)$ are known functions of p.

The equivalent matrix equation becomes

$$\left[H + \frac{1}{2}\right]\{p\} = [G]\{\partial_n p\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.37$$

This is a system of n equations in $2n$ unknowns. Application of the n BCs involves moving columns of H and G to the other side of the equation, finally resulting in a system of n equations in n unknowns,

$$Ax = y \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.38$$

where x is a vector of unknown quantities of p and $\partial_n p$

This can be solved for all the boundary surface values. To find the value at any point in the domain, these boundary surface values are re-substituted into

$$p_p = \oint_S (p\partial_n G_P - G_P \partial_n p) dS \quad .. \quad .. \quad .. \quad .. \quad 4.39$$

4.3.2 Theory of the Acoustics Indirect BEM

$$p_p = \oint_S (\mu \partial_n G_P - \sigma G_P) dS \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.40$$

is the final relation for the Indirect BEM. It is similar in form to that for the Direct formulation. However, rather than the nodal pressure and the nodal normal derivative of pressure,

$$\mu = p^+ - p^- \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.41$$

and $$\sigma = \partial_n p^+ - \partial_n p^- \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.42$$

are used. These are the pressure difference and the velocity difference, respectively, between the exterior and interior fields. The '+' and '-' denote the exterior and interior fields respectively.

4.3.3 Discussion

One peculiarity of the Direct approach is that it either solves the interior problem or the exterior problem, but not both. To simultaneously analyse both the interior and exterior fields, it is necessary to use the Indirect BEM, which can be used to directly obtain the solution to the problem of an ultrasonic transducer radiating into space. However, the results can not be used directly, but must be manipulated into a meaningful physical representation.

4.4 Coupled Methods

The acoustic and structural (mechanical) behaviours of any vibratory system are inter-linked and strongly depend on various coupling factors which relate the two individual behaviours (Figure 4.2).

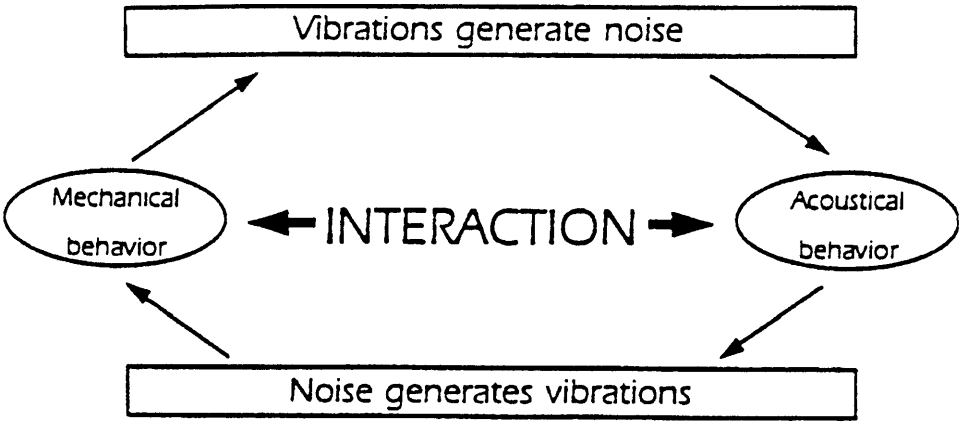


Figure 4.2: Interaction between structural and acoustic problems [Migeot93].

Any vibrating body (source) perturbs the media in which it is immersed and also any surrounding objects with which it is in contact. This results in the generation of acoustic waves that propagate away from the source. The uncoupled structural problem typically neglects the effect of the surrounding media in the calculation of the radiated sound field.

On the other hand, pressure waves (acoustic waves) impinging on any body induce vibrations in it. These pressure waves may have been caused by the body itself (resulting in feedback between the acoustic and structural behaviours), or may have been generated from an entirely different source. The uncoupled acoustics problem typically calculates the pressure field on any bodies placed in the acoustics field (sound scattering), neglecting any structural deflections that could arise and modify the pressure field.

In either case, the calculated results can be used as boundary conditions for the corresponding coupled problem; ie. the structural problem determines the radiated sound field that can be used as velocity boundary condition for the acoustical problem, while the acoustics problem determines the pressure that can be used as subsequent structural loading.

In many problems, these two behaviours can not be separated. The interactions are so intricate that it becomes necessary to study both simultaneously, because each, individually, can not describe the solution correctly. The fluid influences the structure and the structure influences the fluid. Hence, the structural and acoustic equations are

solved simultaneously, by relating their degrees of freedom with suitable coupling constraints. These latter are defined by two basic physical phenomena:-

a) structural vibrations cause sound waves in the fluid, and

b) pressure on a body's surface acts as a supplementary load.

So the coupled problem can be represented as

acoustic	+	mechanical	solve	sound	+	surface
sources		loading	---->	field		displacements

.. .. 4.43

Neglecting damping effects, uncoupled structural behaviour for the FEM can be expressed as

$$\left[K_S - \omega^2 M_S\right]\{u\} = \{F_S\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.44$$

Similarly, uncoupled acoustic behaviour can be expressed as

$$\left[K_F - \omega^2 M_F\right]\{p\} = \{F_A\} \text{ (FEM)} \quad .. \quad .. \quad .. \quad 4.45$$

$$[A(\omega)]\{p\} = \{F_A\} \text{ (Direct BEM)} \quad .. \quad .. \quad .. \quad .. \quad 4.46$$

$$\frac{1}{\rho\omega^2}[H(\omega)]\{\mu\} = \{F_A\} \text{ (Indirect BEM)} \quad .. \quad .. \quad 4.47$$

If coupling is considered in the formulation of the equations, then the structural behaviour generates an additional 'acoustic' loading, $\rho_F \omega^2 [C_S]\{u\}$, while the acoustic behaviour generates an additional 'structural' loading, $[C_F]\{p\}$ or $[C_F]\{\mu\}$. These are usually added to the RHS of the above equations (4.44 to 4.47), and after suitable manipulation leads to the following sets of coupled equations:

Direct BEM

$$\begin{bmatrix} [K_S - \omega^2 M_S] & [C]^t \\ \rho\omega^2 [B(\omega)] & [A(\omega)] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{p\} \end{Bmatrix} = \begin{Bmatrix} \{F_S\} \\ \{F_A\} \end{Bmatrix} \quad .. \quad .. \quad 4.48$$

Indirect BEM

$$\begin{bmatrix} [K_S - \omega^2 M_S] & [C]^t \\ [C] & \frac{1}{\rho\omega^2}[H(\omega)] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{p\} \end{Bmatrix} = \begin{Bmatrix} \{F_S\} \\ \{F_A\} \end{Bmatrix} \quad .. \quad 4.49$$

FEM

$$\begin{bmatrix} [K_S - \omega^2 M_S] & [C]^t \\ \rho\omega^2 [C] & [K_F - \omega^2 M_F] \end{bmatrix} \begin{Bmatrix} \{u\} \\ \{p\} \end{Bmatrix} = \begin{Bmatrix} \{F_S\} \\ \{F_A\} \end{Bmatrix} \quad 4.50$$

A typical method of solving the coupled problems efficiently is by first eliminating all structural unknowns (as the structural impedance matrix is banded symmetric) and then solving the reduced fluid system.

4.5 The Piezoelectric Formulation

For the study of piezoelectric substances, we have the following system of global equations [Lerch90], which couple together the intrinsic structural and potential variations.

$$M\ddot{u} + D_{uu}\dot{u} + K_{uu}u + K_{u\phi}\phi = F_u \quad .. \quad .. \quad .. \quad .. \quad 4.51$$

$$K_{u\phi}^t u + K_{\phi\phi}\phi = F_\phi \quad .. \quad .. \quad .. \quad .. \quad 4.52$$

where

u, ϕ are unknown displacement and potential respectively

M is mass matrix

D_{uu} is mechanical damping matrix

K_{uu} is mechanical stiffness matrix

$K_{u\phi}$ is piezoelectric coupling matrix

$K_{\phi\phi}$ is dielectric stiffness matrix

$F_u = F_B + F_S + F_P$ is the total displacement force due to all the mechanical forces like body forces, surface forces and point forces

$F_\phi = Q_S + Q_P$ is the total potential force due to electrical surface charges and electrical point charges

For a system of n nodes, with 4 DOF per node, there are typically $4n$ equations, ie. $3n$ equations for the mechanical positions (a vector with 3 structural displacements) and n equations for the electric portion - a scalar quantity (eg. potential).

4.5.1 Piezoelectric Analyses

Three types of analyses can be done to determine the required piezoelectric response: static, modal and dynamic.

4.5.1.1 Static Analysis

This is done by ignoring inertial and damping effects in equations 4.51 and 4.52, giving us

$$K_{uu}u + K_{u\phi}\phi = F_u \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.53$$

$$K_{u\phi}^{t}u + K_{\phi\phi}\phi = F_{\phi} \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.54$$

Manipulating equation 4.54, we get

$$\phi = -K_{\phi\phi}^{-1}K_{u\phi}^{t}u + K_{\phi\phi}^{-1}F_{\phi} \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.55$$

Substituting from 4.55 into 4.53 to eliminate all unknown potentials gives us

$$\left[K_{uu} - K_{u\phi}K_{\phi\phi}^{-1}K_{u\phi}^{t}\right]\{u\} = \{F_u\} - \left\{K_{u\phi}K_{\phi\phi}^{-1}F_{\phi}\right\} \quad .. \quad 4.56$$

which is solved to find the solution (displacements), and hence other dependent parameters like strain, electric field, stress and electric flux density.

4.5.1.2 Modal Analysis

Modal analysis is used to extract the natural frequencies and the mode shapes of the piezoelectric structure. This knowledge is useful in determining the dynamic response. Additionally, some transient solution procedures (eg. modal superposition) require the results of a modal analysis.

Ignoring damping and forces in equations 4.51 and 4.52, we get

$$M\ddot{u} + K_{uu}u + K_{u\phi}\phi = 0 \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.57$$

$$K_{u\phi}^{t}u + K_{\phi\phi}\phi = 0 \quad .. \quad .. \quad .. \quad .. \quad .. \quad 4.58$$

As for static analysis, equation 4.58 is used to eliminate potential from equation 4.57. The resulting equation is then used to determine the modes.

4.5.1.3 Dynamic Response Analysis

This is used to determine the frequency response of the piezoelectric structure to one or more sinusoidally-varying forcing functions, all of which must have the same frequency though they can have separate phase.

The displacements (solution) are also assumed to vary sinusoidally at the same frequency, though they may have a different phase compared to those of the forcing function(s). They are specified in terms of either amplitudes and phase angles or real and imaginary parts.

4.5.2 Observations

4.5.2.1 Matrices

The piezoelectric equations have an underlying complexity beneath their matrix equations [Lerch90]. An examination reveals extensive use of various matrix operations like multiplication, transpose, etc. Is it necessary for the software to have generalised matrix functions for the above operations? Or is hard coding of these operations preferable for run-time efficiency? The latter is only possible if all problems fall under the same class. However, if the involved tensors have variable dimensions, then a more generalised format would permit flexibility for later enhancements and extensions. At the present stage, this researcher has not found any definitive answers to these two questions, though most literature use the form of equations reproduced here. This lack of literature could probably be attributed to the fact that this theory is presently being commercially exploited!

4.5.2.2 Axes

The orientation of the axes is particularly intricate. If the orientation is changed, different behaviour is observed (Chapter 3). While it may be mathematically feasible to obtain such results under the prevailing physical conditions and geometry, it

may not be physically possible to manufacture (or obtain in natural form) the material with the axes oriented in the given directions.

4.5.2.3 Material Constants

The physical transducer's behaviour depends on the material constants (Table 3.1). Slight variations in the percentage composition of the individual components of the piezoelectric material can result in different material constants [Berlincourt64]. This implies the ability to repeatedly manufacture the material with the required constants (repeatability), and to accurately measure the constants to guarantee reliable simulations.

It makes no sense to model a transducer with concocted constants, due to the likelihood of being unable to find a real-world material with the requisite properties.

This essentially makes this kind of modelling a preserve of the transducer manufacturers, who have a vested interest in the results and would prefer not to release information that may be of commercial importance to their competitors. **Obtaining accurate constants is critical for the success of these 'Element' methods.**

4.5.2.4 Equation Solvers

This is a finite problem involving only the characteristics of the piezoelectric transducer. So standard FE equation solvers can be used as the matrix equations are banded symmetric - a typical FEM formulation.

However, extension of the problem to include the generation of the radiated field would cause it to be an infinite problem and hence require use of the BEM or WEEM or one of the coupled formulations. In general, the WEEM results in banded symmetric matrices, implying that the FE solver can still be utilised. But the BEM and the coupled approaches usually result in asymmetric full matrices that cannot avail of the efficiency of the FE equation solver.

4.6 How to select Optimal Method of Solution?

As a general rule of thumb, the FEM is constrained to those problems involving low frequencies and having domains of limited size - so that the total number of unknowns is less than some upper bound that is determined by the CPU speed, the system memory capacity and the hard disk storage capacity of the machine on which the problem is being solved. On the other hand, high frequencies and infinite domains are handled by the BEM.

However, the choice is not always so trivial. The ability to select a suitable method (from among FEM, BEM, FEM/FEM, FEM/BEM) for the solution of a given problem is not an easy task. It depends on practical experience and can be facilitated by consideration of the parameters listed in Table 4.1. These very often influence each other. The choices are not exhaustive and vary from problem to problem.

The use of the WEEM also requires further study. Astley[91] found that the WEEM obtained the solution marginally faster than the BEM, even though it had approximately ten times the number of DOF of the latter. This could be attributed to the sparsity of the matrix and the simpler calculations required for the WEEM as compared to the BEM.

However, in the calculation of the exterior solution, the WEEM was much faster then the BEM; because the former required only simple interpolation whereas the latter required an integration over the entire surface for every field point.

In Table 4.1, details about the WEEM that I cannot find in the literature are indicated by a question-mark.

Table 4.1: Comparison of the different element methods

Selection Parameter	FEM	WEEM	BEM
Modelling time	-high (3-D complex)	-medium	-low (surface only)
	-difficult & tedious	-medium	-relatively easy
Time to solve equations	short (solution of banded symmetric matrices)	short (solution of banded symmetric matrices)	long (solution of full matrices)
Time to generate field	quick (interpolation within the element)	quick (interpolation within the element)	slow (integration over the entire surface)
Solution generated	At all the nodes	(?)	At user selectable nodes
Number of DOF	very high	low(?)	low
Computer resources	-large disc space	-small(?)	-less disk space
	-large main memory	-small(?)	-less main memory
Frequency of interest	low	high	high
Volume absorption	easily modelled	(?)	not so easy
Nature of the fluid	heterogeneous		

Table 4.1: Comparison of the different element methods (contd.)

Selection Parameter	FEM	WEEM	BEM
Interior Problem	yes	yes	yes
Exterior Problem	can't be modelled	yes	automatically handled
System Matrices	-freq independent	-freq independent	-freq dependent
	-symmetric	-symmetric	-asymmetric(Direct BEM)
			-symmetric (Indirect BEM)
Variation of frequency	Matrices calculated only once	Matrices calculated only once	Recalculation of Matrices for each frequency under study
Modal response	-yes, preferred as faster	(?)	complicated due to implicit frequency
	-permits mode selection		
Direct Response	yes	(?)	yes

4.7 Conclusions

1) To date, all the applications viewed by the author apply the Helmholtz Equation for the solution of acoustics problems. Appendix B shows how the Helmholtz Equation is a special case (frequency domain harmonic motion) of the linear wave equation, which in turn is a special (isotropic) case of the general (anisotropic) quasiharmonic equation. In fact, it can only be used for those problems whose geometries and boundary conditions are defined by coordinate systems that are separable in the Helmholtz equation. In other words, this precludes the use of the Helmholtz equation for those problems which involve coupling between the axes.

For such problems, some form of the quasiharmonic equation will have to be used. In the 2-D case, this is quite tedious. But in the 3-D case, the problem becomes formidable. So it is quite usual, for problems involving classical linear elasticity theory, to assume infinitesimally small linear gradients and neglect non-linear second-order gradient products (rotation effects), hence reducing the problem to more manageable proportions. Similar approximations will have to be made for piezoelectric problems.

The end result is that **the Helmholtz Equation is not suitable for the analysis and design of piezoelectric transducers. Instead some form of the quasiharmonic equation will have to be used.**

2) Appendix C lists the FEM equations for various 1-D and 2-D problems and assumes equivalent results for 3-D. We note that the overall form of the matrix equations remains the same. However, in spite of approximating the inter-axial coupling (2-D case), the individual elements that comprise the matrices become more complex. In the 3-D case, coupling extends to interactions between all

three axes. Thus **3-D formulations have much more complexity, which is not readily apparent by inspection of the matrix equations.**

3) We note that the Dynamics formulations for both structural and acoustics problems results in the same form of matrix equations, though the constant coefficients may differ.

Chapter 5 - Software

5.1 FE models for piezoelectric transducers

Within the last 5 years there has been quite a spate of 2-D and 3-D FE models of piezoelectric transducers. Previous to that, various researchers have used 1-D theoretical models [Berlincourt64].

5.1.1 2-D models

Kunkel[90] used axisymmetric rectangles, each divided into 4 triangles, to study PZT-5H transducers (Figure 5.1).

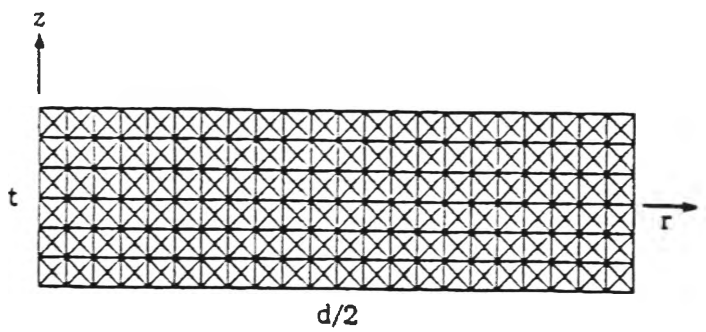


Figure 5.1: Finite-element mesh representing the cross-section of a piezoelectric disk. It consists of axisymmetric rectangles, each divided into 4 triangles. Top and bottom represent disk surfaces. Left side is axis of symmetry $r = 0$, and right side is disk edge $r = d / 2$ [Kunkel90].

Babic[91] used axisymmetric triangular ring elements (Figure 5.2a) for design and optimisation of a PZT-5A piezoceramic transducer (Figure 5.2b). Each element had 6 degrees of freedom, 2 per node, which represented axial and radial displacements respectively. His transducer used a vibrating membrane to ensure efficient transmission of the radiation field to air.

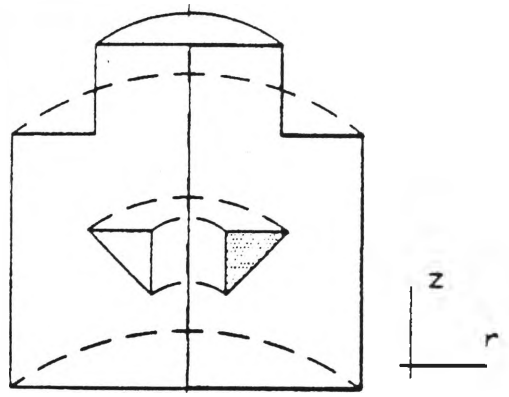


Figure 5.2a: Triangular ring elements used in finite-element analysis of the transducer in Figure 5.2b [Babic91].

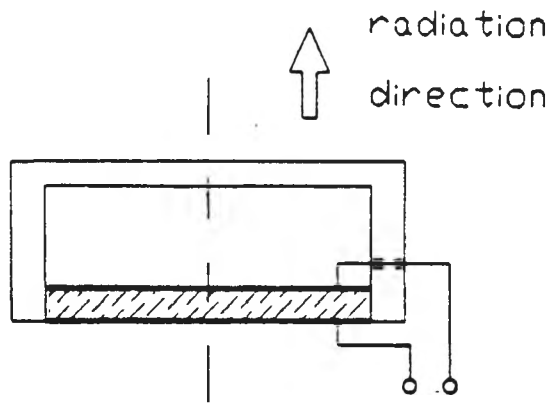


Figure 5.2b: Cross-sectional view of a transducer. The shaded area shows the piezoceramic disk, the shaded lines show the silver electrodes and the C-shaped aluminium housing is used to improve radiation in the direction shown [Babic91].

5.1.2 3-D models

According to Lerch[90], of the Siemens Research Centre in Germany, 1-D models, though in use, are inadequate for correctly computing all physically present modes of 2-D and 3-D piezoelectric media. Preferably, 3-D models, with their associated accurate data of material tensors, are required to accurately model practical transducers. He used 2-D (plane-stress, plane-strain, axisymmetric) and 3-D piezoelectric finite elements (Figure 5.3) to simulate and optimise piezoelectric devices for ultrasonic applications. For experimental verification of the computed eigenmode shapes, he used laser interferometric measurements. He claims that the "thickness mode is the mode of interest for ultrasonic imaging applications", as for suitable width-

to-thickness ratio, most of the electric energy is converted into a normal displacement of the sound emitting face.

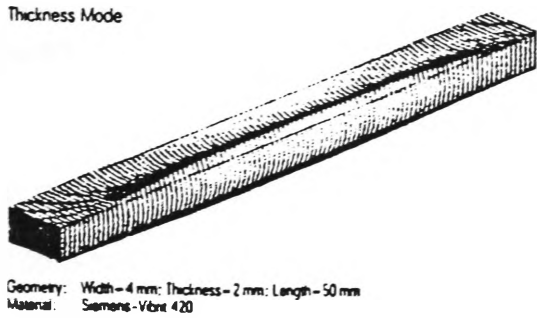


Figure 5.3: 3-D mode shape of the thickness mode of a piezoceramic bar with $W / T = 2.0$ [Lerch90].

Ostergaard[86], of Swanson Analysis Systems (SAS), used 8-noded 3-D brick elements to model a NEPEC6 piezoceramic disc (Figure 5.4) and a PZT4 cube. Subsequently, SAS has made further progress and expanded ANSYS, its proprietary FE analysis software package (Section 5.2.3).

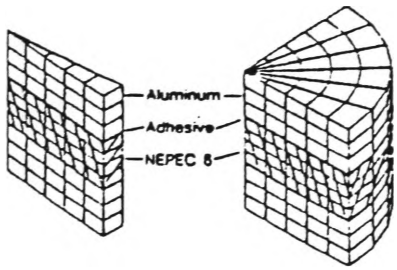


Figure 5.4: Cylindrical transducer model using 8-noded 3-D brick elements [Ostergaard86].

5.1.3 Wave Envelope Element Models

Though these are not particularly suitable for transducers, they are worth a mention here because of their far-field performance, and the fact that the WEEM has a very similar formulation to the FEM and so can be easily incorporated into existing FE programs.

5.2 FE software packages for ultrasonics

5.2.1 STRAND's Approach

STRAND has two approaches to solving Acoustics Problems.

The first involves converting the Wave Equation to the Helmholtz Equation and solving for the complex pressure solution. A separate solution is required for each frequency. This technique is only suitable for uncoupled problems. Thus, if the structural problem generates the sound, the structural solution is obtained first and the surface motion is then decomposed by Fourier analysis into its individual frequency components, which are used as input for the Helmholtz problem.

An alternate approach is to 'fool' the structural analysis part of the program into thinking that the Acoustics problem is an elastic one, with sound pressure replacing displacement, by selecting special choices of the elastic constants (property tensors). However, STRAND only supports symmetric property tensors [Atkinson91], which may not be used for piezoelectric materials. Additionally, Atkinson [91] claims that the method is inefficient, as it solves for 'displacement' in 2 or 3 dimensions, when only one dependent variable is actually required.

Although it is relatively cheap, STRAND is, however, limited to low frequencies or small problem domains [Atkinson91]. This seems to preclude ultrasonic applications and infinite domain problems.

5.2.2 NASTRAN's Approach

Some FE codes, like NASTRAN [Estorff91], set shear = 0 for acoustics problems. This makes them unsuitable for modelling of piezoelectric materials, which have shear. They could, however, be used for those modes of operation in which shear does not play an important role, if such modes exist!

5.2.3 ANSYS' Approach

The ANSYS program [ANSYS5.0] utilises special 2-D and 3-D fluid elements (FLUID29 and FLUID30) designed for the purpose of acoustic analysis. These elements model the fluid medium and the fluid-structure interface (Figure 5.5a). They assume small density changes and compressible fluids. Solutions are in terms of pressures or displacements.

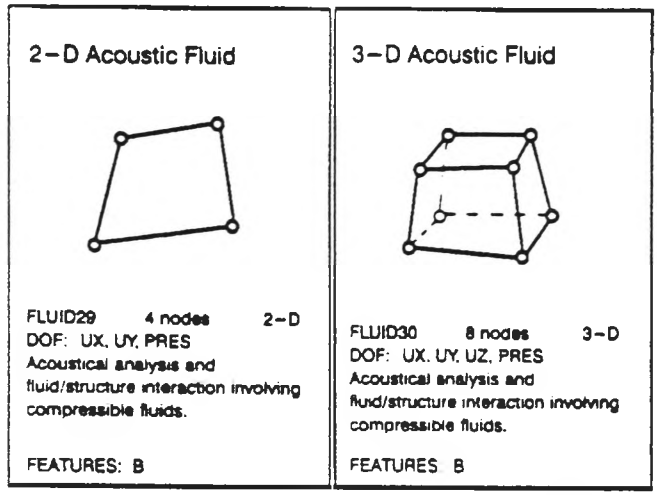


Figure 5.5a: ANSYS5.0 acoustic elements [ANSYS5.0].

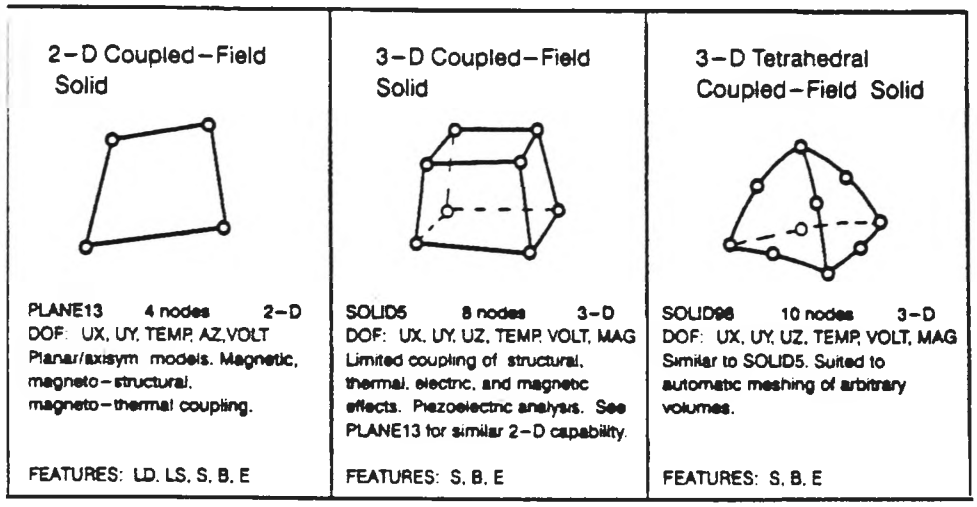


Figure 5.5b: ANSYS5.0 piezoelectric elements [ANSYS5.0].

For the study of piezoelectric structures, 3 solid coupled-field elements (SOLID5, SOLID98, PLANE13) are provided (Figure 5.5b). These permit a variety of

input data such as complete 6-by-3 piezoelectric constants; isotropic, orthotropic and anisotropic elastic stiffness or compliance constants; and dielectric constants.

ANSYS supports substructuring, which can be used to improve the modelling efficiency and/or the solution computation time. Substructuring is the reduction of a group of elements to a single independent element (superelement). This could be particularly useful in the design of array transducers, where each element of the array could be a pre-calculated superelement.

ANSYS' acoustic capabilities are limited to the study of the propagation of sound waves in a contained fluid medium (eg. the distribution of sound waves in an auditorium) or the analysis of structures immersed in a fluid (eg. damping effects of water on a ship's hull). Hence, it can not solve infinite domain problems.

5.3 SYSNOISE for ultrasonics

This package combines both the FEM and the BEM into a powerful tool for solution of acoustics problems. It basically matches our requirements. But it is prohibitively expensive for academic work.

SYSNOISE costs $100,000 for the entire commercial package, though it does provide universities with a stripped down version that handles problems up to 200 nodes in size. This costs $2500 per year, with the provision that a yearly report be made about its use! This limited version of software would be unsuitable for our main areas of interest, viz. a study of transducer analysis and far field problems.

Consider! With our desired frequency being 50 kHz typical, we have a wavelength of 6.8 mm (approx.). With a minimum of 6 elements per wavelength, we would require the nodes to be spaced approx. 1 mm apart. With a piezoceramic bar of dimensions 50 mm by 2 mm by 4 mm (length, thickness, width) [Lerch90], we would have a total of 51 * 3 * 5 = 755 nodes, which is beyond the capability of the educational package.

5.4 Selection of Elements - A Discussion

From the preceding sections, it follows that designing of a minimal package with rudimentary piezoelectric modelling capabilities would entail implementation of 2-D axisymmetric elements (at least). Later, 3-D elements and wave envelope elements could be added for more functionality.

A time frame for implementation of these elements and the associated FE solver is unknown as it is not specified in the literature. Lerch[90], Ostergaard[86] and Kunkel[91] only incorporated their new elements into the existing FE packages.

5.5 The Need for One's Own Software

A study of ultrasonic transducers would require software that could model both the near field and the far field. This capability would inherently support the design and analysis of transducers and the study of radiation and scattering effects.

It is evident that the only package that fulfils both needs is SYSNOISE. Unfortunately, its cost lies beyond the means of an academic institution's budget, while its cheaper educational version confines problems to be most trivial (Section 5.3).

An alternative is to buy some software that provides our needs partially and develop and integrate any additional software that we require into that package. This requires a knowledge of the source code of the original commercial software (an impossible or an expensive proposition).

A further shortcoming is that SYSNOISE, in its present form, is incapable of directly modelling piezoelectric transducers. It has to rely on external pre-processing to obtain the global system equations for the problem, before it can initiate a solution. In

this respect, its developers (NIT) are collaborating with Thomson Sintra ASM (TSASM) to develop software for the study of piezoelectric transducers.

So it is obvious that there is a potential market for this software. Are there any other advantages?

5.6 Reasons for Writing Own Software

Many advantages would accrue from writing an 'Element' Method software analysis package for acoustic echolocation.

1) Firstly, the FEM relies on physical principles that can be represented by suitable partial differential equations. This would require a detailed knowledge of the physics of acoustics, echolocation and transducers, (and the equations that characterise them), before implementation of the software. This would typically broaden the author's knowledge base and could suggest alternate research areas, thus providing future applications for the program.

2) Most commercial packages were primarily written for structural analysis; and have subsequently been coerced to solve acoustics problems by setting the elastic constants to suitable values. Hence the acoustics problems have been constrained to fit the structural analysis formulations.

3) Writing a custom package would make execution marginally faster as the program would not have overheads for selecting different options.

4) One very important motivation for writing one's own software is the exorbitant cost of commercial packages!

5) On completion, the package could be used as an in-house study reference, providing future researchers with a tool for optimal design of transducers, both single and array. Additionally, it could be used to model and analyse radiation and scattering problems that are typically encountered in real-life.

It could also have other offshoots, like biological research into the working of a bat's hearing mechanism, or design of a 'hearing-aid' for humans, or 'intelligent' devices that use sound to 'see' and aid blind men in navigation.

6) The availability of source code would permit future expansion and modification of the program to incorporate additional functionality that is not available in commercial packages.

'Custom design' of elements would enable particular problems to be solved most efficiently.

7) It could also provide the ability to modify and update material parameters as a result of measuring the radiation fields produced by transducers and by measuring the acoustic vibrations. Hence, local knowledge could be added to improve the simulations, resulting in a feedback loop to optimise the transducer design.

5.7 Reasons for not Writing Own Software

The major disadvantage is that of re-inventing the wheel! And no knowledge about an accurate estimate for the development time!

Chapter 6 - Software Design

To get a feel for the software complexity and to obtain knowledge enabling a rough estimate of the development time required, it was decided to implement a small FEM package, called FAME (Finite element Analysis in Modula-2 for Echolocation), which could be used as the groundwork around which additional routines could be built. We have decided to base our software upon an existing Finite Element Method Package called UNAFEM, written in F77 by John Denkmann for David Burnett's book, 'Finite Element Analysis: From Concepts To Applications' [Burnett88]. The following section considers our software design criteria, and justifies our choice of UNAFEM.

6.1 Software Requirements

1) UNAFEM was developed in a detailed structured manner, as it was to be primarily **an educational tool**. This is identical to our purposes for developing FAME.

2) **A good user interface** is needed. The data input should be easily manipulated and corrected with meaningful error messages where required and (the fewest possible keystrokes). The use of a dialog box would be preferable.

   In this respect, UNAFEM fails. It has a non-interactive user interface in which data was read in from a data file by use of a formatted input read statement. Error messages are displayed on screen and caused the program to HALT. Consequently, the Pre-Processing Section will have to be modified extensively to support interactive Dialog Boxes for data entry and online error messages.

3) The data output should be graphically displayed to give a **clear visual representation of the solution.**

UNAFEM used DISPLA, a commercial graphics package, to display the output solution. We will have to develop some simple graphics software to view the same or similar results.

4) **It should be modular and easily expandable** to add more functionality to the package. The final product should be a program capable of solving ultrasonic problems by use of either the FEM or the BEM or the WEEM. These problems would include transducer design and analysis, and radiation and scattering problems.

   UNAFEM, though written in F77, has been developed in conformance with standard structured programming rules. GOTO's have been use with caution, resulting in a highly modular, readable comprehensible package. So, adoption of this package for FAME would reduce the design cycle, with the associated time saving benefits.

5) **It should be easy to test and verify**. Testing and verification of the initial parts of FAME could follow the lines laid out in Burnett [88], hence eliminating time spent in designing test problems and solutions. Subsequent testing and verification of the Acoustics solving portion of FAME, that is not covered in Burnett [88], will have to be developed from scratch.

6) **It should be possible to develop a working prototype soon**. This last criteria has been added because most commercial FEM packages have a relatively long development cycle before reaching the market.

   M2 (Appendix A) is the language being used to develop all the research software in our lab. So, to enable intelligent discussion of program implementation and bugs, it was decided to use M2 to develop 'Element Method' software on the Macintosh. It was felt that the adoption of UNAFEM (with its associated plus points stated above), the advanced features of M2 and the powerful menu-driven user interface libraries of the Macintosh could be combined to achieve working results quickly.

6.2 The Ideal Software Interface

A sophisticated interactive interface would provide immediate graphic feedback to visually indicate to the user the data being input. Hence, action could be taken to correct any incorrect data. Separate windows could be defined to display particular data. Hence input of nodes and elements would be reflected in an appropriate window. Definition of different element properties could result in the corresponding elements being shaded a different hue. Addition of different kinds of loads could be indicated by different types/coloured arrows, whose tip would indicate direction and size would indicate magnitude.

The mouse could be used to point at particular features to obtain related information. It could also facilitate various block editing functions like deleting, copying and adding sections to speed up problem definition. The monitor screen would either represent a 2-D image of the physical world (for 2-D problems) or select between a 2-D slice and 3-D perspective (for 3-D problems).

6.3 Software Estimate of UNAFEM

Since we are adopting UNAFEM as a model for our package, FAME, it makes sense to examine how best it can be translated/ported to M2. As previously noted, we will concentrate on developing an interactive interface. To determine a rough index for FAME's development time, a bit of reverse engineering is in order here.

The COCOMO model, designed by Boehme [81], is a productivity tool that predicts the total number of people required to complete a project on time. It uses an empirical set of constants to estimate the number of man-months of work involved in the project, and hence the number of real-time months within which it should be completed. Each man-month is estimated to be approximately 152 working hours. The

values of the constants depend on different criteria that evaluate hardware, software and personnel characteristics.

The basic COCOMO model, as explained by Sommerville [89], was used to estimate the development time for UNAFEM (4500 lines of code and comments). Very conservatively, we assume that UNAFEM is an 'organic mode' (simple) project, due to its small size and reduced communication overheads as most of the work was done by a single person, John Denkmann. (In retrospect, it appears to be an 'embedded' (complex) project!) We further assume that 20 percent of the code consists of comments, leading to 3600 lines of 'delivered source instructions'.

Application of the relevant formulae provided the result that two persons could finish the software if they worked 6 months full-time (excluding time spent in mathematical formulations like deriving systems equations, element equations, etc.). Use of the intermediate COCOMO model did not significantly alter the above figures.

6.3.1 Implications for FAME

The COCOMO estimate predicts the total project development time required for software design, implementation, debugging and testing. However, it does not indicate the relative contributions of each of these factors to the total figure. This makes adoption of UNAFEM's figures for FAME slightly suspect. Since we are porting UNAFEM to FAME, these figures are the best estimate that we have. However, we should keep in mind that, to reach the level of utility of UNAFEM, FAME does not need any design, though it does need the other three factors. It is only when additional functionality is being incorporated (new elements, method for solution of dynamics problems, user interface, graphics output, etc) that all four factors will come into play. For our purposes, rather than making a detailed estimate of the development time for FAME, we will assume that UNAFEM's estimates are accurate enough and can be used as predictors to develop FAME up to the same functionality as UNAFEM.

6.4 Software Design

6.4.1 Elements Needed

From Sections 5.1 and 5.4, we find that we only require to add the axisymmetric element to UNAFEM to obtain a rudimentary package capable of Acoustics. More advanced features can be implemented with 3-D elements.

Though UNAFEM has not implemented these elements, Burnett [88] has explained them and derived the relevant equations. He states that UNAFEM can be extended to include the above, though the extension is not trivial. Unfortunately, he does not specify a time frame.

However, it makes sense to develop FAME up to the level of functionality of UNAFEM before any other developments. This would provide us with a basic FEM solver, having capability for static analysis, modal analysis and first-order time transient analysis. Additionally, as seen in Appendix D, the isoparametric elements, have potential use in BEM solutions.

One drawback is that the Dynamics (second-order time transient analysis) equations are not implemented. This is essential for study of wave propagation.

6.4.2 Records Needed

Keeping the above in mind, and referring to all variables used in UNAFEM (Appendix G), we defined the following data types. The code for these data structures can be found in Appendix F. We note that these data types affect all subsequent design and they pervade through the entire software.

6.4.2.1 General Types

System contains common variables that specify a particular problem, eg. type and dimension of the problem; total numbers of nodes, elements, properties, load conditions, etc; flags to indicate whether FEM and Eigen calculations have been completed or aborted.

6.4.2.2 Input Types

Each *Node* specifies one of the nodes of the problem mesh.

Each *Element* describes an element of the problem mesh.

Each *Property* specifies physical properties of a particular material

Each *ILC* specifies one of the internal loads of the problem.

Each *EBC* specifies one of the essential BCs of the problem.

Each *NBC* specifies one of the natural BCs of the problem.

Each *Graph* specifies display parameters for the problem.

6.4.2.3 Output Types

Each *NodeSolution* specifies the function value at the node.

Each *ElementFlux* specifies values of the gauss points of the element.

Each *EigenSolution* specifies an eigenvalue and its eigenvector.

With the implementation of these data structures, dynamic storage allocation of all data would be possible (note the pointers), enabling larger problems to be solved and making more efficient use of available memory. This would be preferable to having static allocation of data, as it is at present. A slight debugging disadvantage would be to keep track of the pointers needed to manipulate the data, though this would be offset by the subsequent computation speed increase.

The required data types have been defined and are a preferable way to implement the software (at a later date). However, it has not been put to use as it would

have meant restructuring of the entire program, hence slowing down porting. Instead, the variables in Appendix G are used.

6.4.3 Files and Modules Needed

One time saver suggested was to use a single ASCII file for data input, rather than design an entire user interface with files, menus, dialog boxes and friendly error messages. However, the FEM is a data intensive technique. It was opined more advantageous to have a user friendly interface rather than something cryptic, as it would permit flexibility in problem definition and would save time in the long run. Hence, separate data files and data processing modules were required for each data type defined above.

One file called the system file contains data about the problem parameters. This data specifies the size of the other files indirectly, by indicating the total number of records of a given type.

All the data is not stored contiguously in one file, in order to permit easy modification and expansion of problems. Instead, different data is stored in different data files, with all the data records of a given type stored in a single contiguous file.

Each file's records can be individually updated by a particular module and the corresponding data will be hidden from all others for data integrity. Assuming the grouping together of similar records, separate files would permit a new record to be added directly to the corresponding file, whereas a single data file would require the entire file to be read and re-written in order to write a new record to the file. In the absence of grouping, the concept becomes much more complex, due to the records being of different sizes. Now each record would require some form of identification, such as a tag-field (type) and an index field (number), with corresponding increase in complexity of the processing software.

Table 6.1: Data files and their contents

DATA FILE NAME	INFORMATION CONTAINED
filename:	System RECORD
filename.nodes:	x, y
filename.elems:	ntype, ngpint, nphys, icon
filename.props:	alphax, alphay, beta, gamma, mu
filename.ilcs:	fint
filename.ebcs:	iuebc, uebc
filename.nbcs:	inp, jnp, knp, taunbc
filename.plot:	Graph RECORD

The data files that have been implemented are listed in Table 6.1 with the corresponding data items that they contain. More information regarding the data can be found in Appendices F and G. Note that the user only requires to remember the 'filename'. The program inserts the extensions and manipulates the required data.

6.4.4 Menus Needed

We will provide the 8 menus shown in Appendix H. Note that the greyed out items have not yet been implemented.

1) A FILE Menu to permit design of *New* problems, to *Open* existing problems (either for recalculating or modification) and to *Save* problems to disk.
Page Setup and *Print* options are hooks to be implemented later.

2) An EDIT Menu with commands like *Undo*, *Cut*, *Copy*, *Paste* and *Clear* to facilitate later development of a powerful User Interface.

3) A MISC (miscellaneous) menu from which you can select the dimension of the problem, *1-D*, *2-D*, *Axisymmetric* or *3-D* (to be implemented later), and define whether it is *Single* or *Batch*.

The Batch concept (to be implemented) is targeted at permitting the user to set up the data files for several problems, which can then be solved without any user interaction (maybe, overnight!). Errors would automatically abort the problem and update an error file with the appropriate information. The next problem would then be initiated. Of course, this presumes sufficient disk space to store calculated results and some sort of simple interpretive control language.

4) A Windows menu that permits 9 windows to be *Opened* and *Closed*. Windows may display either function or flux values of either the same or different problems.

5) An Input menu to specify all numerical data relevant to the problem. Intelligent menus grey out any options that aren't permitted. Each option brings up a dialog box. (See Section 6.4.5 for details) Checks are implemented to ensure that the user can not input invalid data and detailed messages inform about the cause of any error.

6) A Problem Type menu that supports solution of:-

a) Boundary Value Problem (BDVP)

b) Eigen Problem (EIGP)

c) Initial Boundary Value Problem (IBVP) (later development)

d) Dynamics Problem (DYNP) (later development)

Other types of problems can be included later.

7) An Output menu which:-

a) generates ASCII text files of various input data and the solution,

b) draws a simple graphic representation of the BDVP solution.

8) The Solve menu initiates the calculation of the solution. It supports:-

a) Finite Element Method (FEM)

b) Boundary Element Method (BEM) (future expansion)

6.4.5 Dialog Boxes Needed

The dialog boxes (Appendix I) were designed to be user-friendly (they should be easy to use), consistent (the same buttons should have the same meaning in different dialog boxes), aesthetic (they should look good, ie. not appear cluttered), and coherent (they should permit modification of related data) with feedback (they should show meaningful error messages).

The Nodal Dialog Box is shown in Figure 6.1.

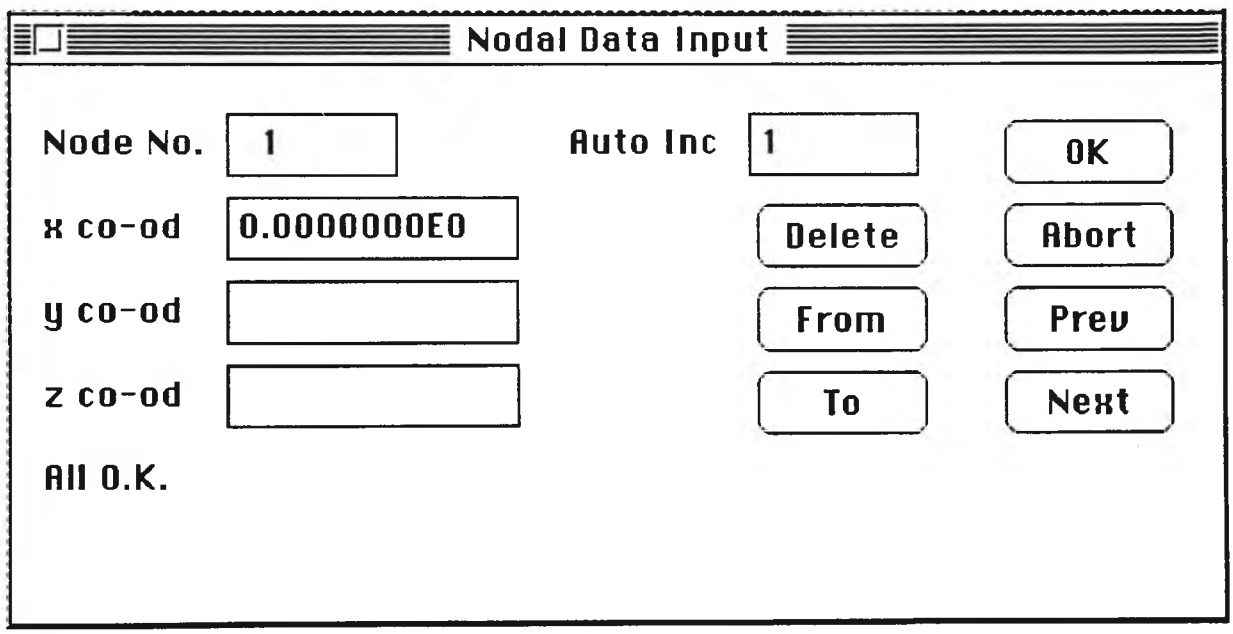


Figure 6.1: Dialog box used to input nodal data.

First, an explanation of the common options/functions is in order. Each dialog box contains a variable which is an index into the corresponding data arrays.

Prev decrements the index and displays preceding data.

Next increments the index and displays succeeding data.

Delete deletes the current data, decrements the index and displays preceding data.

OK verifies all entered data and only closes the dialog box if there is no error. Any error generates an error message in the lowest region of the box.

Cancel clears all the corresponding data and exits the dialog box.

Abort closes the dialog box without any data checks.

From and *To* in conjunction with the *Auto Inc* field behave as a rudimentary automatic data generator, which updates the corresponding data arrays from the *From* index to the *To* index, with an array index step size of *Auto Inc.*

Each dialog box would group together related information (Appendix I shows the dialog boxes), representative of a single element of the corresponding data arrays. Later, each dialog box would be representative of the corresponding data types (as defined in Appendix F). The dialog boxes would require some modifications in order to support 3-D elements and problems.

6.5 Program Flowchart

The general program flow is shown in Figure 6.2. Flowing from the top left corner, the program starts by initialising all arrays and constants, and then entering the infinite command loop, which can only be terminated by 'Quit'ting the program.

The various command options, from top to bottom, are for manipulating files, problem definition and data entry, and solution generation and solution display.

Hooks have been provided for expansion of the software.

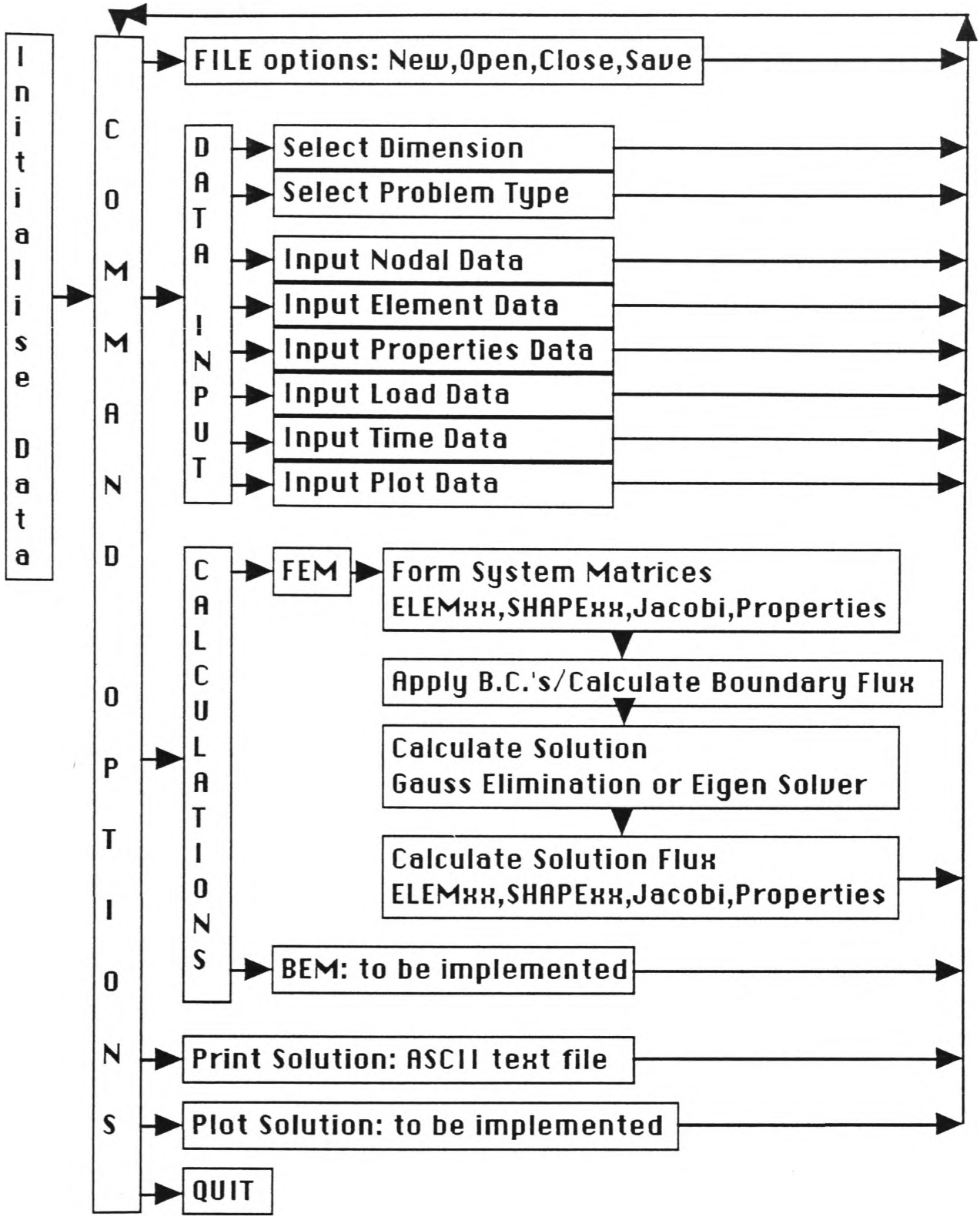


Figure 6.2: Block Diagram Flowchart of FAME.

Chapter 7 - Language Requirements

7.1 Introduction

FORTRAN was one of the first High Level Languages to be developed (1957). At that time, people were under the popular misconception that all problems were numerically intensive and hence their corresponding algorithmic solutions could easily be obtained by the use of some sort of massive 'number crunching' device, viz. the computer. As a result, more attention was given to the development of efficient matrix manipulation routines and functions that were capable of performing these calculations rather than to the development of a methodical and structured language.

In spite of these deficiencies, or maybe because of its non-methodical loose structure, FORTRAN has become one of the most widely used languages and is universally available on both large and small machines throughout the world. The ANSI X3.9-1978 Fortran77 (F77) standard ensured that all programs **conforming to the standard** would be portable.

Is 'FORT-RAN' a 'past-tense' language, as its very name seems to imply? Is there any other modern contender capable of filling its shoes? Why do people invariably turn to FORTRAN whenever numerical computations need to be done? Are there any special features of FORTRAN that make it indispensable in this regard?

To answer these questions, we decided to use M2 as a development language for a FEM software package. This would automatically eradicate any tendency to rely on Fortran's in-built features, and hence provide a better perspective from which to critically evaluate the software language features required to solve numerical problems.

7.2 Useful Language Features

7.2.1 Modularity

The EM's typically have a large amount of data to process. So it is advantageous to have similar data records stored and manipulated within a single module. It is preferable that modules do not export variables and that data-hiding is encouraged. This ensures that only certain predefined procedures can modify the value of a variable. This is to safeguard a module from incorrectly modifying another module's data and also aids in debugging as it localises the program code that is capable of changing the value of a variable.

Since we have separate modules, separate compilation would be in order, to permit debugging and testing to proceed in stages. Compiler checks would ensure that each program entity (library or user-defined) is used in a manner that is consistent with its definition. Any inconsistencies could be used to flag intermodule housekeeping errors that would otherwise be difficult to track down. Additionally, when linking, the compiler could check all dependent modules to ensure that all modules are up to date.

7.2.2 Control Structures

A versatile set of control structures, such as REPEAT - UNTIL; WHILE - DO; WITH - DO; CASE; and FOR loops [King88], permits easy readability and maintainability of the program (Appendix K) and also provides an easy way to implement structured programs. Use of these control structures would make use of the GOTO statement redundant and discourage programmers from lapsing into bad non-structured programming practices, leading to more coherent programs.

7.2.3 Data Types

For numerical computations, double precision complex number support would greatly expand the range of calculations that can be done, and hence, the types of problems that can be solved.

Additionally, a rich variety of data types would permit the programmer to custom build the relevant data structures or data variables required to solve a particular problem. The data type typically specifies the range of values that a particular data variable is permitted to assume. Hence, it provides information for:-

a) compile time checks, eg. data type mismatch in an operation,

b) data storage space,

c) run time checks, eg. array indices exceeding array bounds.

Record types permit great flexibility in custom definition of data structures to suit the application. Procedure types are useful for calling or choosing a different method of solution from a list of solution methods. Opaque types are always pointers to variables, whose structure is not visible outside the defining module. So the only way to use these variables is to use the associated manipulation routines in the definition modules. This is a very powerful tool for compelling modularity.

Chapter 6 (Section 6.4.2) and Appendix F indicate the various data structures that would be useful in implementing the program.

7.2.4 Type Checking

A multiplicity of data types is best handled with very strict type checking, such as limiting operators to only accept operands of similar type. An operation between dissimilar operands is flagged as an error and hence prevents easy mobility between dissimilar data types (as can occur by default in the C language). Likewise,

irregularities between the formal and actual parameters of functions and procedures are to be flagged as errors.

Automatic type conversions should not be permitted as they have enormous potential for erroneous type conversions. Instead, the programmer should ensure that variables are used consistently. Hence, the programmer is forced to evaluate the suitability of variables for a particular task/problem and so consciously makes a knowledgeable decision about the type of the data variable and whether type conversion is relevant in any particular case and hence may be permitted.

Of course, this may also imply that the programmer be sufficiently knowledgeable in the underlying machine dependent implementation of the various data types. (eg. an INTEGER implies varied data storage on different machine architectures and a type conversion from INTEGER to CHAR may potentially result in relevant information being truncated). This latter problem could be solved by preventing type conversions between data objects of different sizes.

7.2.5 Formal Typing

Formal declaration of variables is preferable to the implicit type specification mechanism. Firstly, the latter reduces the readability of the program, where variables just appear to come into existence out of nowhere. Secondly, proper initialisation of the implicit variables may be overlooked. The only advantage of implicit typing lies in the fact that repeated declaration of variables is unnecessary provided the programmer follows the implied convention.

7.2.6 Free Format in Program Source

A free format style of programming, in which the source code can occupy any position on the line, is preferable. This improves programmer flexibility by freeing him

from adhering to the fixed format style of programming, in which the position of the source code on the line has predefined connotations.

7.2.7 Global Data

Common global data can be accessed from any module that requires it. The data may be specified in any order. The compiler ensures that all references to the same variable name from different modules refer to the same data.

7.2.8 Dynamic Memory Allocation

Dynamic memory allocation results in a more flexible program that is capable of 'adjusting' itself to suit the run-time requirements of the problem at hand. However, the penalty incurred is a loss of time/money due to additional processing overhead and also the extra complexity introduced into the program implementation.

7.2.9 Multi Dimensional Arrays

These are a must for any computational software. However, different software languages implement them differently.

Some languages (like F77) support multi-dimensional arrays in which the leftmost subscript is varied the fastest (column major). This means that all the elements of the same column are stored sequentially in memory. On the other hand, other languages (like M2 and C) are row major and all the elements of the same row are stored contiguously in memory.

This seems an insurmountable problem. On initial inspection, it would appear that the same array is being accessed differently. On taking a closer look, it is evident that the difference is compiler generated. References by either language to the element

N(i,j) refer to the same element in the matrix. However, the physical storage position of that element in memory has changed.

This difference in the array representation has been exploited to implement different algorithms that are optimised to run on specific hardware architectures.

7.2.10 Open Array Parameters

Open array parameters permit a procedure to accept variable length arrays as the actual parameters. This concept is very useful as it permits the development of generalised array manipulation procedures, where the initial compile-time dimensions of the arrays are unknown, and are only passed to the procedure at run time.

7.2.11 Size of Variable and Procedure Names

These should be large enough to indicate the function of the variable or the procedure with reasonable clarity. Hence meaningful names can be declared rather than obscure cryptic ones.

7.2.12 Libraries

Numerical computations typically require large numbers of matrix operations to determine the solution. This has led to the development of extensive libraries of machine independent routines, which provide support for various kinds of matrix manipulations and numerical computations, including the solution of systems of linear equations and eigenvalue problems. However, most of these libraries are written in FORTRAN, as it is the de facto scientific computing language, eg. LAPACK [Anderson92].

7.2.13 File System and I/O functions

Versatile input/output functions are required. The input must be flexible enough to permit entry of the variety of input parameters. The output must be capable of visual display of the solution in a meaningful fashion.

Since it is data intensive, a powerful filing system is required to keep track of all the data - input parameters, intermediate calculations and the output results.

7.3 Discussion

7.3.1 Modula-2

The strongest points in favour of M2 are its modularity and strict type checking. Tests have shown [Griffiths91] that the latter results in much fewer errors, resulting in faster development time.

The present versions of M2 (Appendix A) supports all of the above features except 7.2.12 and 7.2.13 and complex numbers.

There are no libraries of numerical routines available!

A separate I/O statement is needed for each variable, thus causing I/O formatting to be very verbose. This tends to increase programmer errors, as the repetitive nature of the task results in programmer boredom and negligence. It may be possible to minimise this problem with the new ISO standard I/O libraries which are based on I/O channels.

The lack of complex number support places a few restrictions on the types of problems that can be solved; in particular, the solution of wave equations is not possible. One way of circumventing this set-back would be to define a complex number as a record of the form:-

```
Complex = RECORD
        Real : REAL;
        Imag : REAL;
END(*Complex*);
```

and define suitable +,-,*,/ operations for the set of complex numbers. In the absence of the operator overloading facility, this would necessitate separate functions for each of the above operations. This would render the program verbose, unwieldy and difficult to comprehend. Hence, it was decided to implement solutions involving complex numbers only in the next software version, using a later version of the compiler, which supports complex numbers.

7.3.2 Fortran77

F77's strong points are M2's weak points and vice versa.

There are powerful libraries available, eg. LAPACK [Anderson92].

It supports sophisticated formatted data processing commands which enable variables of different types to be transferred with a single command. The only disadvantage may be in remembering all the options and what they mean. However, this is amply offset by the elegant and concise representation of the input/output statements.

It has intrinsic compiler support for complex numbers.

7.4 Porting Problems

The design of software is typically an iterative process. No matter how good the design, 'bugs' invariable creep in. Additionally, there is an underlying complexity in the software design cycle that is not readily apparent at the superficial level. eg. 'critical' variables/records are overlooked, resulting in schedule delays due to redesign.

7.4.1 The Black-Box Approach

One basic assumption/mistake made in trying to speed up the software development/translation cycle was to assume the 'black box' approach. That is, if you have a line by line translation of F77 to M2, and all input parameters and returned results are catered for, then it is unnecessary to understand the inner detailed workings of any function or module. This assumption, though plausible, would fail if,

a) the implementation of variables was language or machine dependent, or,

b) the constructs of one language could not be translated to the other's.

Quite obviously, the resultant code would not be particularly efficient. However, at a later stage, a profiler could be used to target required code sections for optimisation.

In the present case, both a) and b) were absent. The main problems reared their heads elsewhere. This is where the black box approach broke down. It became imperative to delve inside functions and modules to discover the source of the error. This necessitated a study of:-

a) Gaussian Elimination (GE) to solve a system of n equations

b) Generalised Jacobi Algorithm Eigensolver (GJAE) for a system of n equations.

Appendix O details the problems faced while porting UNAFEM to FAME.

7.4.2 Conclusions - Porting Problems

Porting of F77 to M2 was not without pain. In spite of stringent checks (both human and computer), problems were caused by typographical errors and logic translation mistakes. Lack of detailed understanding on the author's part compounded the problem, leading to further delays.

This author spent 8 man-months, spread over a time period of approximately half a year, to develop the software up to its present level of performance. This

includes the solution of 1-D BDVP and EIGP. The results, though not relevant to the problem at hand, are shown in Appendix P.

Most of the time went into debugging problems that arose as a result of manual translation errors (Appendix O). No doubt, the final outcome was a clearer understanding of the algorithms! However, it would be preferable if manual errors could be eliminated altogether, as they introduce unnecessary bugs into debugged algorithms and code. An automatic F77 to M2 translator software, if available, could fix the problem.

7.5 Conclusions

The existence of a versatile library of functions is the strongest indicator in F77's favour. However, UNAFEM does not use any predefined library functions for the solution of the matrix equations. This would tend to imply that it is not such an important factor after all, as the required functions can be easily implemented if you have a copy of the algorithms.

This library could also explain the popularity of F77 over other languages for solving numerical problems. In general, people would prefer to have debugged library routines rather than waste time re-inventing the wheel.

On the other hand, M2 has many advantages in its favour. However, these are all general structured programming features. This can be explained by the fact that M2 has been designed by Niklaus Wirth as an exercise in compiler creation rather than for the solution of some specific problem. To target specific applications, additional features need to be incorporated. In this author's opinion, combination of the advantages of M2 with those of F77 would result in a very powerful language that would be well-suited to solve the task in hand.

Obviously, the programmer would have to be willing to invest a lot of time in debugging M2 functions. Though tedious, by making a start, this could be the beginning of a M2 library of numerical routines!

The primary reason for the choice of Modula 2 as the language of implementation was a specific request by my supervisor, who is interested in the application of ALGOL style languages for major software engineering projects.

The comparison of two languages (the source language, FORTRAN, and the target implementation language, Modula 2) was considered adequate to extract the features required for Finite Element Analysis.

Hence, no other languages were considered for the application.

Chapter 8 - Conclusions

This thesis shows the efficacy of echolocation in the natural world (bats) and recommends that studies be done to try to duplicate these effects by artificial means. Consequently, it evaluates a numerical approach to the study of ultrasonic transducers, comparing the individual approaches of FEM, BEM and WEEM, with various coupled approaches. It further proposes guidelines to develop a rudimentary Finite Element Analysis software package, FAME.

8.1 Just Right

The Element Methods appear to be just right for the task of transducer design and analysis. By changing the spatial orientation and inter-node distances in the system and by varying the properties of the elements, it is possible to simulate different solutions and determine the best transducer geometry and material composition for optimal output requirements. This is a cost-effective strategy, as we fabricate only those transducers that exhibit a 95% (or maybe more stringent) chance of meeting our requirements. Further, artificial environments can be created to model radiation and scattering problems. The solutions obtained can be examined in the lab by use of microphones controlled by a precision positioner, to see how theory and observations tally.

8.1.1 A Mix of Methods

The BEM should not be used to eliminate the FEM. Instead, it provides a powerful complement to extend the capabilities of the FEM.

In the ultrasonic frequency range, the number of nodes and elements increases very rapidly as the size of the problem is increased. This restricts the FEM to the solution of relatively small problems like the study of transducers and their resonant modes. Large problems like the study of the radiation field of a transducer will have to be solved by the BEM or the WEEM or the use of coupled methods (FEM/WEEM or FEM/BEM).

1-D piezoelectric models are inadequate for representing the interactions of all the piezoelectric modes. Preferably, for more accurate results, 2-D and 3-D models should be used. Also, the Helmholtz equation can not be used for the solution of piezoelectric systems. Rather, a wave equation formulation for anisotropic media, with its additional complexity and corresponding intricate FE formulation, would have to be developed and solved.

Most acoustic solutions to date are in the frequency domain. However, there are advantages to working in the time domain. Firstly, the solution has only real coefficients (no amplitude and phase!). Secondly, there is only one matrix inversion as opposed to the frequency domain, which incurs one inversion for each frequency! NIT is working on obtaining time domain solutions by using the Kirchoff Integral Equation to solve the Time Harmonic Wave equation.

8.2 An 'Element-Method' Software?

This thesis re-examines old well-worn techniques (FEM, BEM, FEM/FEM, FEM/BEM) and suggests that they should be coalesced into a single software package to enable users/researchers to benefit from the advantages of both techniques (and the ability to solve both near field and far field problems). This would basically permit the solution of problems that would be impossible or impractical by use of either method individually. In particular, the analysis of acoustics problems, and specifically, ultrasonic transducer design and analysis will benefit from this facility.

In this respect, Numerical Integration Technologies (NIT) has already made a start by developing SYSNOISE (over the past 6 years), which provides the above features in the solution of the Helmholtz equation (frequency domain).

Recent developments [Astley91, Coyette89, Coyette92-1, Coyette92-2] indicate that the WEEM may be a viable alternative (computationally more efficient) to the BEM in the far field. This requires further study and research.

8.3 Software Developed

This researcher has begun development on a software package called FAME. It is presently able to solve 1-D Boundary Value Problems and Eigen Problems. Undebugged code is available to solve 1-D Initial Boundary Value Problems and 2-D problems of all three kinds. Besides training this researcher in the mathematical basics behind these numerical methods, it has provided an easy to use tool for the solution of simple non-acoustic problems. It provides hooks that could be used to add relevant modules to extend this program to solve Dynamics problems and also 3-D applications. Incorporation of more efficient algorithms will result in it being able to solve larger problems.

8.3.1 Observations on Software development

One typical problem, characteristic of all software development projects, is for the programmer to admit the capability of software faults due to personal errors. Lack of this results in a wild-goose search for non-existent flaws, with a corresponding waste of time. It would be constructive to stress in all computer science courses that personal errors are one of the major reasons for debugging problems, thus instilling in programmers the concept that they can and do make mistakes.

The 'need-to-know' approach to software (the 'black-box' approach) is cautioned against, because, in the event of errors, the subsequent search for knowledge may require restructuring entire modules (and maybe even the overall code structure), causing more delays in the long term. This author believes that the knowledgeable ('learn-everything') approach is much more productive as compared to the use of a black box.

The graphs displayed by Word5 are sufficiently detailed to impart the required information. Consequently, writing software for graphic depiction of 1-D results was unproductive. Similarly, when expanding to 2-D and 3-D displays, it would be worthwhile examining if the results can be displayed suitably by some off-the-shelf package, rather than re-inventing the wheel. FORTRAN packages are available but are difficult to link to M2. Once again, the cost of the software will play a dominant role in the selection process. Writing custom display software should only be a last resort.

Towards the end of this project, I discovered Fortran90 (F90) [Adams92]. I have glanced at it only cursorily. However, it appears to have most, if not all, of the features that I've stated F77 as lacking. In this respect, there is one comment. Is it preferable to retain use of a package/language (F90) that has grown bulky due to the need to maintain compatibility with older implementations while still incorporating new features? Or should we advance with the times and utilise software that is compact and incorporates all the good features of the previous languages, while eliminating all/most of the bad? This author would prefer the latter!

8.4 Potential Applications

8.4.1 Ultrasonic Transducer Design

Elf Atochem Sensors [ElfAtochem92] provides standard and custom piezoelectric film elements. These could be used to test out our theories and, in the

future, maybe even build our own custom transducers. Experimental verification of the computed eigenmode shapes of the simulated transducers may not be possible as laser interferometers [Lerch90] are not accessible to this researcher! However, the radiation field generated by the transducers could be measured in the laboratory by using the precision positioner, which permits translation along the x and y axes, and rotation around all 3 axes.

8.4.2 Other

The substructuring facility [ANSYS5.0] can be utilised in the analysis and design of array transducers, where each element of the array is a substructure. Modelling bats' vocal tracts (near field problem) and how they interact with the environment (far field problem) could provide further insights into echolocation.

The BEM has potential to support inverse problems like acoustic imaging (sonagrams) [Ciskowski91] and so could be used for modelling the inverse problem in echolocation, ie. to determine the shape, orientation, texture, relative motion, etc. of any body by measuring the ultrasonic echoes it emits. Besides, there is large scope for the BEM in bioacoustics. Coupled with Probabilistic BEM [Daddazio91], it can be used to study and model various aspects of the human body.

As side effects, the same software techniques and analysis can be applied to develop other fields that depend on the piezoelectric effect, eg. smart-sensors and micro-motors.

8.5 Present Applications

In fact, the design of piezoelectric ultrasonic transducers is an ongoing area of research being tackled by a collaboration between a transducer developer (Thomson Sintra Pacific) and experts in numerical methods for Acoustics (NIT).

8.5.1 AMFIBIE

AMFIBIE [Migeot93] is a custom package developed by NIT for Thomson Sintra ASM to solve coupled problems involving non-homogeneous media. It combined the advantages of the FEM pressure formulation and the Direct BEM into a single package. All three acoustic BC's can be applied to both models, and coupling was ensured by special pressure and flow continuity conditions at the interface between the two models.

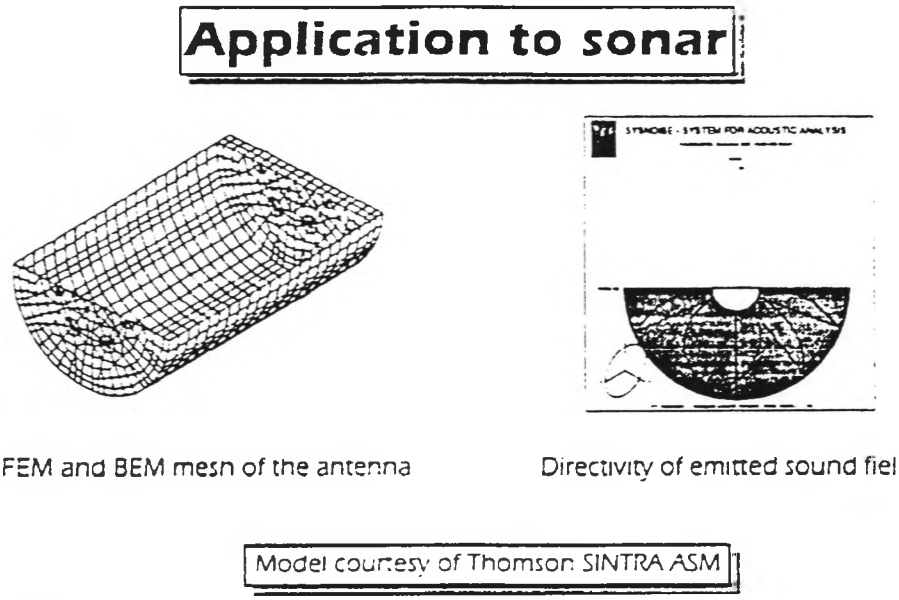


Figure 8.1: FEM-BEM solution for development of a phased array [Migeot93].

It was used to develop a phased array for a sonar application. A 3-D representation was used in which the antenna was modelled by 6-node wedges and 8-node hexahedrons for the FEM. Its surroundings (water) were modelled by 4-node surfaces for the BEM. The mesh used and the sound field obtained are shown in Figure 8.1.

8.5.2 Piezoelectric Development

Another TS ASM project involves piezoelectric transducer design. It aims to couple TS ASM's piezoelements to NIT's acoustic modelling package, SYSNOISE. The resultant system equations will be modified to eliminate electrical unknowns and then solved to get the resultant displacement.

8.6 In Conclusion

To model piezoelectric transducers in air for both transmission and reception, we require:

1) 2-D or 3-D piezoelectric elements

2) the piezoelectric constants for the materials of the transducer

3) a FEM model of the transducer

4) a BEM model of the waves in air

5) coupling between the FEM and BEM models

6) a numerical formulation to solve the coupled Dynamics equations, ie. second order time differential equations.

To apply this software to electrostatic and bimorph transducers would require additional work, including finding the transducer parameters.

It is evident that no available software package can directly support our needs. At present, 2 major companies, NIT and TSASM, are collaborating to develop software that can solve this problem for piezoelectric transducers in water.

We have studied the equations required to solve piezoelectric problems and identified elements that could be used to model piezoelectric transducers. 2-D axisymmetric elements are the minimum required to obtain models with a reasonable amount of accuracy. For more realistic models, 3-D elements must be used. Also, this

would extend the range of possible problems. In addition, wave envelope elements could be used for a complete FEM solution instead of using a BEM model for the waves in air.

We wrote rudimentary FEM software to enable us to estimate the time required to develop an Acoustic FEM package for piezoelectric transducers, and to evaluate the language factors involved.

Based on the work done and results obtained, we estimate that a minimum of three years will be needed to develop such a package. Subsequently, to accurately model the transducers, requires values for the transducer parameters; either from some reliable source (data from some transducer manufacturer) or else by actual measurements. In the latter case, considerable time and effort will be required, and, in this thesis, we have not studied the techniques involved.

Hence, we conclude that the overall work involved is way beyond a Masters Honours thesis. Instead, it is conservatively estimated to be at least one Ph.D's work, though it could possible be even two or more! For this reason, rather than being a solution, this thesis is an evaluation of what is required to solve the problem.

8.7 Tailpiece .....

Numerical modelling of acoustic transducers is a young field and still developing. It combines concepts from widely varying fields, such as crystallography, piezoelectricity, wave theory, materials manufacture and testing, transducer design, ultrasonics, echolocation, matrix algebra, numerical analysis and the various 'element' methods. A mastery of these fields plus a lot of software development will eventually result in a package suitable for our application.

References

[Abramowitz64] Abramowitz, Milton, and Stegun, Irene, 1964. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, U.S. Govt. Print. Off..

[Adams92] Adams, Jeanne C., et al, editors, 1992. *Fortran 90 handbook : complete ANSI/ISO reference*, McGraw-Hill Book Co.

[Altes88] Altes , R.A., 1988. 'Some Theoretical Concepts for Echolocation', *Animal Sonar - Processes and Performance, NATO ASI Series*, Plenum Press, p.725-752.

[Anderson92] Anderson, E., et al, 1992. *LAPACK : users' guide*, Society for Industrial and Applied Mathematics.

[ANSYS5.0] Swanson Analysis Systems, Inc., 1993. *ANSYS Revision 5.0 - Technical Description of capabilities*.

[Astley91] Astley, R.J., and Coyette, J.P., 1991. 'Applications of Wave Envelope Elements to Acoustical Scattering', *The Third IMACS International Symposium on Computational Acoustics*.

[Atkinson91] Atkinson, John, 1991. Lecture notes on 'Applied Acoustics - Short Course'.

[Babic91] Babic, Matjaz, 1991. 'A 200-KHz Ultrasonic Transducer Coupled to the Air with a Radiating Membrane', *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 38, no.3, May, p.252-255.

[Bathe82] Bathe, Klaus-Jurgen, 1982. *Finite element procedures in engineering analysis*, Prentice-Hall.

[Bekesy60] Bekesy, Georg Von, 1960. *Experiments in hearing*, McGraw-Hill.

[Benthien91] Benthien, George W., and Schenk, Harry A., 1991. 'Structural-Acoustic Coupling', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.109-129.

[Berlincourt64] Berlincourt, Don A., et al, 1964. 'Piezoelectric and Piezomagnetic Materials and Their Function in Transducers', *PHYSICAL ACOUSTICS : principles and methods, Vol.1, Part A*, Academic Press, p.169-267.

[Bernhard91] Bernhard, Robert J., and Smith, David C., 1991. 'Acoustic Design Sensitivity Analysis', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.77-93.

[Bettess77] Bettess, P., 1977. 'Infinite Elements', *International Journal of Numerical Methods in Engineering*, Vol.11, p.53-64.

[Biber80] Biber, C., et al, 1980. 'The Polaroid Ultrasonic Ranging System', *Audio Engineering Society Preprint*, 67th Convention.

[Boehm81] Boehm, Barry W., 1981. *Software engineering economics*, Prentice-Hall.

[Bjorno86] Bjorno, L, 1986. 'Ultrasonic Sensors', *Sensors and Sensory Systems for Advanced Robots, NATO ASI Series*, Springer-Verlag, p.309-339.

[Brebbia91] Brebbia, Carlos A., et al, 1991. 'Computational Formulation', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.14-60.

[Brebbia80] Brebbia, Carlos A, 1980. 'Fundamentals of Boundary Elements', *New developments in boundary element methods : Proceedings of the second International Seminar*, Computational Mechanics Publications, p.3-33.

[Burnett88] Burnett, David S., 1988. *Finite element analysis : from concepts to applications*, Addison-Wesley Pub. Co.

[Chadwick80] Chadwick, R.S., 1980. 'Studies in Cochlear Mechanics', *Lecture notes in Biomathematics; no. 43, Mathematical Modelling of the Hearing Process : Proceedings of the NSF-CBMS Regional Conference*, Springer-Verlag, p.9-54.

[Chapra88] Chapra, Steven C., and Canale, Raymond P., 1988. *Numerical methods for engineers*, 2e, McGraw-Hill.

[Ciskowski91] Ciskowski, Robert D., 1991. 'Applications in Bio-acoustics', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.147-175.

[Cook81] Cook, Robert D., 1981. *Concepts and applications of finite element analysis*, 2e, Wiley.

[Coyette92-1] Coyette, J.P., 1992. 'Modelling Radiation and Scattering from Vibrating Structures; A Comparison of Boundary Element and Wave Envelope Solutions', *17th International Seminar on Modal Analysis.*

[Coyette92-2] Coyette, J.P., 1992. 'Validation of a New Wave Envelope Formulation for handling Exterior Acoustic and Elasto-Acoustic Problems in the Frequency Domain', *DGLR/AIAA 14th Aeroacoustics Conference.*

[Coyette89] Coyette, J.P., et al, 1989. 'Numerical Analysis of Acoustic and Elasto-Acoustic Problems using Combined Finite Element and Boundary Element Methods', *7th International Modal Analysis Conference.*

[Daddazio91] Daddazio, Raymond P., and Ettouney, Mohammed M., 1991. 'Probabilistic Acoustic Analysis', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.95-108.

[Diekkamper83] Diekkamper, Rolf, 1983. 'Vectorised Finite Element Analysis of Nonlinear Problems in Structural Mechanics', *Parallel computing 83 : Proceedings of the International Conference on Parallel Computing*, Elsevier Science Pub. Co., p.293-298.

[Dongarra79] Dongarra, J.J., et al, 1979. *LINPACK : users' guide*, Society for Industrial and Applied Mathematics.

[Duller83] Duller, A.W.G., and Paddon, D.J., 1983. 'Processor Arrays and the Finite Element Method', *Parallel computing 83 : Proceedings of the International Conference on Parallel Computing*, Elsevier Science Pub. Co., p.131-136.

[ElfAtochem92] Elf Atochem Sensors, Inc., 1992. *Standard and Custom Piezo Film Components.*

[Estorff91] Estorff, O. von, 1991. 'Numerical Treatment of Acoustic Problems - An Overview', *20th International FE Congress.*

[Flandrin88] Flandrin, P., 1988. 'Time-Frequency Processing of Bat Sonar Signals', *Animal Sonar - Processes and Performance, NATO ASI Series*, Plenum Press, p.797-802.

[Geisler76] Geisler, C. Daniel, 1976. 'Mathematical Models of the Mechanics of the Inner Ear', *Handbook of Sensory Physiology, Vol.5, Auditory System, Part 3, Clinical and special topics*, Springer-Verlag, p.391-415.

[Gourlay73] Gourlay, Alexander Robertson, and Watson, G.A., 1973. *Computational methods for matrix eigenproblems*, Wiley.

[Griffiths91] Griffiths, L., 1991. 'Modula-2 is Three Times less Error Prone than C', Proceedings of the Second International Modula-2 Conference, September, p.332-338.

[Hunt82] Hunt, Frederick Vinton, 1982. *Electroacoustics : the analysis of transduction, and its historical background*, American Institute of Physics.

[Kay86] Kay, L., 1986. 'Airborne Ultrasonic Imaging of a Robot Workspace', *Robot Sensors*, Vol. 2, p.287-295.

[Khanna80] Khanna, S.M., and Leonard, D.G.B., 1980. 'Basilar Membrane Response measured in Damaged Cochleas of Cats', *Lecture notes in Biomathematics; no. 43, Mathematical Modelling of the Hearing Process : Proceedings of the NSF-CBMS Regional Conference*, Springer-Verlag, p.70-84.

[King88] King, K.N., 1988. *Modula-2: A Complete Guide*, D.C. Heath and Company.

[**Kratz83**] Kratz, Matthias, 1983. 'Some Aspects of Using Vector Computers for Finite Element Analyses', *Parallel computing 83 : Proceedings of the International Conference on Parallel Computing*, Elsevier Science Pub. Co., p.349-354.

[**Kuhl54**] Kuhl, W., et al, 1954. "Condenser Transmitters and Microphones with Solid Dielectric for Airborne Ultrasonics", ACUSTICA, Vol4, No5, p.519-532.

[**Kunkel90**] Kunkel, H.A., et al., 1990. 'Finite-Element Analysis of Vibrational Modes in Piezoelectric Ceramic Disks', *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 37, no.4, July, p.316-328.

[**Kuttruff91**] Kuttruff, Heinrich, 1991. *Ultrasonics fundamentals and Applications*, Elsevier Applied Science.

[**Lamancusa88**] Lamancusa, J.S., 1988. 'Ultrasonic Sensors', *International Encyclopedia of Robotics*, p.1563-1571..

[**Lerch90**] Lerch, Reinhard, 1990. 'Simulation of Piezoelectric Devices by 2-D and 3-D Finite-Elements', *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. 37, no.2, May, p.233-247.

[**Magori87**] Magori, V., and Walker, H., 1987. 'Ultrasonic Presence Sensors with Wide Range and High Local Resolution', *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, Vol. UFFC-34, no.2, March, p.202-211.

[**Martin86**] Martin, J.F., et al, 1986. 'Acoustic Imaging in Three Dimensions', *Sensors and Sensory Systems for Advanced Robots, NATO ASI Series*, Springer-Verlag, p.342-359.

[**McKerrow93**] McKerrow, P.J., 1993. 'Control of Ultrasonic Mapping by a Mobile Robot with an Expert System', *Robots for Competitive Industries : Proceedings of an International Conference of the Australian Robot Association and the International Federation of Robotics*, Mechanical Engineering Pub. Ltd., p.300-309.

[Mead91] Mead, C.A.,1991, 'Innovators', *IEEE Spectrum*, December.

[Meissner80] Meissner, Loren P., and Organick, Elliott I., 1980. *Fortran 77 : featuring structured programming, 3e*, Addison-Wesley Pub. Co.

[Migeot93] Migeot, Jean Louis, of Numerical Integration Technologies presented *A Short Course on Numerical Acoustics* at the Acoustics and Vibration Centre of the Australian Defence Force Academy, 28-29 June, 1993.

[Monchaud87] Monchaud, S., 1987. 'SONAIR Ultrasonic Range Finders', *Sensor Devices and Systems for Robotics, NATO ASI Series*, Springer-Verlag, p.112-126.

[Neuweiler89] Neuweiler, G., 1989. 'Foraging Ecology and Audition in Echolocating Bats', *TREE*, Vol.4, no.6, p.160-166.

[Ostergaard86] Ostergaard, D.F., and Pawlak, T.P., 1986. '3-D Finite-Elements for Analysing Piezoelectric Structures', *IEEE Ultrasonics Symposium*, p.639-644.

[Oxford64] Fowler, H.W., and Fowler, F.G., editors, 1964. *The Concise Oxford Dictionary of Current English, 5e*, Oxford at the Clarendon Press.

[Polaroid80] 'Technical Specifications for Polaroid Electrostatic Transducer', Polaroid Corporation, Cambridge, 1980.

[Pressman87] Pressman, Roger S., 1987. *Software engineering : a practitioner's approach, 2e*, McGraw-Hill.

[RoboScrub93] Robotic Floor Scrubber © Windsor Industries, Inc., U.S.A.

[Schoenwald87] Schoenwald, J.S., 1987. 'Acoustic Range Sensing for Robotic Control', *Sensor Devices and Systems for Robotics, NATO ASI Series*, Springer-Verlag, p.93-109.

[Seippel83] Seippel, Robert G., 1983. *Transducers, Sensors and Detectors*, Reston Publishing Co.

[Seybert93] Seybert, A.F., et al, 1993. 'A Coupled FEM/BEM for Fluid-Structure Interaction Using Ritz Vectors and Eigenvectors', *Transactions of the ASME - Journal of Vibration and Acoustics*, Vol.115, April, p.152-158.

[Shaw91] Shaw, Richard P., 1991. 'A Brief History of "Boundary Integral Equation/Element Methods" in Acoustics', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.1-11.

[Smith84] Smith, B.V., and Gazey, B.K., 1984. 'High Frequency Sonar Transducers: A Review of Current Practice', *IEE Proceedings*, Vol.131, Part F, no.3, June, p.285-297.

[Sommerville89] Sommerville, Ian, 1989. *Software engineering, 3e*, Addison-Wesley.

[Stakgold68] Stakgold, Ivar, 1968. *Boundary value problems of mathematical physics*, Vol.2, New York : Macmillan.

[Stebbins83] Stebbins, W.C., 1983. *The Acoustic Sense of Animals*, Harvard University Press.

[Strand5] Strand5 : user manual, reference guide, 1988, N.S.W. : G+D Computing.

[Suga90] Suga, N., 1990. 'Biosonar and Neural Computation in Bats', *Scientific American*, June, p.34-41.

[Suthers88] Suthers, R.A., 1988. 'The Production of Echolocation Signals by Bats and Birds', *Animal Sonar - Processes and Performance, NATO ASI Series*, Plenum Press, p.23-45.

[Suzuki91] Suzuki, Shinji, 1991. 'Applications in the Automotive Industry', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.131-146.

[Toshio91] Toshio, Terai, 1991. 'BEM Applications in Architectural Acoustics', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.193-223.

[Wilkinson63] Wilkinson, James Hardy, 1963. *Rounding errors in algebraic processes*, Her Majesty's Stationery Office.

[Wilkinson65] Wilkinson, James Hardy, 1965. *The algebraic eigenvalue problem*, Oxford;Clarendon.

[Wu91] Wu, Tim W., and Seybert, Andrew F., 1991. 'Acoustic Radiation and Scattering', *Boundary Element Methods in Acoustics*, Computational Mechanics Publications, p.61-76.

[Zienkiwicz83] Zienkiewicz, O. C., and Morgan, K., 1983. *Finite elements and approximation*, Wiley.

Appendix A - Tools Used

A.1 Hardware Used

1) MacintoshII © Apple Computer, Inc.

2) Quadra 950 © Apple Computer, Inc.

3) IBM PC/AT 386/486 © IBM Corporation

A.2 Software Used

1) Modula-2 V4.2 © 1990 p1 Gesellschaft für Informatik mbH.

2) Modula-2 V4.3.2 © 1991 p1 Gesellschaft für Informatik mbH.

3) MPW V3.2 © 1986-1991 Apple Computer, Inc.

4) Microsoft Word 5.1a © 1987-1992 Microsoft Corporation.

5) Aldus SuperPaint 3.0-1E © 1986, 1988-1991 Silicon Beach Software, Inc. .

6) Strand 5 © 1989 G&D Computing, Pty. Ltd.

7) SYSNOISE © Numerical Integration Technologies.

Appendix B - Types of Equations

This Appendix has been included to indicate the basic assumptions made in deriving the Helmholtz equations. In particular, note equations B.2.3 and B.2.5 that convert an anisotropic equation to an isotropic one.

Typical boundary conditions have been assumed and hence are not included.

B.1 1-D Problem

A typical 1-D Boundary Value Problem (BDVP) can be represented in mathematical notation by the following differential equation:-

$$-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}U(x)\right)+\beta(x)U(x)=f(x) \quad .. \quad .. \quad \text{B.1.1}$$

where

x is the independent variable

$U(x)$ is the unknown function

$\alpha(x),\beta(x)$ are known functions (usually representing physical properties of the system)

$f(x)$ is a known function (usually representing loads applied to the domain of the system)

B.2 2-D Problem

2-D problems can, in general, be represented by ***the quasiharmonic equation***:-

$$-\{\nabla\}^t([\alpha]\{\nabla U\})+\beta U = f \quad .. \quad .. \quad .. \quad .. \quad \text{B.2.1}$$

where

x, y are the dependent/independent variables

$U = U(x,y)$ is the unknown function

$\alpha(x,y) = [\alpha] = \begin{bmatrix} \alpha_x & \alpha_{xy} \\ \alpha_{xy} & \alpha_y \end{bmatrix}, \beta(x,y)$ are known functions (usually representing physical properties of the system)

$f(x,y)$ is a known function (usually representing loads applied to the domain of the system)

$\{\nabla\}^t = \left\{\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y}\right\}$ is the gradient operator for 2-D

A matrix expansion of B.2.1 results in

$$-\frac{\partial}{\partial x}\left(\alpha_x \frac{\partial U}{\partial x}\right)-\frac{\partial}{\partial x}\left(\alpha_{xy} \frac{\partial U}{\partial y}\right)$$
$$-\frac{\partial}{\partial y}\left(\alpha_{xy} \frac{\partial U}{\partial x}\right)-\frac{\partial}{\partial y}\left(\alpha_y \frac{\partial U}{\partial y}\right)+\beta U = f$$

.. .. B.2.2

Now $\alpha_{xy}(x,y) = 0 \Leftrightarrow [\alpha] = \begin{bmatrix} \alpha_x & 0 \\ 0 & \alpha_y \end{bmatrix}$ B.2.3

and putting B.2.3 into B.2.2 we get

$$-\frac{\partial}{\partial x}\left(\alpha_x \frac{\partial U}{\partial x}\right)-\frac{\partial}{\partial y}\left(\alpha_y \frac{\partial U}{\partial y}\right)+\beta U = f \quad .. \quad .. \quad \text{B.2.4}$$

which is the anisotropic equation solved by UNAFEM. It however restricts that the anisotropicity be only along two mutually perpendicular axes that are parallel to the x and y axes of the system.

Further $\alpha_x(x,y) = \alpha_y(x,y) \Leftrightarrow [\alpha] = \alpha \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ .. B.2.5

Substituting B.2.5 into B.2.4 we get

$$-\alpha \nabla^2 U + \beta U = f \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{B.2.6}$$

where $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$, is the Laplacian operator for 2-D.

We note that we have now reduced the material to isotropicity, ie. its properties are independent of direction.

Now if $\beta(x,y) = 0$, then B.2.6 becomes

$$\nabla^2 U = -\frac{f}{\alpha} = b \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{B.2.7}$$

which is ***Poisson's equation***, where b is some known function of space, or of the potential U itself, or includes time dependent terms.

Now if $f(x,y) = 0$, then B.2.7 becomes

$$\nabla^2 U = 0 \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{B.2.8}$$

which is ***Laplace's equation or the harmonic equation.***

If $b = \frac{1}{c^2}\frac{\partial^2 U}{\partial^2 t}$ then B.2.7 becomes

$$\nabla^2 U = \frac{1}{c^2}\frac{\partial^2 U}{\partial^2 t} \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{B.2.9}$$

which is the ***wave equation in the time domain.***,

where

U is a velocity potential (function of both position and time)

and c is the velocity of sound in the medium.

If U is a harmonic function, ie. $U(x,t) = \tilde{U}(x)e^{i\omega t}$ B.2.10

then B.2.9 becomes $\nabla^2\tilde{U} + k^2\tilde{U} = 0$ B.2.11

which is the ***Helmholtz equation (in the frequency domain),***

where

$k = \frac{\omega}{c}$ is the wave number

ω is the angular frequency.

We note that Helmholtz equation is a special case of the Wave Equation and that the time domain and frequency domain models are equivalent for harmonic cases.

B.3 3-D Problem

The quasiharmonic equation for 3-D problems is:-

$$-\{\nabla\}^t([\alpha]\{\nabla U\})+\beta U = f \quad \text{B.3.1}$$

where

x, y, z are the dependent/independent variables

$\alpha(x, y, z)$ is 3-by-3 matrix relating all 3 axes

$\{\nabla\}^t = \left\{\frac{\partial}{\partial x} \quad \frac{\partial}{\partial y} \quad \frac{\partial}{\partial z}\right\}$ is the gradient operator for 3-D

and all terms are functions of all 3 independent variables

In the 3-D domain, anisotropic materials will have interaction between all 3 axes. Only in the special case of isotropicity will the form reduce to that of equation B.2.7, with the corresponding Laplacian operator for 3-D.

B.4 Wave Equation for Acoustics

In the case of acoustics problems, the unknown function in the wave equation is the pressure field. So, setting $U = p$ in B.2.9 gives,

$$\nabla^2 p = \frac{1}{c^2}\frac{\partial^2 p}{\partial^2 t} \quad \text{B.4.1}$$

Assuming a steady state harmonic motion,

$$p(x, y, z, t) = \tilde{p}(x, y, z)e^{i\omega t} \quad \text{B.4.2}$$

and we get the Helmholtz equation

$$\nabla^2 \tilde{p} + k^2 \tilde{p} = 0 \quad \text{B.4.3}$$

Appendix C - FE Equations

This Appendix considers different types of problems and lists their PDEs, BCs and FEM formulations. It has been included to demonstrate how the underlying unseen complexity of the FE formulations increases as we move from 1-D to 2-D to 3-D.

C.1 1-D Boundary Value Problem (BDVP)

$$-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}U(x)\right)+\beta(x)U(x)=f(x) \quad .. \quad .. \quad \text{C.1.1}$$

with boundary conditions

$$U(x_a)=U_a \text{ or } \left[-\alpha(x)\frac{d}{dx}U(x)\right]_{x_a}=\tau_a \quad .. \quad .. \quad \text{C.1.2}$$

and

$$U(x_b)=U_b \text{ or } \left[-\alpha(x)\frac{d}{dx}U(x)\right]_{x_b}=\tau_b \quad .. \quad .. \quad \text{C.1.3}$$

Equivalent Matrix Representation

$$[K]\{u\}=\{F\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{C.1.4}$$

where

$$K_{ij}^{(e)} = \int^{(e)} \left(\frac{\partial \phi_i^{(e)}(x)}{\partial x} \alpha(x) \frac{\partial \phi_j^{(e)}(x)}{\partial x} \right) dx + \int^{(e)} \phi_i^{(e)}(x) \beta(x) \phi_j^{(e)}(x) dx$$

.. .. C.1.5

and

$$F_i^{(e)} = \int^{(e)} f(x) \phi_i^{(e)}(x) dx - \left[\left(-\alpha(x) \frac{d}{dx} \tilde{U}^{(e)}(x,a) \right) \phi_i^{(e)}(x) \right]^{(e)}$$

.. .. C.1.6

C.2 1-D Eigen Problem (EIVP)

$$-\frac{d}{dx}\left(\alpha(x) \frac{d}{dx} U(x) \right) + \beta(x) U(x) - \lambda \gamma(x) U(x) = 0 \qquad \text{C.2.1}$$

The system undergoes natural oscillations (forcing function = 0) and has the Equivalent Matrix Representation

$$[K]\{u\} - \lambda [M]\{u\} = 0 \qquad .. \quad .. \quad .. \quad .. \quad .. \quad .. \qquad \text{C.2.2}$$

C.3 1-D Initial Boundary Value Problem (IBVP)

$$\mu(x) \frac{\partial}{\partial t} U(x,t) - \frac{\partial}{\partial x}\left(\alpha(x) \frac{\partial}{\partial x} U(x,t) \right) + \beta(x) U(x,t) = f(x,t)$$

.. .. C.3.1

with

at xa,t>t0

$$U(x_a,t)=U_a(t) \text{ or } \left[-\alpha(x)\frac{\partial}{\partial x}U(x,t)\right]_{x_a}=\tau_a(t) \quad .. \quad \text{C.3.2}$$

and

at xb,t>t0

$$U(x_b,t)=U_b(t) \text{ or } \left[-\alpha(x)\frac{\partial}{\partial x}U(x,t)\right]_{x_b}=\tau_b(t) \quad .. \quad \text{C.3.3}$$

The initial conditions could be defined as

at t0,xa<x<xb

$$U(x,t_0)=U_0(x) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{C.3.4}$$

Equivalent Matrix Representation

$$[C]\{\dot{u}(t)\}+[K]\{u(t)\}=\{F(t)\} \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{C.3.5}$$

C.4 1-D Dynamics Problem (DYNP)

Wave Propagation and vibration problems are typically expressed by second order time partial differential equations as

$$\rho(x)\frac{\partial^2}{\partial t^2}U(x,t)+$$
$$\mu(x)\frac{\partial}{\partial t}U(x,t)-\frac{\partial}{\partial x}\left(\alpha(x)\frac{\partial}{\partial x}U(x,t)\right)+\beta(x)U(x,t)=f(x,t)$$

.. .. C.4.1

The initial conditions could be defined as

at t0,xa<x<xb

$$U(x,t_0)=U_0(x) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{C.4.2}$$

and

$$\left(\frac{\partial}{\partial t}U(x,t)\right)_{t_0}=V_0(x) \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{C.4.3}$$

Equivalent Matrix Representation

$$[M]\{\ddot{u}(t)\}+[C]\{\dot{u}(t)\}+[K]\{u(t)\}=\{F(t)\} \quad .. \quad .. \quad .. \quad .. \quad \text{C.4.4}$$

C.5 2-D Problems

As we move on to 2-D problems, we again note the similarity in the form of the PDEs. All parameters are functions of both x and y.

Boundary Value Problems

$$-\frac{\partial}{\partial x}\left(\alpha_x(x,y)\frac{\partial}{\partial x}U(x,y)\right)-\frac{\partial}{\partial y}\left(\alpha_y(x,y)\frac{\partial}{\partial y}U(x,y)\right)+\beta(x,y)U(x,y)=f(x,y)$$

.. .. C.5.1

Eigen Value Problems

$$-\frac{\partial}{\partial x}\left(\alpha_x(x,y)\frac{\partial}{\partial x}U(x,y)\right)-\frac{\partial}{\partial y}\left(\alpha_y(x,y)\frac{\partial}{\partial y}U(x,y)\right)$$
$$+\beta(x,y)U(x,y)-\lambda\gamma(x,y)U(x,y)=0$$

.. .. C.5.2

Initial Boundary Value Problems

$$\mu(x,y)\frac{\partial}{\partial t}U(x,y,t)$$
$$-\frac{\partial}{\partial x}\left(\alpha_x(x,y)\frac{\partial}{\partial x}U(x,y)\right)-\frac{\partial}{\partial y}\left(\alpha_y(x,y)\frac{\partial}{\partial y}U(x,y)\right)$$
$$+\beta(x,y)U(x,y)-\lambda\gamma(x,y)U(x,y)=0$$

.. .. C.5.3

The matrix equations have the same form as the corresponding 1-D problem, but the elements of the matrices are now double integrals, eg. element of stiffness matrix is

$$K_{ij}^{(e)}=\int^{(e)}\int\frac{\partial\phi_i^{(e)}}{\partial x}\alpha_x\frac{\partial\phi_j^{(e)}}{\partial x}dxdy+\int^{(e)}\int\frac{\partial\phi_i^{(e)}}{\partial y}\alpha_y\frac{\partial\phi_j^{(e)}}{\partial y}dxdy$$
$$+\int^{(e)}\int\phi_i^{(e)}\beta\phi_j^{(e)}dxdy$$

.. .. C.5.4

C.6 3-D Problems

The basic form of the PDEs and the matrix equations remain the same, though the complexity increases (triple integrals).

Appendix D - BE Formulations

This Appendix is included to demonstrate that the Galerkin method has been used to solve BEM problems. The isoparametric elements developed have a dual utility and can be utilised in both FEM and BEM formulations.

Brebbia [91] used the Galerkin method to develop the BIE for equations D.4 to D.7 over the 2-D domain Ω, which has two 1-D surfaces, respectively named Γ_1 and Γ_2, as shown in the figure,

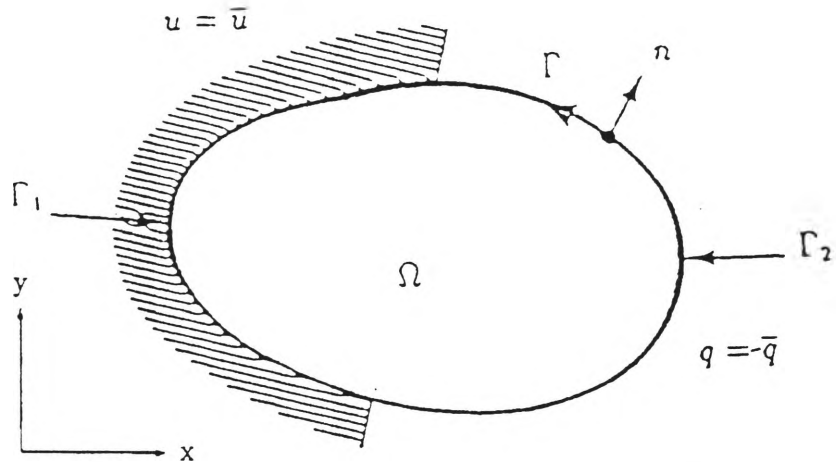


Figure D.1: Geometric definition for Laplace's equation [Brebbia91].

and subject to the conditions:-

1) Dirichlet or Essential

$$U = \overline{U} \text{ on } \Gamma_1 \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{D.1}$$

2) Neumann or Natural

$$q = \frac{\partial U}{\partial n} = \overline{q} \text{ on } \Gamma_2 \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{D.2}$$

3) Robin or Mixed

$$\alpha U + \beta q = \gamma \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{D.3}$$

Laplace's equation D.4

Poisson's equation D.5

The wave equation (Time Domain mode) D.6

Helmholtz equation (Frequency Domain mode) D.7

Here U is a function of position only and relates to acoustic pressure by

$$p = -i\omega\rho_0 U \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{D.8}$$

where ρ_0 is the density of the fluid.

He further derived the following 1-D boundary elements to solve 2-D problems:

a) Constant element

b) Isoparametric linear element

c) Isoparametric quadratic element

d) Isoparametric cubic element

We note that the equations of b), c) and d) are identical to those developed for the FEM. This is to be expected. However, a) has a different formulation.

We also note that the isoparametric elements developed for 2-D problems in FEM [Burnett88] can be used for the 3-D problems in BEM [Wu91]. Appendix N gives the equations for 1-D and 2-D isoparametric quadratic elements.

Appendix E - Detailed FEM Steps

This Appendix has been included for completeness purposes only. Most of the steps can be found in any standard text book describing the FEM.

The typical approach to the FEM can be split up into two broad sections (theoretical and numerical) sub-divided into twelve steps. We shall develop the steps assuming that we are solving the 1-D boundary value problem described by the following differential equation,

$$-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}U(x)\right)+\beta(x)U(x)=f(x) \qquad \text{E.1}$$

whose residual is

$$R(x,a)=-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)+\beta(x)\tilde{U}^{(e)}(x,a)-f(x) \qquad \text{E.2}$$

The general form of the element trial solution and its derivative can be written as

$$\tilde{U}^{(e)}(x,a) = \sum_{j=1}^{N} a_j \phi_j^{(e)}(x) \quad \text{E.3}$$

$$\frac{d}{dx}\tilde{U}^{(e)}(x,a) = \sum_{j=1}^{N} a_j \frac{d}{dx}\phi_j^{(e)}(x) \quad \text{E.4}$$

where

$\phi_j^{(e)}(x)$ is the j-th shape function (usually polynomial) for the e^{th} element

$a_j, \forall j = 1..N$ are unknown parameters (constants), or the degrees of freedom (DOF).

N is the local node numbering for the element.

E.1 - The Theoretical Section

The theoretical section describes the mathematical manipulations necessary to lay the ground-work for the numerical solution. It basically involves the underlying calculus that determine the expressions and formulae to be used in the calculation of the solution.

Step 1) Using Method of Weighted Residuals, we **derive the Galerkin residual equations**, one for each DOF a_i

$$\int^{(e)} R(x,a)\phi_i^{(e)}(x)dx = 0, \forall i = 1..N \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{E.5}$$

Putting E.2 in E.5, we get

$$\int^{(e)} \left(-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a) \right) + \beta(x)\tilde{U}^{(e)}(x,a) - f(x) \right) \phi_i^{(e)}(x)dx = 0 \quad .. \quad .. \quad .. \quad .. \quad \text{E.6}$$

Step 2) The general formula for integration by parts of the two functions u and v over the interval x_a to x_b is given by the relation

$$\int_{x_a}^{x_b} \frac{du}{dx} v dx = [uv]_{x_a}^{x_b} - \int_{x_a}^{x_b} u \frac{dv}{dx} dx$$

Integrating by parts the highest derivative term in the equation E.6 gives us

$$\int^{(e)}\left(-\frac{d}{dx}\left(\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)\right)\phi_i^{(e)}(x)dx =$$

$$\left[\left(-\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)\phi_i^{(e)}(x)\right]^{(e)} + \int^{(e)}\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\frac{d}{dx}\phi_i^{(e)}(x)dx \quad .. \quad .. \quad \text{E.7}$$

Hence, substituting E.7 into E.6 above, and the Galerkin residual equations becomes

$$\int^{(e)}\left(\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\frac{d}{dx}\phi_i^{(e)}(x)+\beta(x)\tilde{U}^{(e)}(x,a)\phi_i^{(e)}(x)\right)dx =$$

$$\int^{(e)} f(x)\phi_i^{(e)}(x)dx - \left[\left(-\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)\phi_i^{(e)}(x)\right]^{(e)} \qquad \forall i = 1..N \quad .. \quad .. \quad .. \quad \text{E.8}$$

Step 3) Substitute the general form of the element trial solution into the integrals in the residual equations. Hence **get the element equations.**

Substituting E.3 and E.4 into equation E.8 gives

$$\int^{(e)}\left(\alpha(x)\left(\sum_{j=1}^{N}a_j\frac{d}{dx}\phi_j^{(e)}(x)\right)\frac{d}{dx}\phi_i^{(e)}(x)+\beta(x)\left(\sum_{j=1}^{N}a_j\phi_j^{(e)}(x)\right)\phi_i^{(e)}(x)\right)dx =$$

$$\int^{(e)} f(x)\phi_i^{(e)}(x)dx-\left[\left(-\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)\phi_i^{(e)}(x)\right]^{(e)} \quad .. \quad .. \quad .. \quad .. \quad \text{E.9}$$

$$\sum_{j=1}^{N}\left(\int^{(e)}\left(\left(\frac{d}{dx}\phi_i^{(e)}(x)\right)\alpha(x)\left(\frac{d}{dx}\phi_j^{(e)}(x)\right)\right)dx+\int^{(e)}\left(\phi_i^{(e)}(x)\beta(x)\phi_j^{(e)}(x)dx\right)\right)a_j =$$

$$\int^{(e)} f(x)\phi_i^{(e)}(x)dx-\left[\left(-\alpha(x)\frac{d}{dx}\tilde{U}^{(e)}(x,a)\right)\phi_i^{(e)}(x)\right]^{(e)}$$

$$\forall i = 1..N \quad .. \quad .. \quad .. \quad .. \quad \text{E.10}$$

Note that E.10 represents the set of the element equations for a typical element. It can be represented in the conventional matrix format by the notation:

$$\begin{bmatrix} K_{11}^{(e)} & K_{12}^{(e)} & . & . & K_{1N}^{(e)} \\ K_{21}^{(e)} & K_{22}^{(e)} & . & . & K_{2N}^{(e)} \\ . & . & . & . & . \\ . & . & . & . & . \\ K_{N1}^{(e)} & K_{N2}^{(e)} & . & . & K_{NN}^{(e)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ . \\ . \\ a_N \end{bmatrix} = \begin{bmatrix} F_1^{(e)} \\ F_2^{(e)} \\ . \\ . \\ F_N^{(e)} \end{bmatrix} \quad \text{E.11}$$

where

$$K_{ij}^{(e)} = \int^{(e)} \left(\frac{\partial \phi_i^{(e)}(x)}{\partial x} \alpha(x) \frac{\partial \phi_j^{(e)}(x)}{\partial x} \right) dx + \int^{(e)} \phi_i^{(e)}(x) \beta(x) \phi_j^{(e)}(x) dx$$

$$F_i^{(e)} = \int^{(e)} f(x) \phi_i^{(e)}(x) dx - \left[\left(-\alpha(x) \frac{d}{dx} \tilde{U}^{(e)}(x,a) \right) \phi_i^{(e)}(x) \right]^{(e)}$$

Step 4) **Develop specific expressions for the shape functions** (element trial functions). For the standard elements, these are usually Lagrange interpolation polynomials with the values of the functions at the nodes defined to equal 1. Hence, for a linear interpolation between 2 nodes, we have

$$\phi_1(x_a) = 1 \quad \phi_2(x_a) = 0$$
$$\phi_1(x_b) = 0 \quad \text{and} \quad \phi_2(x_b) = 1 \qquad \text{E.12}$$

This can be clearly understood from Figure E.1 which graphically depicts the relationship.

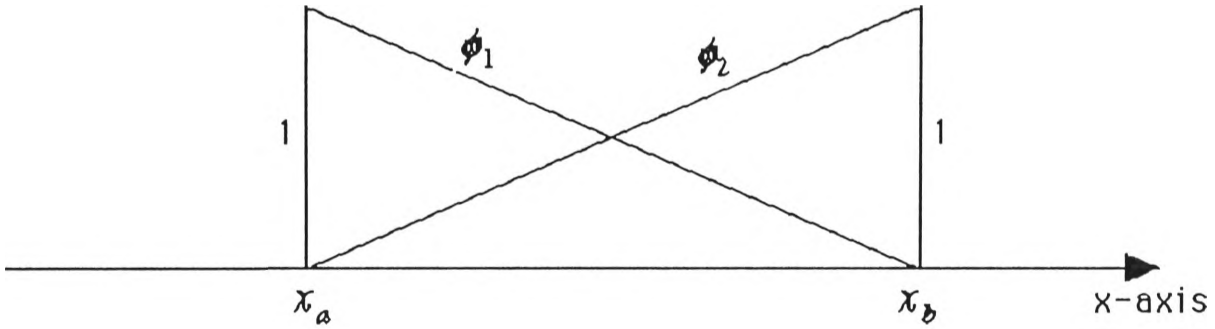


Figure E.1

In general, for N nodes, we have the following definition:-

$$\phi_j(x_i) = \delta_{ji} = 1, j = i$$
$$\phi_j(x_i) = \delta_{ji} = 0, j \neq i \qquad \text{E.13}$$

In the case of isoparametric elements, the development is nearly the same. In fact, if the elements are strictly symmetric, then the trial functions are identical in both cases.

Note that this step is Element dependent.

Step 5) Put the shape functions into the element equations and **transform the integrals into a form suitable for numerical evaluation**.

This requires that the integrals in $K_{ij}^{(e)}$ and $F_i^{(e)}$ be evaluated analytically. This has two aspects.

Firstly, we need to know the functions that describe $\alpha(x)$, $\beta(x)$ and $f(x)$. These functions may be constants, or they may be low order polynomial approximations.

Secondly, we need flexibility in order to be able to have a general problem solver.

Hence, we assume that the above values are constants or simple functions over the entire element, and let the user input them. Then their values are calculated by Gauss-Legendre quadrature.

Note that this step is Element dependent.

Step 6) **Prepare expressions for the flux** using the shape functions. ie. $flux = \tilde{\tau}^{(e)}(x_i) = \left[-\alpha(x)\frac{d\tilde{u}^{(e)}}{dx}\right]_{x_i}$

Note that this step is element dependent.

E.2 - The Numerical Section

The numerical section does the actual number-crunching.

Step 7) This step handles all Pre-Processing, requiring **entry of all input data**, including

a) Control parameters, such as type and dimension of problem

b) geometric data, such as node numbers and co-ordinates

c) physical data, such as values of $\alpha(x)$ and $\beta(x)$

d) load data, such as $f(x)$.

Step 8) **Assembly of the System Equations** involves two steps. First, an evaluation of the interior terms in the element equations for each element individually. Then all corresponding terms in the element equations referring to the same node are coalesced into a single term for that particular node (DOF).

eg. Assuming a simple mesh of 2 linear elements (element 1 consisting of nodes 1 and 2; element 2 consisting of nodes 2 and 3), we would end up with the following banded matrix/system equations

$$\begin{bmatrix} K_{11}^{(1)} & K_{12}^{(1)} & \\ K_{21}^{(1)} & K_{22}^{(1)}+K_{22}^{(2)} & K_{23}^{(2)} \\ & K_{32}^{(2)} & K_{33}^{(2)} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} FI_1^{(1)} \\ FI_2^{(1)}+FI_2^{(2)} \\ FI_3^{(2)} \end{bmatrix} + \begin{bmatrix} \tilde{\tau}^{(1)}(x_1) \\ -\tilde{\tau}^{(1)}(x_2)+\tilde{\tau}^{(2)}(x_2) \\ -\tilde{\tau}^{(2)}(x_3) \end{bmatrix} \quad \text{E.14}$$

where

the superscript $^{(e)}$ (in all arrays) refers to the element number

the subscript i in a_i, FI_i, τ_i refers to the node number

FI_i represents the interior load

In general, for a system having N nodes, we would end up with the familiar FEM matrix equation:

$$[K][a]=[F] \quad \text{E.15}$$

where

$[K]$ is a banded symmetric stiffness matrix of order N

$[a],[F]$ are column matrices of order N representing unknowns and loads respectively

Step 9) **Apply the boundary conditions.**

The natural boundary conditions (BC's) and inter-element boundary conditions (IBC's) are unconstrained. So, they would be directly added to their corresponding matrices, $[F]$.

The essential boundary conditions (EBC's) are constrained. So suitable constraints would have to be imposed on the system equations. In general, some a_i would be set equal to the EBC's. This would imply modifying the $[F]$ and $[K]$ matrices.

Step 10) **Solve the system equations**, typically by Gaussian Elimination, to find the values for all a_i, other than those that have been assigned values by the EBC's.

Step 11) **Evaluate the flux** and/or other related quantities in all or some of the elements by re-substituting for the known a_i.

Step 12) **Display the solution and evaluate its accuracy**. If need be, a partially or completely refined mesh is constructed (p-refinement, which uses higher order polynomials; or h-refinement, which uses smaller elements; or both) to generate a more accurate solution in some particular region or over the entire domain of the problem respectively. (And we loop back to Step 1)

E.3 - Other Types of Problems in 1-D

Appendix C indicates that additional matrices $[C]$ and $[M]$ come into play when we consider the solutions to the Eigenproblem (EIGP), the Initial Boundary Value problem (IBVP) and the Dynamics problem (DYNP). Their development is along similar lines to the 12 steps outlined above.

EIGP requires FEM matrix assembly and an eigensolver to obtain the solution. IBVP requires FEM matrix assembly and GE to obtain the solution and Finite Differences (FD) for the time iterations. DYNP needs to develop some form of time stepping.

Appendix F - Data Types

This Appendix lists the various data types defined during the design of FAME.

F.1 General Types

```
Dimension =
    (DIM1,          (* one dimension problem *)
    DIM2,           (* two dimension problem *)
    DIMAXI,         (* axisymmetric problem *)
    DIM3);          (* three dimension problem *)

Problem =
    (BDVP,          (* boundary value problem *)
    EIGP,           (* eigen problem *)
    IBVP,           (* initial boundary value problem *)
    DYNP);          (* dynamics problem *)
```

```
(* These are common variables that specify a particular problem *)
System = RECORD
    Stitle : Str255;                    (* title of problem *)
    Sdimen : Dimension;                 (* no. of dimensions of problem *)
    Sprob : Problem;                    (* type of problem *)

    Snumnp : INTEGER;                   (* no. of nodal points *)
    Snumel : INTEGER;                   (* no. of elements *)
    Snumpp : INTEGER;                   (* no. of physical properties *)
    Snumilc : INTEGER;                  (* no. of internal load conditions *)
    Snumebc : INTEGER;                  (* no. of essential BC's *)
    Snumnbc : INTEGER;                  (* no. of natural BC's *)
    Snumtdata : INTEGER;                (* no. of time stepping data *)
    Snumpdata : INTEGER;                (* no. of plot data *)

    Seigen_calc_done : BOOLEAN;
    Seigen_solution_found : BOOLEAN;
    Sfem_calc_done : BOOLEAN;
    Sfem_solution_found : BOOLEAN;
END;
```

F.2 Input Types

```
(* Each Node specifies the co-ordinates of one of the nodes of the problem mesh *)
Node = RECORD
    number : INTEGER;                    (* number of the node *)
    x : REAL;                            (* x coordinate of the node *)
    y : REAL;                            (* y coordinate of the node *)
    z : REAL;                            (* z coordinate of the node *)
    node_valid : BOOLEAN;                (* validity of the node *)
    next_node : POINTER TO Node;         (* pointer to next node *)
    prev_node : POINTER TO Node;         (* pointer to previous node *)
END;

(* Each Element describes an element of the problem mesh *)
Element = RECORD
    number : INTEGER;                    (* number of the element *)
    type : INTEGER;                      (* type of the element *)
    gauss : INTEGER;                     (* no. of Gauss points of elem. *)
    prop : INTEGER;                      (* property no. of the element *)
    nodes : ARRAY [1..8] OF INTEGER;     (* numbers of all the nodes belonging to
                                            this element*)
    next_elem : POINTER TO Element;      (* points to next Element *)
    prev_elem : POINTER TO Element;      (* points to previous Element *)
END;
```

```
(* Each Property specifies coefficients of quadratic functions that describe the physical
properties of a particular material/media *)
Property = RECORD
    number : INTEGER;                    (* number of this property *)
    alphax : ARRAY [1..6] OF REAL;
    alphay : ARRAY [1..6] OF REAL;
    beta : ARRAY [1..6] OF REAL;
    gamma : ARRAY [1..6] OF REAL;
    mu : ARRAY [1..6] OF REAL;
    next_prop : POINTER TO Property;     (* pointer to next Property *)
    prev_prop : POINTER TO Property;     (* pointer to previous Property *)
END;

(* Each ILC specifies one of the internal loads of the problem *)
ILC = RECORD
    number : INTEGER;                    (* number of this ILC *)
    node : INTEGER;                      (* ILC is applied to this node *)
    fint :  REAL;                        (* load value of this ILC *)
    ilc_pres : BOOLEAN;                  (* validity of this ILC *)
    next_ilc : POINTER TO ILC;           (* pointer to next ILC *)
    prev_ilc : POINTER TO ILC;           (* pointer to previous ILC *)
END;
```

```
(* Each EBC specifies one of the essential BCs of the problem *)
EBC = RECORD
    number : INTEGER;                (* number of this EBC *)
    iuebc : INTEGER;                 (* EBC is applied to this node *)
    uebc :  REAL;                    (* load value of this EBC *)
    ebc_pres : BOOLEAN;              (* validity of this EBC *)
    next_ebc : POINTER TO EBC ;      (* pointer to next EBC *)
    prev_ebc : POINTER TO EBC ;      (* pointer to previous EBC *)
END;

(* Each NBC specifies one of the natural BCs of the problem *)
NBC = RECORD
    number : INTEGER;                (* number of this NBC *)
    inp : INTEGER;                   (* NBC is applied to this node *)
    jnp : INTEGER;                   (* NBC is applied to this node *)
    knp : INTEGER;                   (* NBC is applied to this node *)
    taunbc : REAL;                   (* load value of this NBC *)
    nbc_pres : BOOLEAN;              (* validity of this NBC *)
    next_ebc : POINTER TO NBC;       (* pointer to next NBC *)
    prev_ebc : POINTER TO NBC;       (* pointer to previous NBC *)
END;
```

```
(* Each Graph specifies display parameters for the problem *)
Graph = RECORD
    number : INTEGER;                (* number of this graph *)
    variable : INTEGER;              (* variable that it describes *)
    Gxmax_needed : REAL;             (*input*)
    Gxmin_needed : REAL;             (*input*)
    Gx_num_divs : INTEGER;           (*input*)
    Gx_scale_factor : REAL;          (*calculated*)
    Gx_scale : INTEGER;              (*calculated*)
    Gx_string : Str63;
    Gymax_needed : REAL;             (*input*)
    Gymin_needed : REAL;             (*input*)
    Gy_num_divs : INTEGER;           (*input*)
    Gy_scale_factor : REAL;          (*calculated*)
    Gy_scale : INTEGER;              (*calculated*)
    Gy_string : Str63;
    next_Graph : POINTER TO Graph;   (* pointer to next Graph *)
    prev_Graph : POINTER TO Graph;   (* pointer to previous Graph *)
END;
```

F.3 Output Types

```
(* Each NodeSolution specifies the function value at the node *)
NodeSolution = RECORD
    iu : INTEGER;
    tauxnp : REAL;
    tauynp : REAL;
END;

(* Each ElementFlux specifies values of the flux/gauss points for the given element *)
ElementFlux = RECORD
    xgp : ARRAY [1..4] OF REAL;
    ygp : ARRAY [1..4] OF REAL;
    tauxgp : ARRAY [1..4] OF REAL;
    tauygp : ARRAY [1..4] OF REAL;
END;

(* Each EigenSolution specifies an eigenvalue and its eigenvector *)
EigenSolution = RECORD
    eigsml : REAL;                    (* specify eigenvalue *)
    vecsml : ARRAY [1..61] OF REAL;   (* specify eigenvector *)
END;
```

Appendix G - Variables Used

This Appendix has been included to indicate the actual variables used in the implementation of FAME.

```
DEFINITION MODULE FECommon;
(*Robert Pierre C. D'Souza; Feb 1, 1993*)

TYPE (* some necessary types ; RPD; Feb 1, 1993 *)
Print_Choice = (Print, No_Print);              (* To print or not to print *)
Jacobi_Choice = (Complete_Solution, Forward_Reduction);
                                              (* Complete Solution of matrix;
                                              or only Forward Reduction *)
Dimension = (DIM1,                            (* one dimension problem *)
             DIM2,                            (* two dimension problem *)
             DIM3);                           (* three dimension problem *)
Option =     (FORM,                           (* form element matrices *)
             FLUX);                           (* calculate flux  *)
Problem = (BDVP,                              (* boundary value problem *)
             EIGP,                            (* eigen problem *)
             IBVP);                           (* init. boundary value prob. *)

VAR (translated/ported from those listed in UNAFEM [Burnett88])
dimen : Dimension;
prob : Problem;
title : Str255;
```

```
numnp,                          (* no. of nodal points *)
numel,                          (* no. of elements *)
numpp,                          (* no. of physical properties *)
numilc,                         (* no. of internal load conditions *)
numebc,                         (* no. of essential BC's *)
numnbc,                         (* no. of natural BC's *)
numtdata,                       (* no. of time data ; RPD; Feb 12, 93 *)
numpdata,                       (* no. of plot data ; RPD; Feb 12, 93 *)
mband                           (* half bandwidth *)
        : INTEGER;

maxnp,                          (* max no. of nodal points *)
maxel,                          (* max no. of elements *)
maxpp,                          (* max no. of physical properties *)
maxil,                          (* max no. of internal load conditions *)
maxebc,                         (* max no. of essential BC's *)
maxnbc,                         (* max no. of natural BC's *)
maxtdata,                       (* max no. of time data ; RPD; 12/2/93 *)
maxpdata,                       (* max no. of plot data ; RPD; 12/2/93 *)
maxbnd                          (* max half bandwidth *)
        : INTEGER;

ntype : ARRAY [1..200] OF INTEGER;
ngpint : ARRAY [1..200] OF INTEGER;
nphys : ARRAY [1..200] OF INTEGER;
icon : ARRAY [1..8],[1..200] OF INTEGER;

k : ARRAY [1..300],[1..50] OF REAL;
f : ARRAY [1..300] OF REAL;
```

```
a : ARRAY [1..300] OF REAL;

keig : ARRAY [1..61],[1..61] OF REAL;
meig : ARRAY [1..61],[1..61] OF REAL;

ck : ARRAY [1..300],[1..50] OF REAL;
anm1 : ARRAY [1..300] OF REAL;
fnm1 : ARRAY [1..300] OF REAL;
fn : ARRAY [1..300] OF REAL;
fo : ARRAY [1..300] OF REAL;
clump : ARRAY [1..300] OF REAL;
delu : ARRAY [1..300] OF REAL;
uo : ARRAY [1..300] OF REAL;
keffdu : ARRAY [1..300] OF REAL;
keffuo : ARRAY [1..300] OF REAL;

intrvl,nsteps,ntime : INTEGER;
time,dt,theta,ao : REAL;

nnodes : ARRAY [1..30] OF INTEGER;
intdef : ARRAY [1..30] OF INTEGER;
ioptau : ARRAY [1..30] OF INTEGER;

ke : ARRAY [1..8],[1..8] OF REAL;
fe : ARRAY [1..8] OF REAL;
me : ARRAY [1..8],[1..8] OF REAL;
ce : ARRAY [1..8],[1..8] OF REAL;

xi, eta, jac :REAL;
```

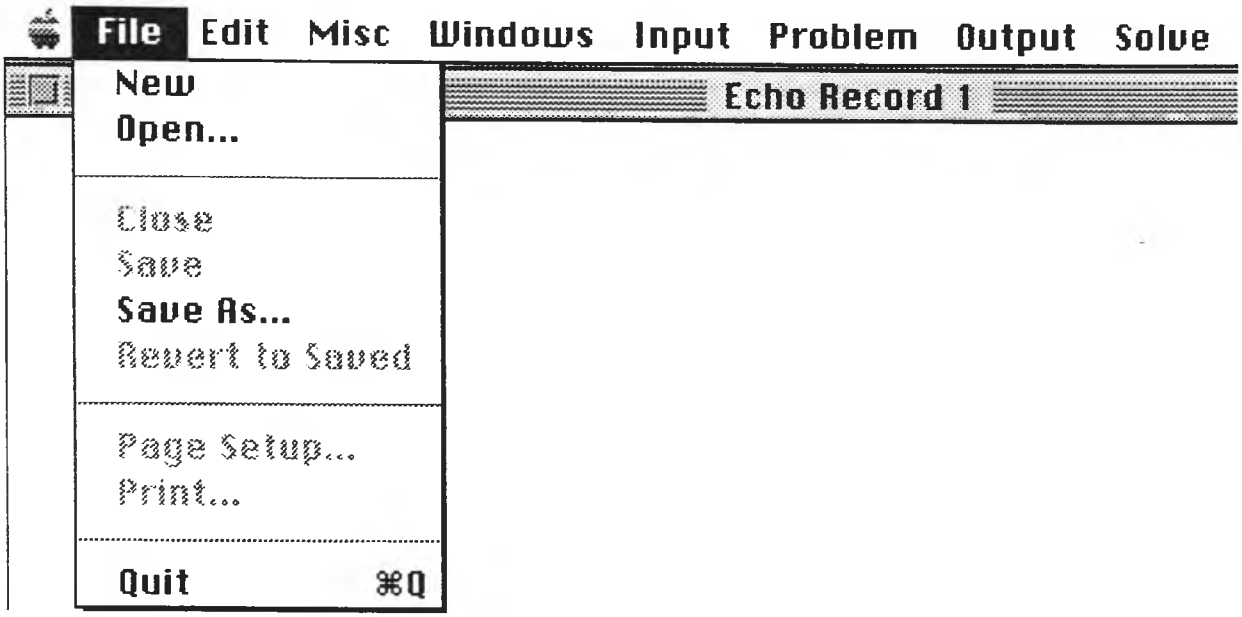
```
jacinv : ARRAY [1..2],[1..2] OF REAL;
xx, yy, phi : ARRAY [1..8] OF REAL;
dphdx : ARRAY [1..8] OF REAL;
dphdy : ARRAY [1..8] OF REAL;
dphdxi : ARRAY [1..8] OF REAL;
dphdet : ARRAY [1..8] OF REAL;
nno : INTEGER;

alphax : ARRAY [1..6],[1..20] OF REAL;
alphay : ARRAY [1..6],[1..20] OF REAL;
beta : ARRAY [1..6],[1..20] OF REAL;
gamma : ARRAY [1..6],[1..20] OF REAL;
mu : ARRAY [1..6],[1..20] OF REAL;

x : ARRAY [1..300] OF REAL;
y : ARRAY [1..300] OF REAL;
node_valid : ARRAY [1..300] OF BOOLEAN; (* RPD; Feb 1, 93 *)

fint : ARRAY [1..200] OF REAL;
ilcs_pres : ARRAY [1..300] OF BOOLEAN; (* RPD; Feb 1, 93 *)

iu : ARRAY [1..300] OF INTEGER;
tauxnp : ARRAY [1..300] OF REAL;
tauynp : ARRAY [1..300] OF REAL;

xgp : ARRAY [1..4],[1..200] OF REAL;
ygp : ARRAY [1..4],[1..200] OF REAL;
tauxgp : ARRAY [1..4],[1..200] OF REAL;
tauygp : ARRAY [1..4],[1..200] OF REAL;
```

```
gp : ARRAY [1..5],[1..5] OF REAL;
wt : ARRAY [1..5],[1..5] OF REAL;

uc : ARRAY [1..200] OF REAL;
xc : ARRAY [1..200] OF REAL;
yc : ARRAY [1..200] OF REAL;
tauxc : ARRAY [1..200] OF REAL;
tauyc : ARRAY [1..200] OF REAL;

ntip : ARRAY [1..7] OF INTEGER;
gpxi : ARRAY [1..7],[1..7] OF REAL;
gpet : ARRAY [1..7],[1..7] OF REAL;
wght : ARRAY [1..7],[1..7] OF REAL;

tr1 : ARRAY [1..6],[1..4] OF REAL;
tr2 : ARRAY [1..8],[1..4] OF REAL;

iuebc : ARRAY [1..100] OF INTEGER;
uebc : ARRAY [1..100] OF REAL;

inp : ARRAY [1..100] OF INTEGER;
jnp : ARRAY [1..100] OF INTEGER;
knp : ARRAY [1..100] OF INTEGER;
taunbc : ARRAY [1..100] OF REAL;

xvec : ARRAY [1..61],[1..61] OF REAL;
eigv : ARRAY [1..61] OF REAL;
```

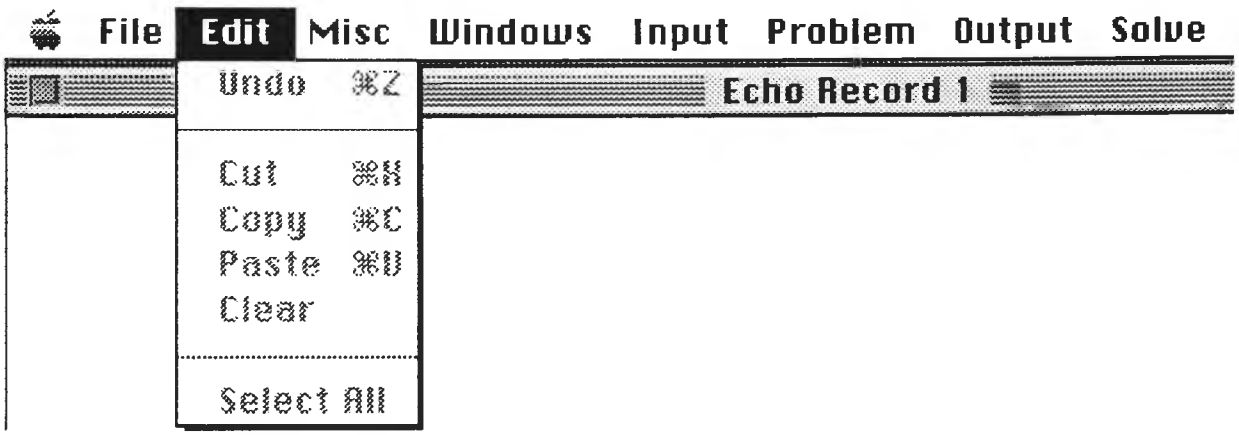
```
time_data : INTEGER; (* RPD; Feb 1, 93 *)
tplot : ARRAY [1..501] OF REAL;
ftplot : ARRAY [1..3],[1..501] OF REAL;
nnodei : ARRAY [1..3] OF INTEGER;

eigsml : ARRAY [1..5] OF REAL;
vecsml : ARRAY [1..61],[1..5] OF REAL;
numeig : INTEGER;

(* Some additional variables needed; RPD; Feb 1, 93 *)
(* Specifies if "batch_mode"(TRUE) or "single_prob_mode"(FALSE) *)
        batch_mode : BOOLEAN;
(* Specify output options; "enabled"(TRUE) or "disabled"(FALSE) *)
        printer_enabled : BOOLEAN;
        file_enabled : BOOLEAN;
        screen_enabled : BOOLEAN;
(* Specify calc options; "done"(TRUE) or "not done"(FALSE) *)
        eigen_calc_done : BOOLEAN;
        eigen_solution_found : BOOLEAN;
        fem_calc_done : BOOLEAN;
        fem_solution_found : BOOLEAN;
(* this is a blank string to clear any strings *)
        blank_str63 : Str63;
        blank_str : Str255;
END FECommon.
```

Appendix H - All Menus

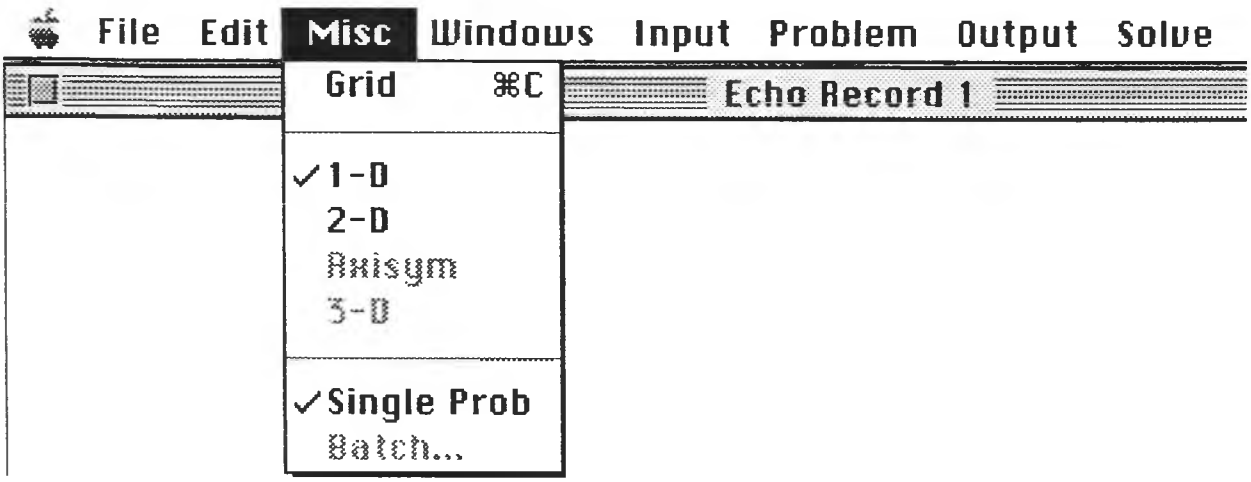
H.1- File Menu



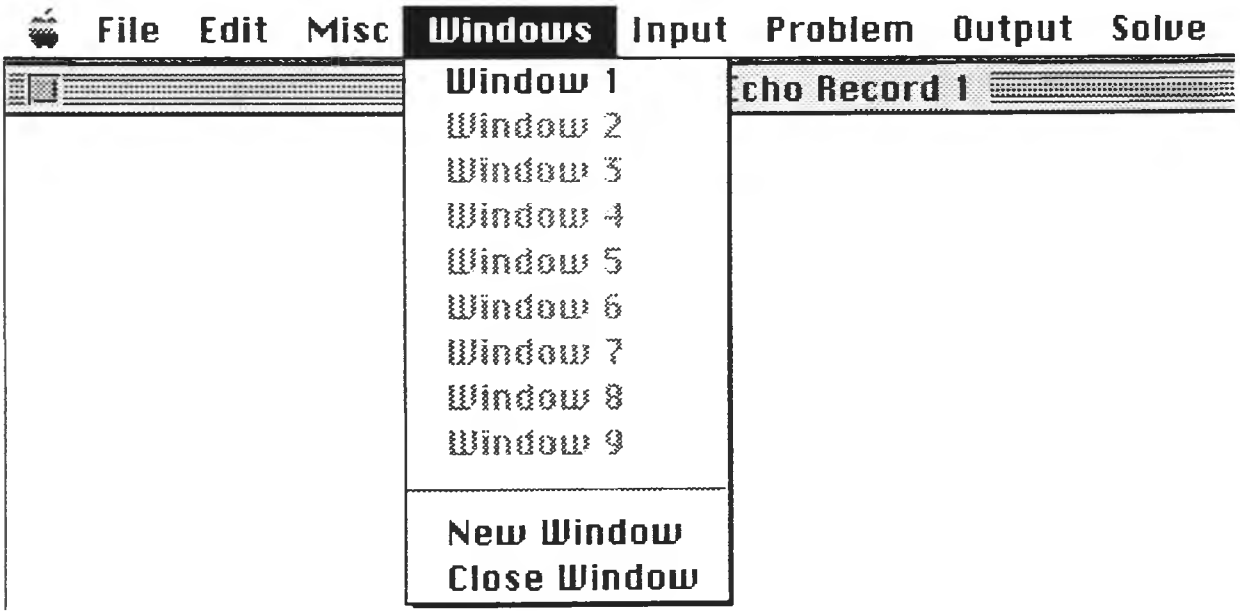
H.2- Edit Menu



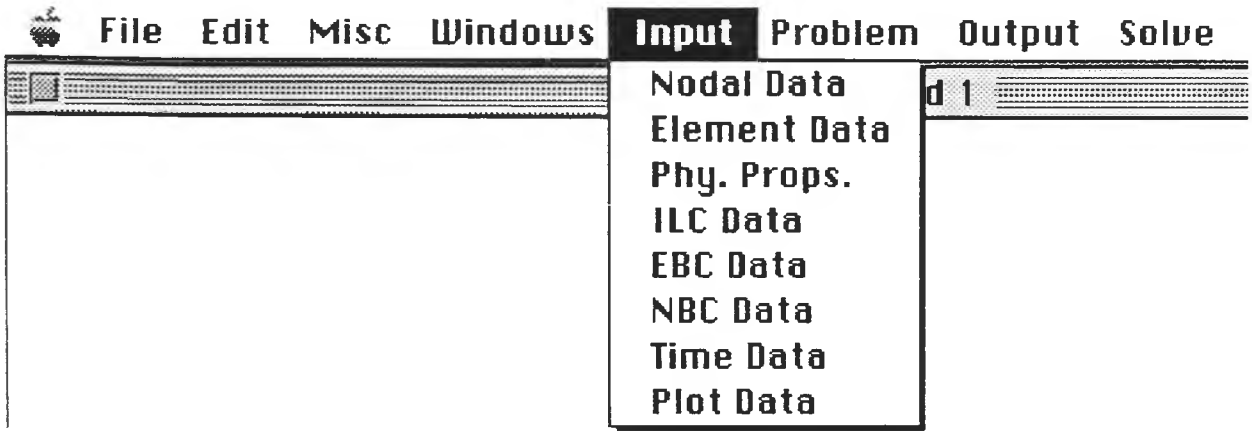
H.3 Misc Menu



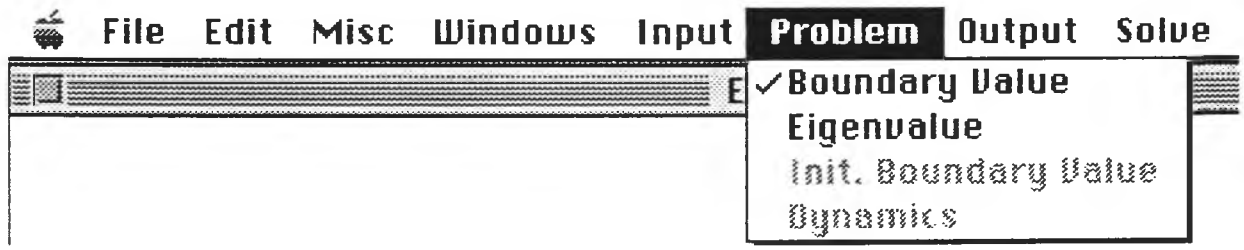
H.4 Windows Menu



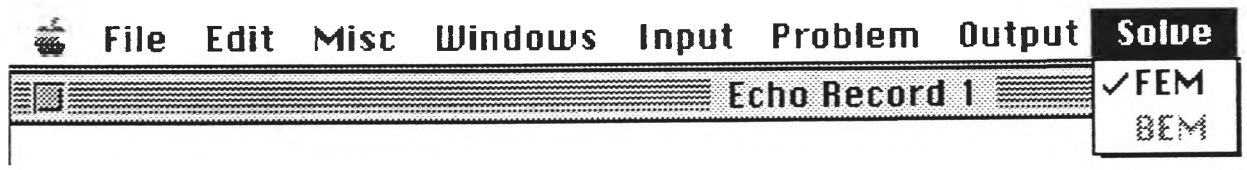
H.5 Input Menu



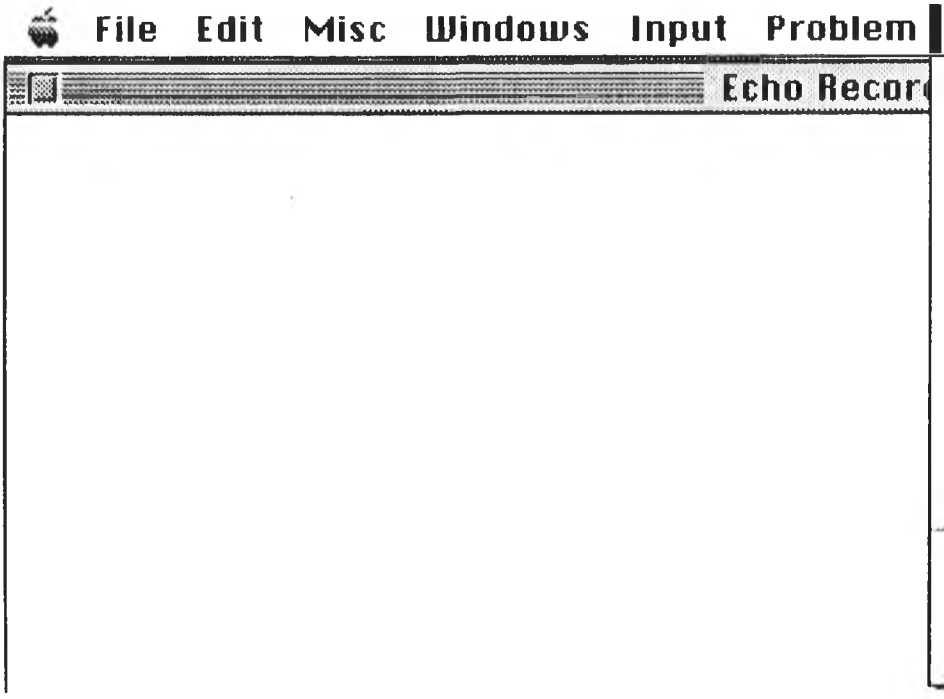
H.6 Problem Menu

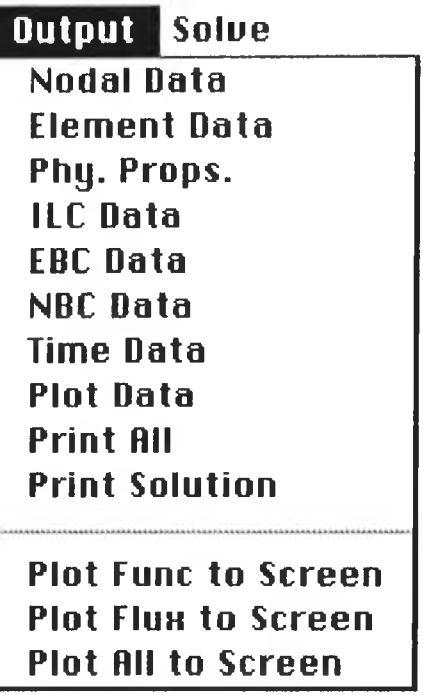


H.7 Solve Menu



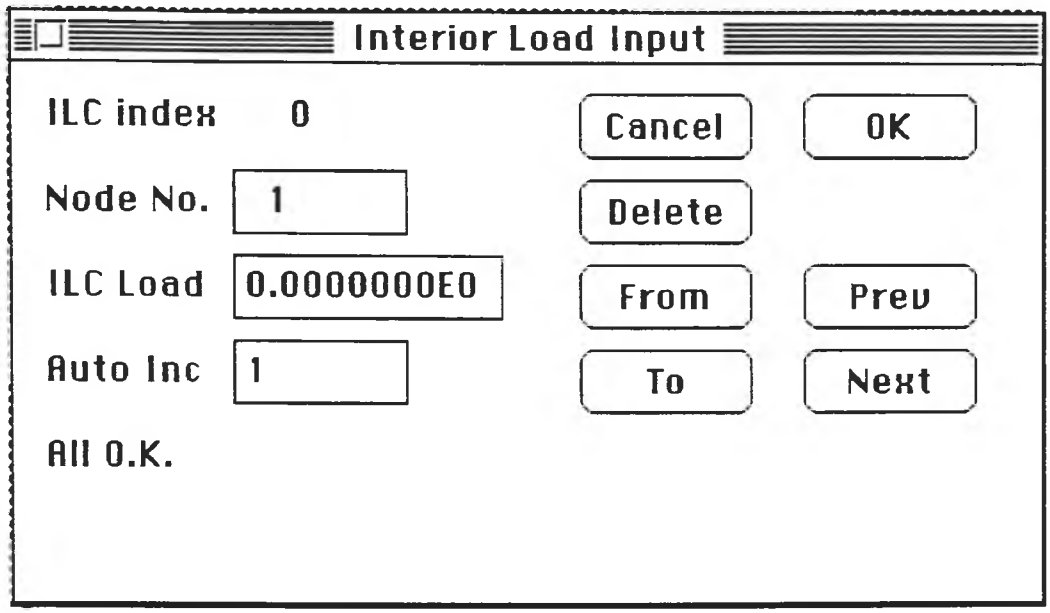
H.8 Output Menu



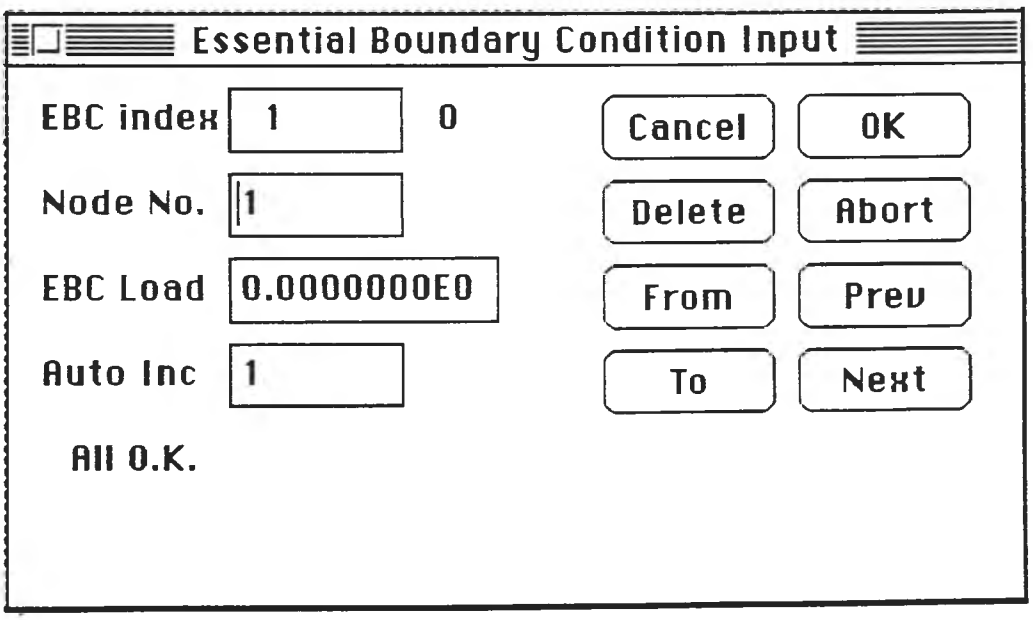

Output
Solve
Nodal Data
Element Data
Phy. Props.
ILC Data
EBC Data
NBC Data
Time Data
Plot Data
Print All
Print Solution
Plot Func to Screen
Plot Flux to Screen
Plot All to Screen

Appendix I - All Dialogs

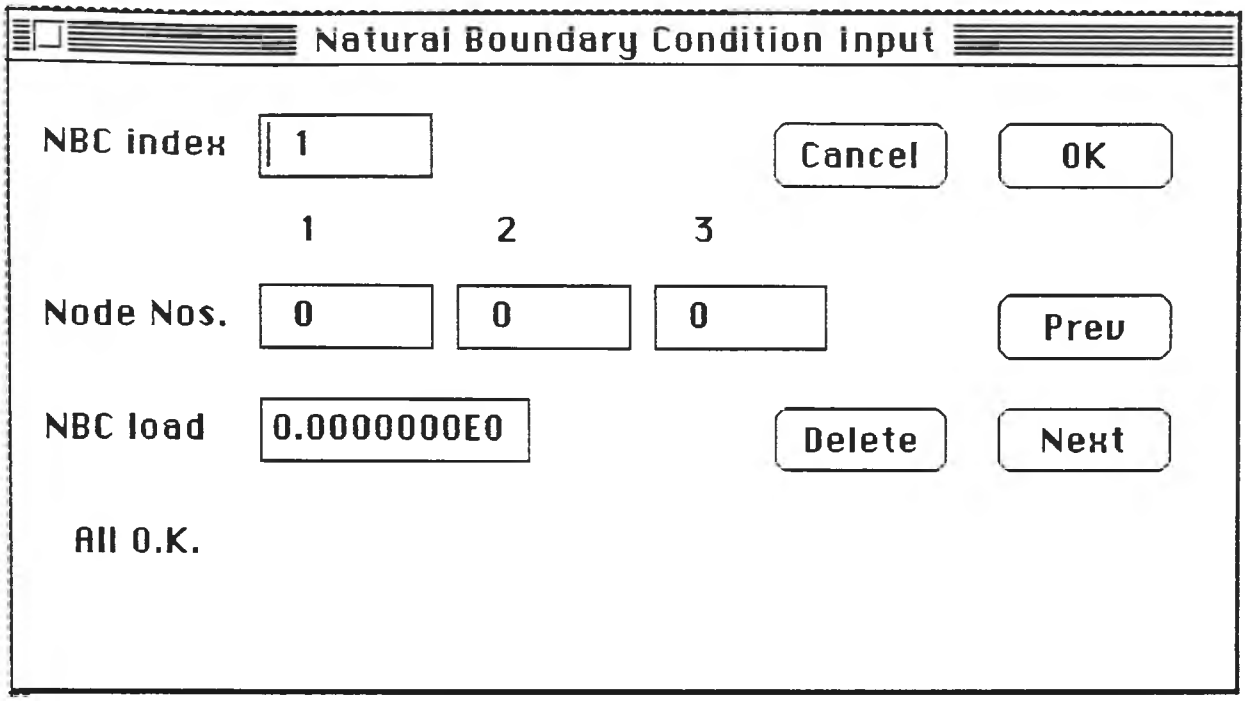
I.1 The interior load conditions data input dialog box.



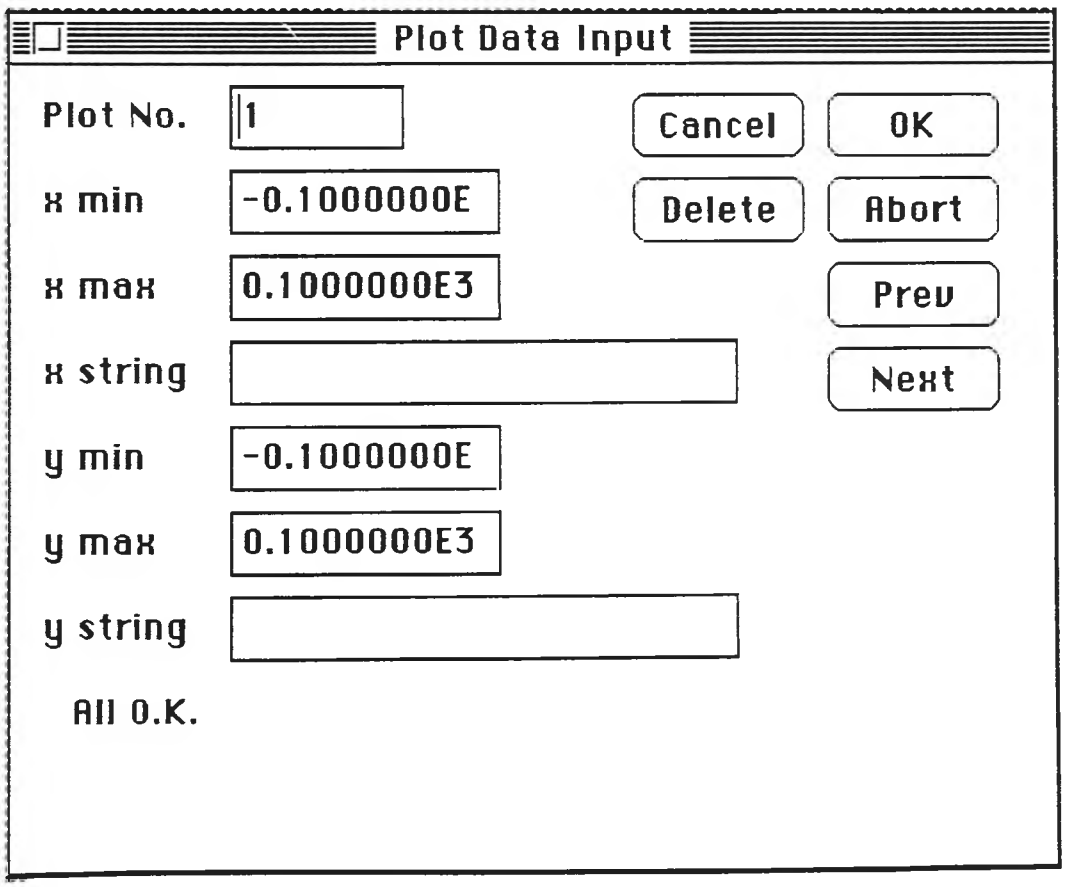
I.2 The essential boundary conditions data input dialog box.



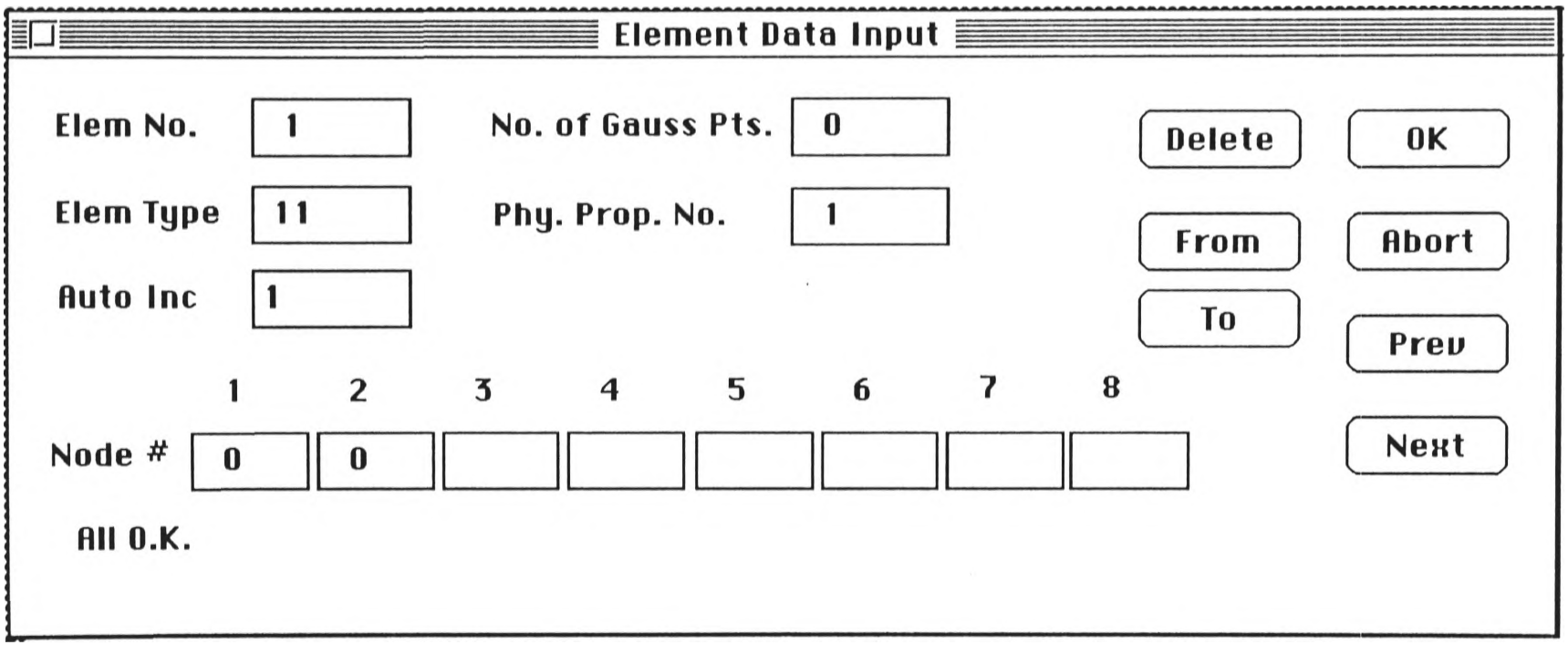
I.3 The natural boundary conditions data input dialog box.



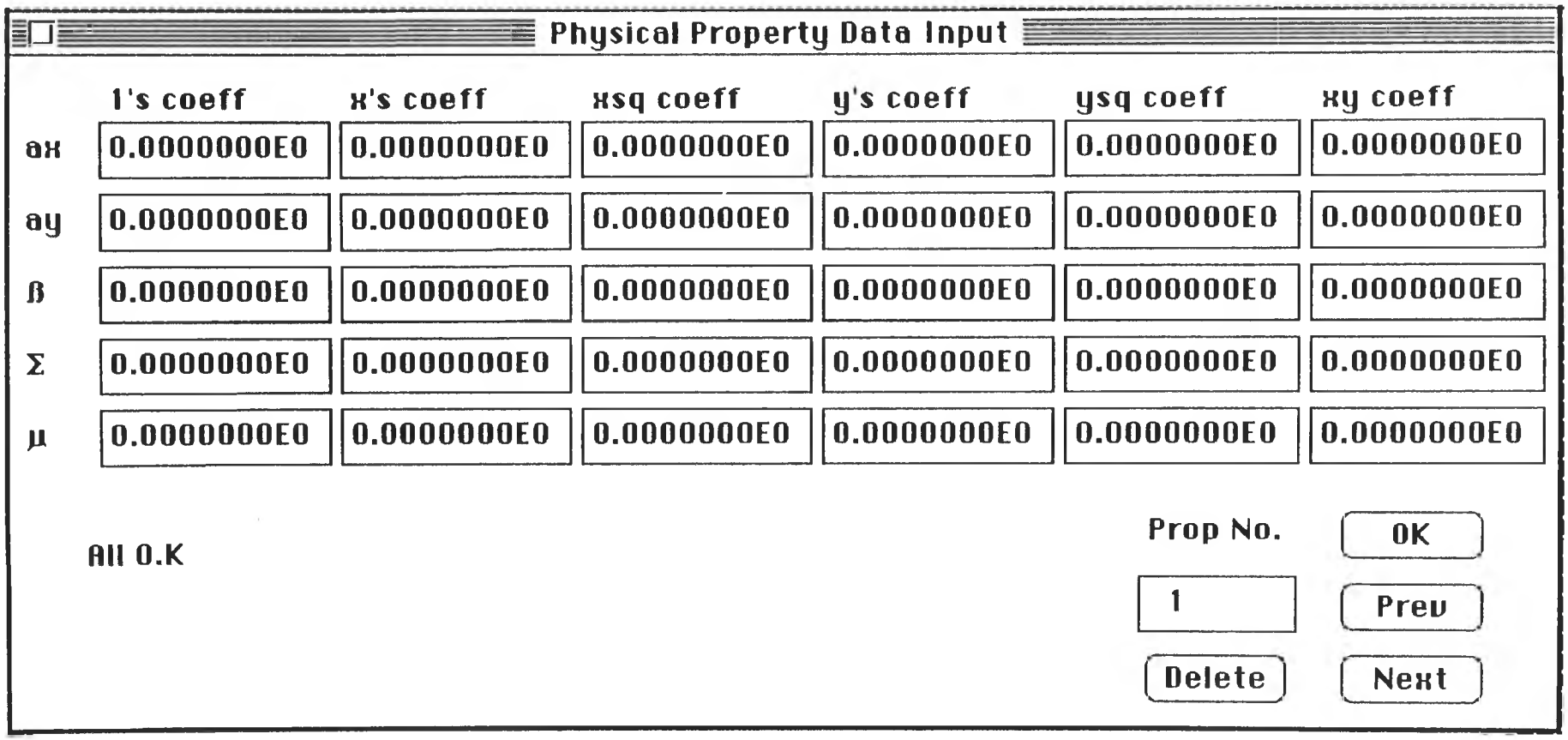
I.4 The plot data input dialog box.



I.5 The element data input dialog box.



I.6 The physical property data input dialog box.



Appendix J - FORTRAN Code

```
      SUBROUTINE GENJAC (A,B,X,EIV,D,N,RTOL,NSMAX,IFPR,IOUT)
C
C     REFERENCE: BATHE AND WILSON, 'NUMERICAL METHODS IN FINITE
C                ELEMENT ANALYSIS',  PP 458-469.
C                TITLE OF SUBROUTINE - JACOBI
C
C-----------------------------------------------------------------
C
C     P R O G R A M
C         TO SOLVE THE GENERALIZED EIGENPROBLEM USING THE
C         GENERALIZED JACOBI ITERATION
C
C     - - INPUT VARIABLES - -
C         A(N,N) = STIFFNESS MATRIX (ASSUMED POSITIVE DEFINITE)
C         B(N,N) = MASS MATRIX (ASSUMED POSITIVE DEFINITE)
C         X(N,N) = MATRIX STORING EIGENVECTORS ON SOLUTION EXIT
C         EIGV(N) = VECTORS STORING EIGENVALUES ON SOLUTION EXIT
C         D (N) = WORKING VECTOR
C         N = ORDER OF MATRICES A AND B
```

```
C          RTOL = CONVERGENCE TOLERANCE(USUALLY SET TO 10**-12)
C          NSMAX = MAXIMUM NUMBER OF SWEEPS ALLOWED
C                                (USUALLY SET TO 15)
C          IFPR = FLAG FOR PRINT DURING ITERATION
C               EQ.0 NO PRINTING
C               EQ.1 INTERMEDIATE RESULTS ARE PRINTED
C          IOUT = OUTPUT DEVICE NUMBER
C
C      - - OUTPUT - -
C          A(N,N) = DIAGONALIZED STIFFNESS MATRIX
C          B(N,N) = DIAGONALIZED MASS MATRIX
C          X(N,N) = EIGENVECTORS STORED COLUMNWISE
C          EIGV(N) = EIGENVALUES
C
C-----------------------------------------------------------------
C      IMPLICIT REAL*8(A-H.O-Z)
C      ABS(X) = DABS(X)
C      SQRT(X) = DSQRT(X)
C-----------------------------------------------------------------
C      THIS PROGRAM IS USED IN SINGLE PRECISION ARITHMETIC ON
C      CDC EQUIPMENT AND DOUBLE PRECISION ARITHMETIC ON IBM
C      OR UNIVAC MACHINES.  ACTIVATE, DEACTIVATE, OR ADJUST ABOVE
C      CARDS FOR SINGLE OR DOUBLE PRECISION ARITHMETIC
```

```
C-------------------------------------------------------------
      DIMENSION A(N,N),B(N,N),X(N,N),EIGV(N),D(N)
C
C     INITIALIZE EIGENVALUE AND EIGENVECTOR MATRICES
C
      DO 10 I=1,N
      IF (B(I,I).GT.O.) GO TO 4
      II = 1
      WRITE (IOUT,2020) II
      STOP
    4 D(I) = A(I,I)/B(I,I)
   10 EIGV(I) = D(I)
      DO 30 I = 1,N
      DO 20 J = 1,N
   20 X(I,J) = 0.
   30 X(I,I) = 1.0
      IF (N.EQ.1) RETURN
C
C     INITIALIZE SWEEP COUNTER AND BEGIN ITERATION
C
      NSWEEP = 0
      NR = N-1
   40 NSWEEP = NSWEEP +1
```

```
      IF (IFPR.EQ.1) WRITE (IOUT,2000) NSWEEP
C
C     CHECK IF PRESENT OFF-DIAG EL LARGE ENOUGH TO REQ. ZEROING
C
      EPS = (.01**NSWEEP)**2
      DO 210 J = 1,NR
      JJ = J + 1
      DO 210 K = JJ,N
      EPTOLA = (A(J,K)*A(J,K))
      EPTOLB = (B(J,K)*B(J,K))
      IF ( (EPTOLA.LT.EPS*A(J,J)*A(K,K)) .AND.
     1     (EPTOLB.LT.EPS*B(J,J)*B(K,K)) ) GO TO 210
C
C     IF ZEROING IS REQ. CAL. ROTATION MATRIX ELEMENTS CA AND CG
C
      AKK = A(K,K)*B(J,K) - B(K,K)*A(J,K)
      AJJ = A(J,J)*B(J,K) - B(J,J)*A(J,K)
      AB  = A(J,J)*B(K,K) - A(K,K)*B(J,J)
      CHECK = (AB*AB + 4.*AKK*AJJ)/4.
      IF (CHECK) 50,60,60
   50 II = 2
      WRITE (IOUT,2020) II
      STOP
```

```
     60 SQCH = SQRT(CHECK)
        D1=AB/2.+SQCH
        D2=AB/2.-SQCH
        DEN = D1
        IF (ABS(D2).GT.ABS(D1)) DEN = D2
        IF (DEN)80,70,80
     70 CA = 0.
        CG = -B(J,K)/B(K,K)
        GO TO 90
     80 CA = AKK/DEN
        CG = -AJJ/DEN
C
C       PERFORM GEN. ROTATION TO ZERO PRESENT OFF-DIAGONAL ELEM
C
     90 IF (N-2)100,190,100
    100 JP1 = J + 1
        JM1 = J - 1
        KP1 = K + 1
        KM1 = K - 1
        IF (JM1 - 1) 130,110,100
    110 DO 120 I=1,JM1
        AJ = A(I,J)
        BJ = B(I,J)
```

```
112         AK = A(I,K)
113         BK = B(I,K)
114         A (I,J) = AJ + CG*AK
115         B (I,J) = BJ + CG*BK
116         A (I,K) = AK + CA*AJ
117     120 B (I,K) = BK + CA*BJ
118     130 IF (KP1-N) 140,140,160
119     140 DO 150 I = KP1,N
120         AJ = A(J,I)
121         BJ = B(J,I)
122         AK = A(K,I)
123         BK = B(K,I)
124         A(J,I) = AJ+CG*AK
125         B(J,I) = BJ+CG*BK
126         A(K,I) = AK+CA*AJ
127     150 B(K,I) = BK+CA*BJ
128     160 IF (JP1-KM1)170,170,190
129     170 DO 180 I = JP1,KM1
130         AJ = A(J,I)
131         BJ = B(J,I)
132         AK = A(I,K)
133         BK = B(I,K)
134         A(J,I) = AJ + CG*AK
```

```
135         B(J,I) = BJ + CG*BK
136         A(I,K) = AK + CA*AJ
137     180 B(I,K) = BK + CA*BJ
138     190 AK = A(K,K)
139         BK = B(K,K)
140         A(K,K) = AK + 2.*CA*A(J,K) + CA*CA*A(J,J)
141         B(K,K) = BK + 2.*CA*B(J,K) + CA*CA*B(J,J)
142         A(J,J) = A(J,J) + 2.*CG*A(J,K) + CG*CG*AK
143         B(J,J) = B(J,J) + 2.*CG*B(J,K) + CG*CG*BK
144         A(J,K) = 0.
145         B(J,K) = 0.
146   C
147   C     UPDATE THE EIGENVECTOR MATRIX AFTER EACH ROTATION
148   C
149         DO 200 I = 1,N
150         XJ = X(I,J)
151         XK = X(I,K)
152         X(I,J) = XJ + CG*XK
153     200 X(I,K) = XK + CA*XJ
154     210 CONTINUE
155   C
156   C     UPDATE THE EIGENVALUES AFTER EACH SWEEP
157   C
```

```
158         DO 220 I = 1,N
159         IF (B(I,I).GT.O.) GO TO 220
160         II = 3
161         WRITE (IOUT,2020) II
162         STOP
163       220 EIGV(I) = A(I,I)/B(I,I)
164         IF(IFPR.EQ.O) GO TO 230
165         WRITE (IOUT,2030)
166         WRITE (IOUT,2010) (EIGV(I),I=1,N)
167   C
168   C     CHECK FOR CONVERGENCE
169   C
170       230 DO 240 I = 1,N
171         TOL = RTOL*D(I)
172         DIF = ABS(EIGV(I)-D(I))
173         IF (DIF.GT.TOL) GO TO 280
174       240 CONTINUE
175   C
176   C     CHECK ALL OFF-DIAG ELS TO SEE IF ANOTHER SWEEP IS REQUIRED
177   C
178         EPS = RTOL**2
179         DO 250 J=1,NR
180         JJ = J + 1
```

```
      DO 250 K = JJ,N
      EPSA = (A(J,K)*A(J,K))
      EPSB = (B(J,K)*B(J,K))
      IF ( (EPSA.LT.EPS*A(J,J)*A(K,K)) .AND.
     1     (EPSB.LT.EPS*B(J,J)*B(K,K)) ) GO TO 250
      GO TO 280
  250 CONTINUE
C
C     FILL BOTTOM TRI OF RESULTANT MATRICES & SCALE EIGENVECTORS
C
  255 DO 260 I = 1,N
      DO 260 J = 1,N
      A(J,I) =A(I,J)
  260 B(J,I) =B(I,J)
      DO 270 J = 1,N
      BB=SQRT(B(J,J))
      DO 270 K = 1,N
  270 X(K,J) = X(K,J)/BB
      RETURN
C
C     UPDATE D MATRIX AND START NEW SWEEP, IF ALLOWED
C
  280 DO 290 I = 1,N
```

```
 290 D(I) = EIGV(I)
     IF (NSWEEP.LT.NSMAX) GO TO 40
     GO TO 255
2000 FORMAT (27HOSWEEP NUMBER IN 'JACOBI' = .I4)
2010 FORMAT (1HO.6E20.12)
2020 FORMAT (3HII=.I2/25HO*** ERROR  SOLUTION STOP /
    1        31H MATRICES NOT POSITIVE DEFINITE)
2030 FORMAT (36HOCURRENT EIGENVALUES IN 'JACOBI' ARE /)
     END
```

Appendix K - Modula-2 Code

```
(*
 *
 *    reference:  Bathe and Wilson,  'numerical methods in finite
 *                element analysis',  pp 458-460,
 *                title of subroutine - jacobi
 *
 *-------------------------------------------------------------
 *  p r o g r a m
 *    to solve the generalized eigenproblem using the
 *    generalized jacobi iteration
 *
 * - - input variables - -
 *    a [n,n]      :=    stiffness matrix [assumed positive definite]
 *    b [n,n]      :=    mass matrix [assumed positive definite]
 *    x [n,n]      :=    matrix storing eigenvectors on solution exit
 *    eigv[n]      :=    vectors storing eigenvalues on solution exit
 *    d [n]        :=    working vector
 *    n            :=    order of matrices a and b
 *    rtol         :=    convergence tolerance [usually set to 10e-12]
 *    nxmax        :=    maximum number of sweeps allowed
 *                       [usually set to 15]
 *    ifpr         :=    flag for print during iteration
 *                  eq = 0; no printing
 *                  eq = 1; intermediate results are printed
```

```
28  *       iout          :=      output device number
29  *
30  * - - output - -
31  *       a[n,n]        :=      diagonalized stiffness matrix
32  *       b [n,n]       :=      diagonalized mass matrix
33  *       x [n,n]       :=      eigenvectors stored columnwise
34  *       eigv [n]      :=      eigenvalues
35  *-------------------------------------------------------------
36  *       implicit real*8[a-h,o-z]
37  *       abs[x] := dabs[x]
38  *       sort[x] := dsqrt[x]
39  *-------------------------------------------------------------
40  *
41  *)
42
43
44  PROCEDURE GenJac (n: INTEGER; rtol : REAL; nsmax,ifpr,iout : INTEGER);
45  (* a[n,n],b[n,n],x[n,n],eigv[n], global access;  d[n], local in GenJac *)
46
47  VAR
48        i, j, nsweep, nr, jj, k, jp1, jm1, kp1, km1 : INTEGER;
49        eps, eptola, eptolb, akk, ajj, ab, check, sqch : REAL;
50        d1, d2, den, ca, cg, aj, bj, ak, bk, xj, xk :REAL;
51        tol, dif, epsa, epsb, bb : REAL;
52        dif_more_than_tol, condition : BOOLEAN;
53        d : ARRAY [1..61] OF REAL;
54        loop1,loop2 : INTEGER;
55  BEGIN
```

```
56  (*      initialize eigenvalue and eigenvector matrices*)
57          FOR i := 1 TO n DO
58                  IF (meig[i,i]  <= 0.0) THEN
59                          Handle_Calc_Errors_Dialog (82,i);
60                                  (* matrices not +ve definite *)
61                          RETURN;
62                  ELSE
63                          d[i] :=  keig[i,i]/meig[i,i];
64                          eigv[i] := d[i];
65                  END(*IF*);
66          END(*FOR*);
67
68          FOR i := 1 TO n DO
69                  FOR j := 1 TO n DO
70                          xvec[i,j] := 0.0;
71                  END(*FOR*);
72                  xvec[i,i] := 1.0;
73          END(*FOR*);
74
75          IF n = 1 THEN
76                  RETURN;
77          END(*IF*);
78
79  (*      initialize sweep counter and begin iteration *)
80          nsweep := 0;
81          nr := n-1;
82
83          REPEAT
```

```
84              nsweep := nsweep +1;
85
86              IF (ifpr = 1) THEN
87                   (*show message write [iout,2000] nsweep;
88                                 format [27hosweep number in 'jacobi' := ,i4] *)
89              END(*IF*);
90
91   (*    check if present off-diagonal element is large enough to require zeroing *)
92              eps := Power ( (Power(0.01, FLOAT(nsweep) )), 2.0);
93
94              FOR j := 1 TO nr DO
95                   jj := j + 1;
96
97                   FOR k := jj TO n DO
98                        eptola := keig[j,k]*keig[j,k];
99                        eptolb := meig[j,k]*meig[j,k];
100
101                       IF ( (eptola < (eps*keig[j,j]*keig[k,k])) AND (eptolb < (eps*meig[j,j]*meig[k,k])) ) THEN
102                            (*do nothing! exit IF*)
103                       ELSE
104
105  (*    if zeroing is required calculate the rotation matrix elements ca and cg *)
106                            akk :=keig[k,k]*meig[j,k]-meig[k,k]*keig[j,k];
107                            ajj := keig[j,j]*meig[j,k] - meig[j,j]*keig[j,k];
108                            ab  := keig[j,j]*meig[k,k] - keig[k,k]*meig[j,j];
109                            check := (ab*ab + 4.0*akk*ajj)/4.0;
110
111                            IF check < 0.0 THEN
```

```
112                                     Handle_Calc_Errors_Dialog (83,
113                                         nsweep*4096 + j*64 + k);
114                                             (* matrices not +ve definite *)
115                                     RETURN;
116                                 END(*IF*);
117
118                                 sqch := Sqrt(check);
119                                 d1:=ab/2.0+sqch;
120                                 d2:=ab/2.0-sqch;
121                                 IF ABS(d2) > ABS(d1) THEN
122                                     den := d2;
123                                 ELSE
124                                     den := d1;
125                                 END(*IF*);
126
127                                 IF den = 0.0 THEN
128                                     ca := 0.0;
129                                     cg := -meig[j,k]/meig[k,k];
130                                 ELSE
131                                     ca := akk/den;
132                                     cg := -ajj/den;
133                                 END(*IF*);
134
135 (*      perform the generalised rotation to zero the present off-diagonal element *)
136                                 IF (n-2) <> 0 THEN
137                                     jp1 := j + 1;
138                                     jm1 := j - 1;
139                                     kp1 := k + 1;
```

140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167

```
km1 := k - 1;

IF (jm1-1) >= 0 THEN
        FOR i := 1 TO jm1 DO
                aj := keig[i,j];
                bj := meig[i,j];
                ak := keig[i,k];
                bk := meig[i,k];
                keig [i,j] := aj + cg*ak;
                meig [i,j] := bj + cg*bk;
                keig [i,k] := ak + ca*aj;
                meig [i,k] := bk + ca*bj;
        END(*FOR*);
END(*IF*);

IF (kp1-n) <= 0 THEN
        FOR i := kp1 TO n DO
                aj := keig[j,i];
                bj := meig[j,i];
                ak := keig[k,i];
                bk := meig[k,i];
                keig[j,i] := aj+cg*ak;
                meig[j,i] := bj+cg*bk;
                keig[k,i] := ak+ca*aj;
                meig[k,i] := bk+ca*bj;
        END(*FOR*);
END(*IF*);
```

```
168                         IF (jp1-km1) <= 0 THEN
169                             FOR i := jp1 TO km1 DO
170                                 aj := keig[j,i];
171                                 bj := meig[j,i];
172                                 ak := keig[i,k];
173                                 bk := meig[i,k];
174                                 keig[j,i] := aj + cg*ak;
175                                 meig[j,i] := bj + cg*bk;
176                                 keig[i,k] := ak + ca*aj;
177                                 meig[i,k] := bk + ca*bj;
178                             END(*FOR*);
179                         END(*IF*);
180                     END(*IF*);
181
182                     ak := keig[k,k];
183                     bk := meig[k,k];
184                     keig[k,k] := ak + 2.0*ca*keig[j,k] +
185                                       ca*ca*keig[j,j];
186                     meig[k,k] := bk + 2.0*ca*meig[j,k] +
187                                       ca*ca*meig[j,j];
188                     keig[j,j] := keig[j,j] + 2.0*cg*keig[j,k] +
189                                              cg*cg*ak;
190                     meig[j,j] := meig[j,j] + 2.0*cg*meig[j,k] +
191                                              cg*cg*bk;
192                     keig[j,k] := 0.0;
193                     meig[j,k] := 0.0;
194
195 (*      update the eigenvector matrix after each rotation *)
```

```
196                                 FOR i := 1 TO n DO
197                                     xj := xvec[i,j];
198                                     xk := xvec[i,k];
199                                     xvec[i,j] := xj + cg*xk;
200                                     xvec[i,k] := xk + ca*xj;
201                                 END(*FOR*);
202                             END(*ELSE eptola*);
203                         END(*FOR k*);
204                     END(*FOR j*);
205
206     (*      update the eigenvalues after each sweep *)
207                     FOR i := 1 TO n DO
208                         IF (meig[i,i] > 0.0) THEN
209                             eigv[i] := keig[i,i]/meig[i,i];
210                         ELSE
211                             Handle_Calc_Errors_Dialog (84, nsweep*4096 + i);
212                                 (* matrices not +ve definite *)
213                             RETURN;
214                         END(*IF*);
215                     END(*FOR*);
216
217                     IF ifpr <> 0 THEN
218                         (* write current eigenvalues in 'jacobi' are
219                                 format [1ho,6e20, [eigv[i],i:=1,n] *)
220                     END(*IF*);
221
222
223     (*      check for convergence *)
```

```
224             i := 1;
225             dif_more_than_tol := FALSE;
226             WHILE ( (i<=n) AND (dif_more_than_tol = FALSE) ) DO
227                     tol := rtol*d[i];
228                     dif := ABS(eigv[i]-d[i]);
229                     IF dif > tol THEN
230                             dif_more_than_tol := TRUE;
231                     END(*IF*);
232                     i := i+1;
233             END(*WHILE*);
234
235     (*      check all off-diagonal elements to see if another sweep is required *)
236             IF NOT dif_more_than_tol THEN
237                     condition := FALSE;
238                     eps := rtol*rtol;
239
240                     j := 1;
241                     WHILE ( (j<=nr) AND (condition = FALSE) ) DO
242                             jj := j + 1;
243                             k := jj;
244                             WHILE ( (k<=n) AND (condition = FALSE) ) DO
245                                     epsa := keig[j,k]*keig[j,k];
246                                     epsb := meig[j,k]*meig[j,k];
247                                     IF ( (epsa < eps*keig[j,j]*keig[k,k]) AND
248                                             (epsb < eps*meig[j,j]*meig[k,k]) ) THEN
249                                             (*do nothing! exit IF*)
250                                     ELSE
251                                             condition := TRUE;
```

```
252                             END(*IF*);
253                             k := k+1;
254                         END(*WHILE*);
255                         j := j+1;
256                     END(*WHILE*);
257
258 (*      fill out bottom triangle of resultant matrices and scale eigenvectors *)
259                     IF NOT condition THEN
260                         FOR i := 1 TO n DO
261                             FOR j := 1 TO n DO
262                                 keig[j,i] := keig[i,j];
263                                 meig[j,i] := meig[i,j];
264                             END(*FOR*);
265                         END(*FOR*);
266
267                         FOR j := 1 TO n DO
268                             bb := Sqrt(meig[j,j]);
269                             FOR k := 1 TO n DO
270                                 xvec[k,j] := xvec[k,j]/bb;
271                             END(*FOR*);
272                         END(*FOR*);
273                         RETURN;
274                     END(*IF*);
275                 END(*IF*);
276
277 (*      update d matrix and start new sweep, if allowed *)
278                 FOR i := 1 TO n DO
279                     d[i] := eigv[i];
```

```
280             END(*FOR*);
281         UNTIL (nsweep >= nsmax);
282
283         FOR i := 1 TO n DO
284             FOR j := 1 TO n DO
285                 keig[j,i] := keig[i,j];
286                 meig[j,i] := meig[i,j];
287             END(*FOR*);
288         END(*FOR*);
289
290         FOR j := 1 TO n DO
291             bb := Sqrt(meig[j,j]);
292             FOR k := 1 TO n DO
293                 xvec[k,j] := xvec[k,j]/bb;
294             END(*FOR*);
295         END(*FOR*);
296
297     END GenJac;
```

Appendix L - Error Messages

This Appendix lists some of the error messages indicated by FAME. They are not exhaustive, but are only representative of the kinds of errors reported. Where applicable, similar error messages will also be present for the other data types. Most of the following examples apply to *Nodes*.

Error 0: "All O.K."

Error 1: "Number of nodal points exceeds ", maxnp, " the maximum allowable."

Error 7: "Node number ", value, " is not in permissible range of node numbers. Defaulting to 1."

Error 8: "Node number ", value, " is out of order."

Error 18: "maximum bandwith as calculated ", value, " exceeds maximum permissible bandwidth ", maxbnd.

Error 19: "zero or negative determinant of the jacobian matrix is found for element ", value.

Error 21: "for an eigen problem, the number of nodes exceeds ", maxbnd, " the maximum allowable for eigen problems.".

Error 22: "The node ", value, " seems to be missing!".

Error 23: "The node ", value, " was not used in any element.".

Error 24: "Duplicate essential boundary loads input.".

Error 26: "The number of nodes = ", value. "There must be two or more nodes for each problem.".

(* error number 50 onwards have been added by Robert Pierre D'souza 6/2/93 *)

Error 50: "Index = ", value, " is out of range. Default value is 1.".

Error 51: "Index = ", value, " has reached the maximum limit.".

Error 52: "Index = ", value, " has reached the minimum limit.".

Error 54: "You have deleted internal node ", value. "The nodal data is no longer contiguous.".

Error 59: "Not a 2-D problem! So 'alphay' not required.".

Error 67: "An Eigen problem can not have ILCs.".

Error 70: "A 1-D problem can not have y co-ordinates.".

Error 72: "Eigen Solution Analysis not done. Aborting Print!".

Error 76: "FROM node =", value, " and TO node =", value. "FROM and TO nodes have to be different.".

Error 82: "Error in eigen solution. The matrices are not positive definite. meig[i,i] is negative."

Appendix M - Shape Functions

Shape functions for a 1-D Quadratic Isoparametric bar element. It can be used to solve 1-D FEM problems or 2-D BEM problems.

$$\phi_1(\xi) = \frac{1}{2}\xi(\xi - 1) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{M.1}$$

$$\phi_2(\xi) = (1 + \xi)(1 - \xi) \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{M.2}$$

$$\phi_3(\xi) = \frac{1}{2}\xi(\xi + 1) \quad .. \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{M.3}$$

Shape functions for a 2-D Quadratic Isoparametric quadrilateral element. It can be used to solve 1-D FEM problems or 2-D BEM problems.

$$\phi_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta)(-\xi - \eta - 1) \quad .. \quad .. \quad \text{M.4}$$

$$\phi_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta)(\xi - \eta - 1) \quad .. \quad .. \quad \text{M.5}$$

$$\phi_3(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 + \eta)(\xi + \eta - 1) \quad .. \quad .. \quad \text{M.6}$$

$$\phi_4(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 + \eta)(-\xi + \eta - 1) \quad .. \quad .. \quad \text{M.7}$$

$$\phi_5(\xi, \eta) = \frac{1}{2}(1 - \xi^2)(1 - \eta) \quad .. \quad .. \quad .. \quad .. \quad \text{M.8}$$

$$\phi_6(\xi, \eta) = \frac{1}{2}(1 + \xi)(1 - \eta^2) \quad .. \quad .. \quad .. \quad .. \quad \text{M.9}$$

$$\phi_7(\xi, \eta) = \frac{1}{2}(1 - \xi^2)(1 + \eta) \quad .. \quad .. \quad .. \quad .. \quad \text{M.10}$$

$$\phi_8(\xi, \eta) = \frac{1}{2}(1 - \xi)(1 - \eta^2) \quad .. \quad .. \quad .. \quad .. \quad \text{M.11}$$

Appendix N - Modula-2 P1 Reply

Date: 21 Jun 93 17:39 GMT

From: GER.XSE0109@applelink.apple.com (Germany - P1 GmbH, H Henne,IDV)

Subject: Re-no subject

To: PIERRE@CS.UOW.EDU.AU

Message-Id: <740684721.5759264@AppleLink.Apple.COM>

Status: RO

Hello,

sorry for my late response, but I just returned from a two week visit to Canada.

What you have noticed is a known problem for the current release of our compiler. After V4.2 we changed the error handling to using the same entry point as MacsBug, but the MacsBug error entry is not called for FPU errors on 68040 CPUs. We have just a few days ago detected how to circumvent this and plan an according change for the next releases (V5.1/V4.4.2). The latest release V5.0.1/V4.4.1 does at least report divison by zero, but it does not yet handle it properly.

The characters I and N stand for 'Infinity' and 'NAN' (not a number), these are generated by the FPU if no errors are reported.

Elmar Henne

p1 GmbH

Appendix O - Porting Problems

This Appendix contains some of the debugging problems faced when porting the F77 program UNAFEM to a M2 version FAME.

O.1 No Equivalence Problems

F77 supports multi-dimensional arrays, which can be equivalently represented as a linear array having the same number of elements. This concept of equivalence is used to :-

a) access the same data by using different data formats,

b) access the same data by using different names,

c) conserve storage space by making different variables occupy the same memory locations at different times during program execution.

In UNAFEM, b) has been implemented. This permits programmers to individually develop modules using names that are most convenient to them. On completion of the individual debugging, both modules are integrated by declaring the corresponding variables to be EQUIVALENT. Obviously, in this case, the number of dimensions and the size of each dimension of the array are the same.

A potential porting problem, that did not arise, is case a). eg. a two dimensional array has been declared equivalent to a linear array. In such a case, all manipulations to the equivalent linear array would have to be re-coded to generate the desired output results.

M2's variant records provide the same features as a), b) and c). Besides, ADR and ADDRESS can change the type of anything, by bypassing the type-checking mechanism of M2.

O.2 Typographical Errors

To retain continuity between the F77 and M2 software, it was decided to use the variables without modifying their names. Subsequently, debugging of the software and testing of sample programs (with known solutions; [Burnett88]) would impart on-line information about software intricacies and the inter-relationships between the software and the matrix equations it implemented. This increased understanding would enable more comprehensible names to be declared at a later stage, hence improving readability and maintainability of the package.

F77 supports only 6 character variable and function names, resulting in rather cryptic names and hence many typographical errors, most of which were caught by M2 at compile time (as undefined variables!)

The more insidious typographical errors were those that correctly spelt another variable. If these slipped through the human checker (me), they only declared themselves at run time when they generated erroneous results. This required generation of a global cross-reference file to trace the interactions of the variables across modules, and also necessitated studying the internal workings of the functions.

O.3 Gaussian Elimination (GE)

The first series of tests, modelled with 44 standard 1-D linear elements, passed with no problems, the results tallying with those in the text [Burnett88]. The next series of tests, based on 10 1-D isoparametric quadratic elements, gave erroneous results. A check of the element's implementation revealed that, through oversight, some constant DATA had not been initialised as they were mistaken for variables. This is a grave flaw in M2, which did not even flag a warning to indicate the anomaly (unlike some implementations of the C language, which warn if variables are used before initialisation).

However, no other bugs could be detected, which only left the GE algorithm, even though it had apparently been debugged by use on the linear elements. Reference to Zienkiewicz[83] and Wilkinson[63] eliminated the possibility of rounding errors or of an ill-conditioned problem. Manual iterations through the GE function with a hand held calculator and reference to a text on numerical analysis [Chapra88] discovered the modification used to speed up GE for solution of a banded matrix of n equations, and, finally, elucidated the result that, during translation, a single line of code in the forward elimination section of the algorithm had slipped into a FOR loop, when it really belonged outside.

Further development of the 1-D isoparametric cubic and quintic elements was without incident.

O.4 Generalised Jacobi Algorithm Eigensolver (GJAE)

O.4.1 Open Array Parameters

Both F77 and M2 (Appendix A) support open-array parameters, though the default index of the first element of the open array parameter is different. It is the first element for F77, while it is the zeroth element for M2. Limited knowledge regarding the detailed workings of the GJAE precluded the modification of the indices of all the arrays in the code (in case of any implementation dependent code). An alternative approach would have been to keep the indices of all the matrices unchanged and to re-specify all the arrays as beginning from 0 rather than 1, and leave the zeroth element of each unused. This latter option has not been implemented either.

So, it was decided to access all the open array parameters globally, requiring the function call to be changed from (Appendix J, line 1)

```
SUBROUTINE GENJAC (A,B,X,EIV,D,N,RTOL,NSMAX,IFPR,IOUT)
```

to (Appendix K, line 44)

PROCEDURE GenJac (n: INTEGER; rtol : REAL; nsmax,ifpr,iout : INTEGER).

Additionally, all references to formal parameters within the function had to be changed to global references (Appendices J and K).

O.4.2 Common Terminal Statements

GJAE contained several nested loops with a common terminal statement.

(eg. Appendix J, lines 195-198)

```
      DO 270 J = 1,N
      BB=SQRT(B(J,J))
      DO 270 K = 1,N
270 X(K,J) = X(K,J)/BB
```

which was equivalent to [Meissner80]

```
      DO 270a J = 1,N
      BB=SQRT(B(J,J))
      DO 270b K = 1,N
270 X(K,J) = X(K,J)/BB
270b CONTINUE
270a CONTINUE
```

and was translated into M2 (Appendix K, lines 290-295) as

```
FOR j := 1 TO n DO
    bb := Sqrt(meig[j,j]);
    FOR k := 1 TO n DO
        xvec[k,j] := xvec[k,j]/bb;
    END(*FOR*);
END(*FOR*);
```

O.4.3 A REPEAT-UNTIL Loop

The F77 code portion (Appendix J, lines 63-205)

```
      NSWEEP = 0
      NR = N-1
 40 NSWEEP = NSWEEP +1
            ...
            ...
      IF (NSWEEP.LT.NSMAX) GO TO 40
```

and converted into a M2 REPEAT-UNTIL loop (Appendix K, lines 80-281) as

```
      nsweep := 0;
      nr := n-1;
      REPEAT
            nsweep := nsweep +1;
            ...
            ...
      UNTIL (nsweep >= nsmax);
```

O.4.4 Arithmetic IF's

The arithmetic IF's required special attention. Initially, due to a typographical error on line 108, lines 104-118,

```
 100  JP1 = J + 1
      JM1 = J - 1
      KP1 = K + 1
      KM1 = K - 1
      IF (JM1 - 1) 130,110,100
```

```
110 DO 120 I=1,JM1
                ...
                ...
120   ...
130 IF (KP1-N) 140,140,160
```

were translated as

```
        REPEAT
                jp1 := j + 1;
                jm1 := j - 1;
                kp1 := k + 1;
                km1 := k - 1;
        UNTIL ( (jm1-1) <= 0 );
        IF (jm1-1) = 0 THEN
                FOR i := 1 TO jm1 DO
                        ...
                        ...
                END(*FOR*);
        ELSE
                Code from label 130 onwards was inserted here!
        END(*IF*);
```

During debugging, this module caused the program to loop forever. Resort to Gourlay [73], Wilkinson [65] and Bathe [82] revealed the theory, and the fact that line 108 (Appendix J) should have been:-

```
        IF (JM1 - 1) 130,110,110
```

Correspondingly, the M2 translation simplified to lines 137-153 (Appendix K)

```
        jp1 := j + 1;
```

```
jm1 := j - 1;
kp1 := k + 1;
km1 := k - 1;
IF (jm1-1) >= 0 THEN
        FOR i := 1 TO jm1 DO
                ...
                ...
        END(*FOR*);
END(*IF*);
```

Knowledge of the algorithm and a questioning approach to translation would have detected this during the translation cycle, because it is obvious (in retrospect) that if the test at the end of the REPEAT-UNTIL loop fails once, then the program will loop forever, due to the invariant values of the variables within the loop!

O.4.5 A WHILE Loop with multiple conditions

Lines 170-174 (Appendix J) are handled specially. We note that line 173, if the condition is true, indicates a jump out of the loop over lines 175-202 to line 203, by using a GOTO. An alternate terminating condition for the loop is specified by the loop count, which, on reaching the maximum, permits lines 175-202 to be executed.

```
230 DO 240 I = 1,N
        TOL = RTOL*D(I)
        DIF = ABS(EIGV(I)-D(I))
        IF (DIF.GT.TOL) GO TO 280
240 CONTINUE
        ...
        lines 175-202
        ...
```

```
280 DO 290 I = 1,N
```

The M2 translation became lines 224-233 (Appendix K)

```
i := 1;
dif_more_than_tol := FALSE;
WHILE ( (i<=n) AND (dif_more_than_tol = FALSE) ) DO
        tol := rtol*d[i];
        dif := ABS(eigv[i]-d[i]);
        IF dif > tol THEN
                dif_more_than_tol := TRUE;
        END(*IF*);
        i := i+1;
END(*WHILE*);
```

We note that the IF condition, on passing, sets a flag. The WHILE loop tests both conditions, the loop count and the flag, either of which can cause the loop to terminate. The flag, if FALSE, conditionally permits the ensuing code to execute; else, it is jumped over.

Similar concepts were used to translate lines 175-202 (Appendix J) to lines 236-276 (Appendix K).

O.5 Compiler Problems

To speed up work, I tried to use *Modula-2 V4.3.2 © 1991 p1 Gesellschaft für Informatik mbH* on a Quadra 950.

Unfortunately, it did not flag a divide-by-zero error at the point of occurrence. Instead, it carried on with all its calculations, and declared final values to be 'infinite's or 'NAN's, making pinpointing of numerical problems difficult.

A letter to the company elicited the response (Appendix N) that future software releases would fix the problem.

Also, complex number support was promised! Both have been delivered in Version 5.0.1.

O.6 Error Messages

All variables and error messages in UNAFEM were rather cryptic. So their detailed interactions were difficult to comprehend, leading to the design decision that a first stage approach would be to port F77 to M2, retaining UNAFEM's variable names.

This was got around by 'discovery' during the run-time and debugging stages, which enriched understanding and led to the incorporation of twice the number of error messages, some of which are reproduced in Appendix L.

Appendix P - FAME Results

This Appendix shows that, to improve the solution accuracy, it is necessary to either increase the number of elements or the order of the shape function, or both. For the same number of DOF, higher order elements show better results.

These facts are already known and the testing indicates that the program works properly as it is able to reproduce/corroborate standard results.

P.1 Graphics Display

A simple graphics screen layout was developed to depict 1-D solutions pictorially (Figures P.1 to P.3). It assumes a rectangular window with (x_{min}, y_{min}) and (x_{max}, y_{max}) being the co-ordinates of the top-left and bottom-right corners respectively. From the Plot Data input menu, the user inputs $(x_{min(req)}, y_{min(req)})$ and $(x_{max(req)}, y_{max(req)})$ to define the limits of the region within which he would like to examine results.

This permits the program to calculate scaling factors along both the axes

$$scale\_factor_x = \frac{(x_{max} - x_{min})}{(x_{max(req)} - x_{min(req)})} \quad .. \quad .. \quad .. \quad \text{P.1}$$

$$scale\_factor_y = \frac{(y_{max} - y_{min})}{(y_{max(req)} - y_{min(req)})} \quad .. \quad .. \quad .. \quad \text{P.2}$$

Hence, to plot a point (x_{val}, y_{val}), the program scales it by

$$x_{plot} = (x_{val} - x_{min(req)}) * scale\_factor_x \quad .. \quad .. \quad \text{P.3}$$

$$y_{plot} = y_{max} - (y_{val} - y_{min(req)}) * scale\_factor_y \quad .. \quad \text{P.4}$$

P.2 BDVP Displays and Observations

The following 3 graphs (Figures P.1 to P.3) were obtained using the formulae above. Figures P.1 and P.3 are solutions of the same problem; using a different number of nodes and different shape functions. They have, respectively, 44 linear and 10 quadratic elements. As is obvious, higher-order shape functions require fewer nodes for the same accuracy.

Figure P.2 is a 'zoomed' version of Figure P.1. With a constant y-axis, the left half of the x-axis has been expanded to fill the entire screen. This provides a programmable (via the Plot Data input option) selection to observe any portion of the solution in detail.

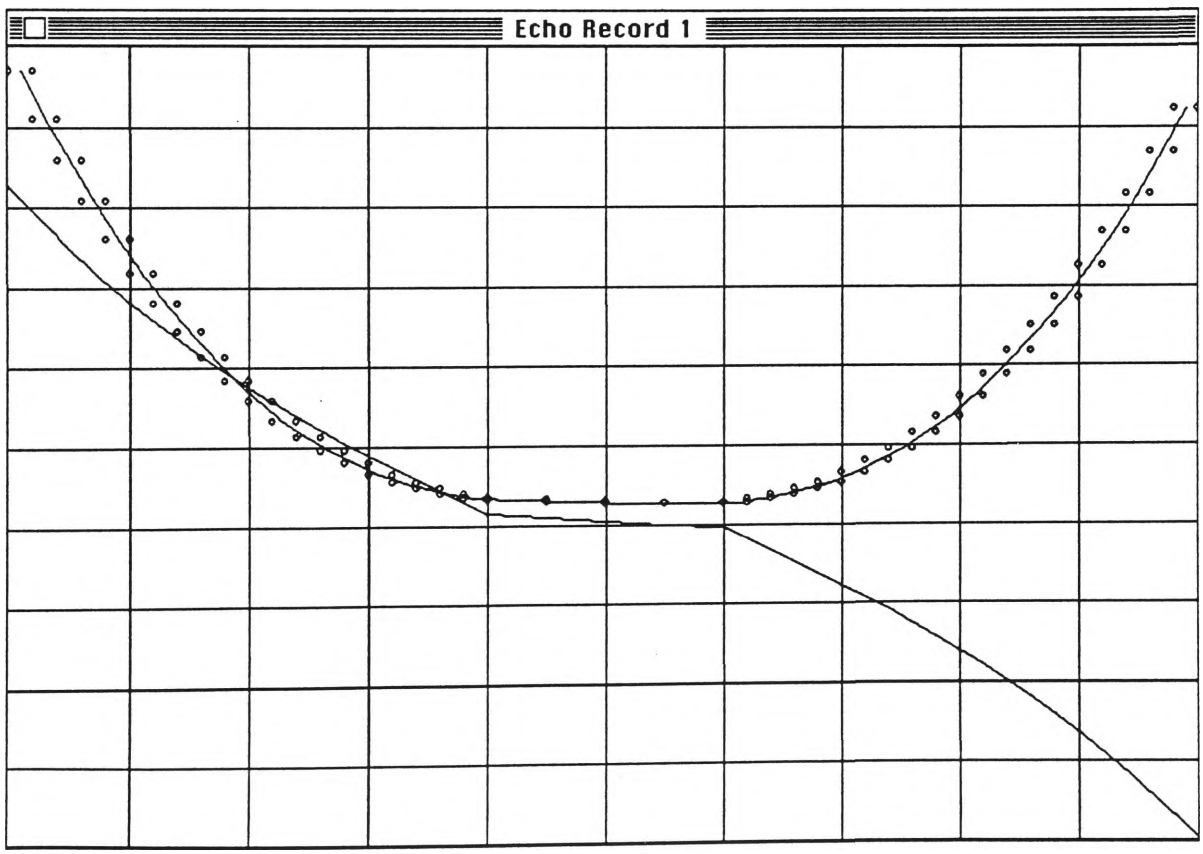


Figure P.1: Display of solution to problem using 44 linear elements.

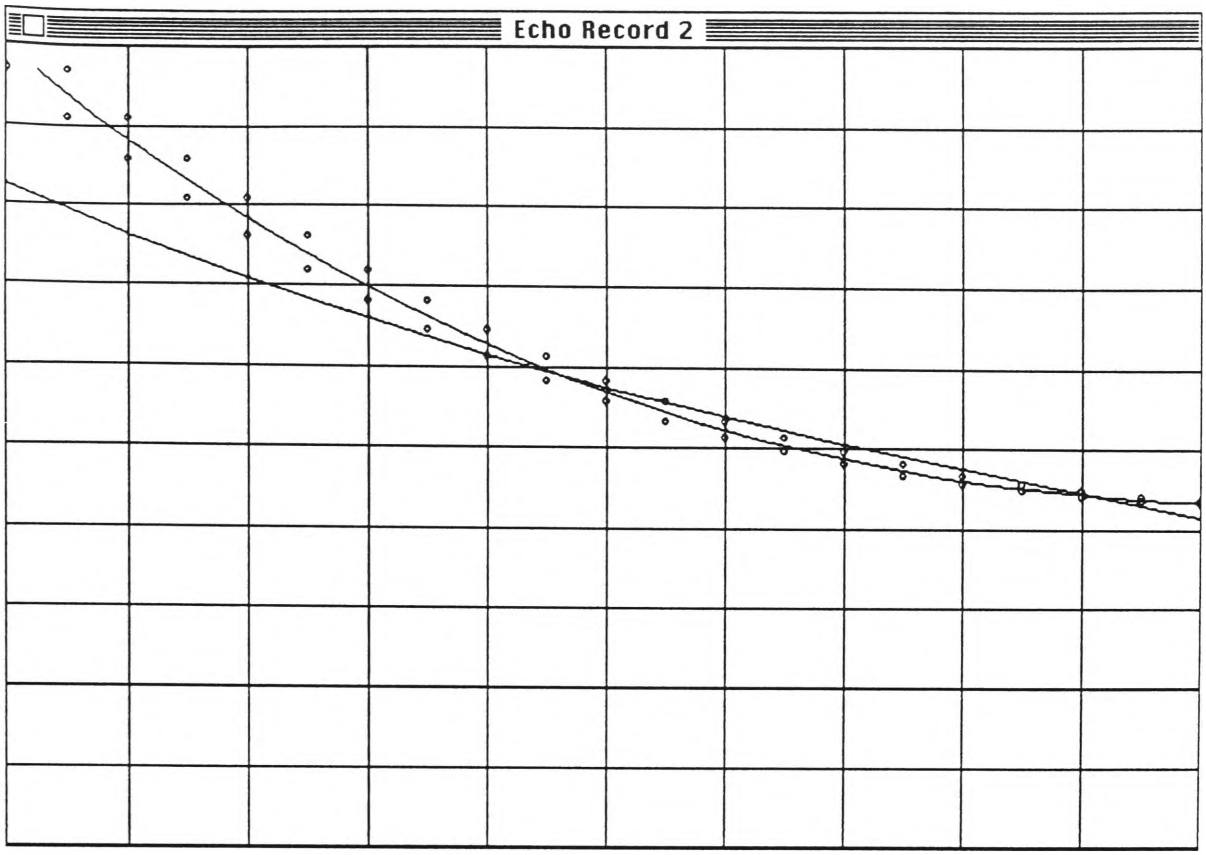


Figure P.2: 'Zoomed' display indicating left 40% of Figure P.1.

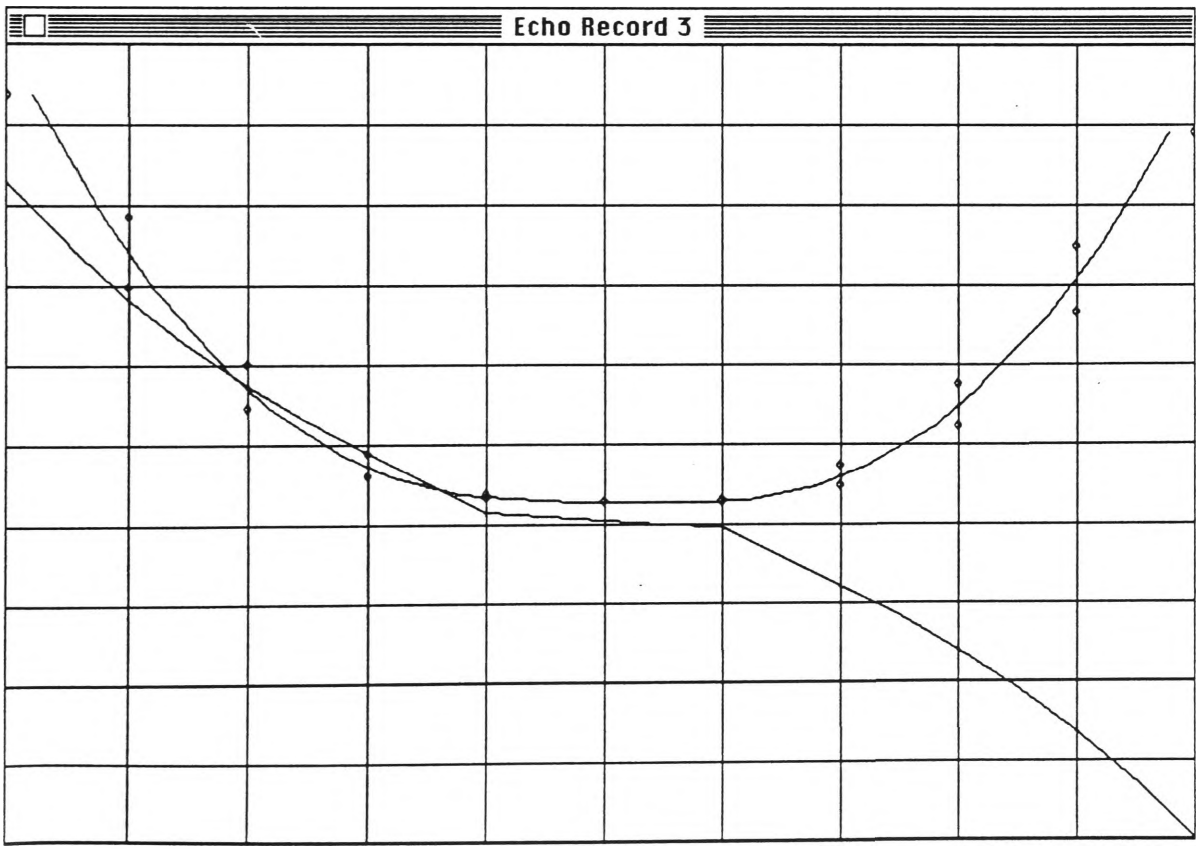


Figure P.3: Display of solution to problem using 10 quadratic elements.

One shortcoming with my implementation is the lack of scales along the axes. These would provide a useful indication of the magnitude of the output.

The numerical results of Figure P.3 are duplicated in the Table P.1, and Microsoft Word 5.1a has been used to graph them in Figure P.4. We note the increase in information content, due to use of the scales along the axes.

Table P.1: Numerical Solution to problem in Figure P.3

Node number	x-coordinate	function value	flux value
1	0.0000E1	0.4125E2	0.9910E-1
2	0.5000E1	0.3740E2	0.8599E-1
3	0.1000E2	0.3409E2	0.7298E-1
4	0.1500E2	0.3122E2	0.6464E-1
5	0.2000E2	0.2870E2	0.5625E-1
6	0.2500E2	0.2646E2	0.5145E-1
7	0.3000E2	0.2441E2	0.4662E-1
8	0.3500E2	0.2251E2	0.4476E-1
9	0.4000E2	0.2068E2	0.4307E-1
10	0.4500E2	0.2045E2	0.4294E-1
11	0.5000E2	0.2022E2	0.4260E-1
12	0.5500E2	0.1999E2	0.4261E-1
13	0.6000E2	0.1975E2	0.4242E-1
14	0.6500E2	0.1796E2	0.4376E-1
15	0.7000E2	0.1611E2	0.4526E-1
16	0.7500E2	0.1413E2	0.4966E-1
17	0.8000E2	0.1197E2	0.5402E-1
18	0.8500E2	0.9558E1	0.6183E-1
19	0.9000E2	0.6817E1	0.6960E-1
20	0.9500E2	0.3665E1	0.8181E-1
21	0.1000E3	0.0000E1	0.9412E-1

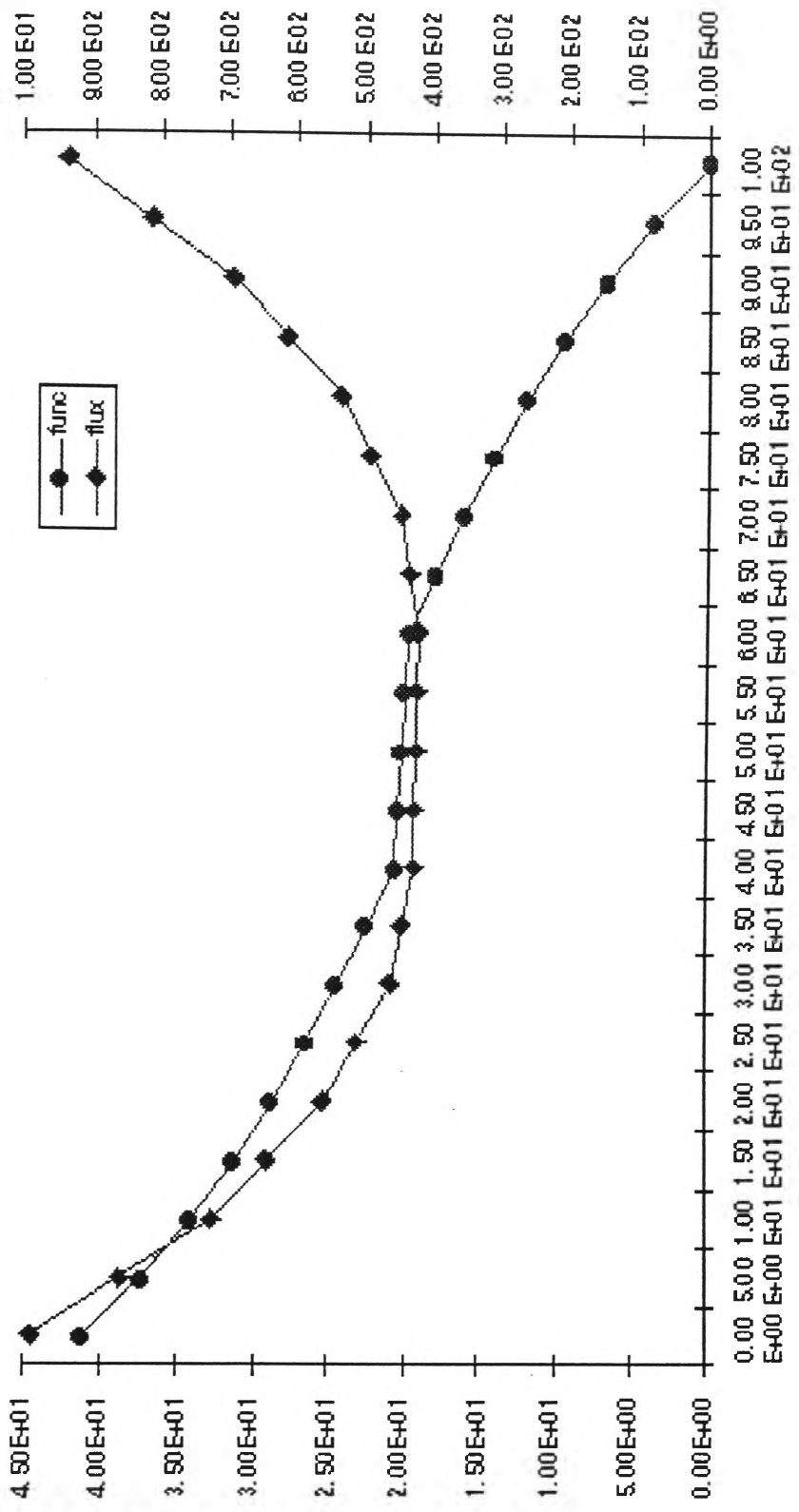


Figure P.4: Word 5.1a display of solution to problem using 10 quadratic elements.

P.3 Eigenresults and Observations

As an example of the use of FAME to study Eigen modelling, consider a cable of length 20, fixed at both ends, modelled by 6 linear elements. Its first three modes are shown below.

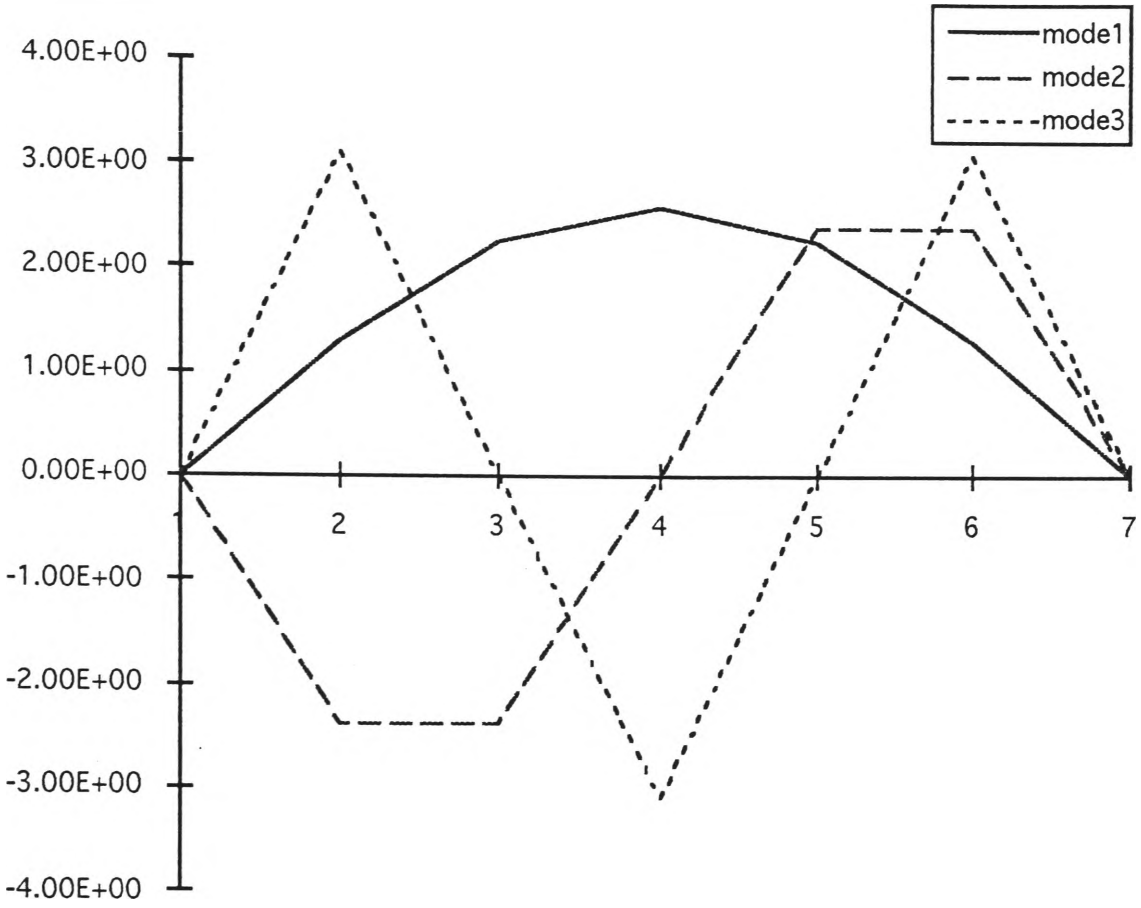


Figure P.5: Word 5.1a display of solution to eigenproblem (first three modes).

Table P.2: Numerical values for the eigenproblem

Node	mode 1	mode 2	mode 3
1	0.0000E1	0.0000E1	0.0000E1
2	0.1298E1	-0.2407E1	0.3108E1
3	0.2248E1	-0.2407E1	0.2742E-6
4	0.2596E1	0.5869E-7	-0.3108E1
5	0.2248E1	0.2407E1	-0.3105E-6
6	0.1298E1	0.2407E1	0.3108E1
7	0.0000E1	0.0000E1	0.0000E1

These are the fourth and fifth mode of the problem stated above.

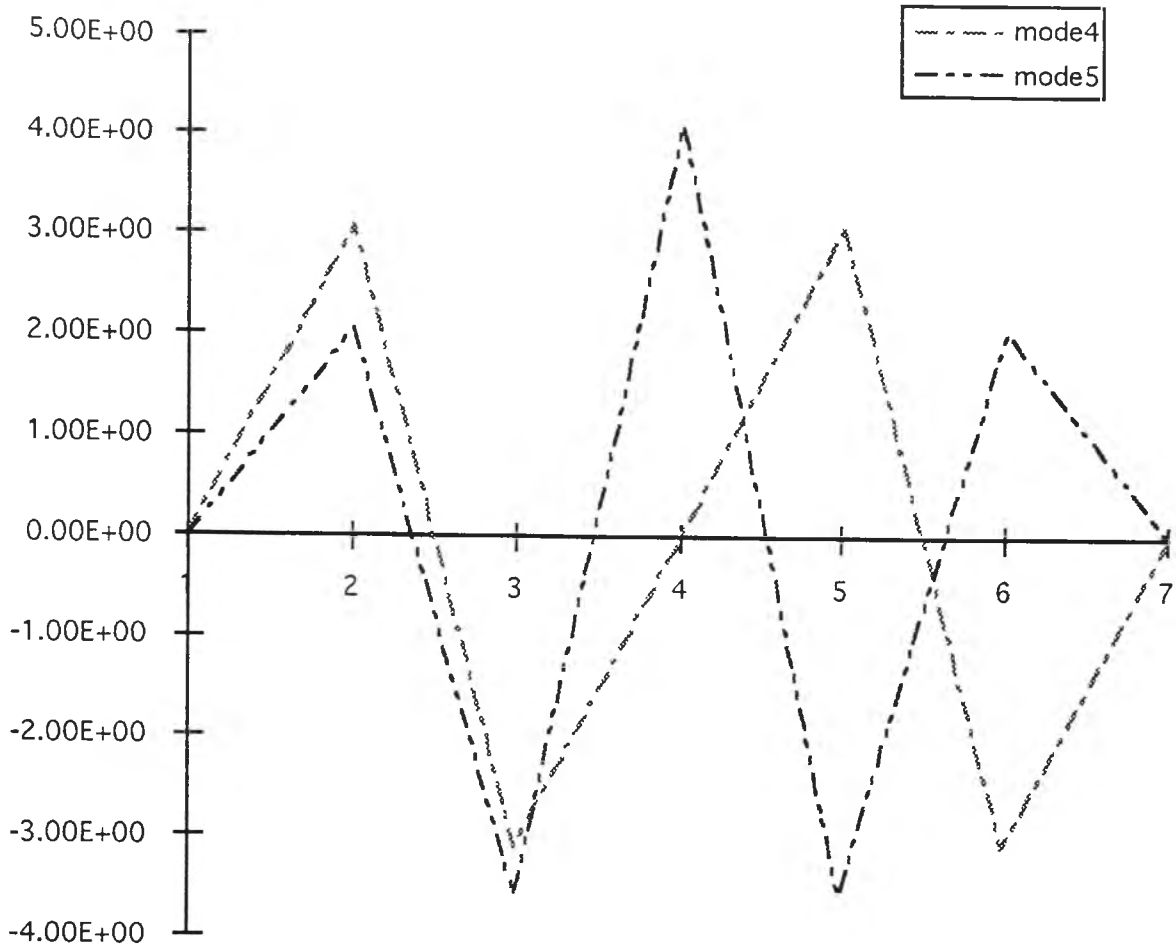


Figure P.6: Word 5.1a display of solution to eigenproblem (fourth and fifth modes).

Table P.3: Numerical values for the eigenproblem

Node	mode 4	mode 5
1	0.0000E1	0.0000E1
2	0.3108E1	0.2064E1
3	-0.3108E1	-0.3574E1
4	0.6081E-6	0.4127E1
5	0.3108E1	-0.3574E1
6	-0.3108E1	0.2064E1
7	0.0000E1	0.0000E1

The modal frequency can be calculated from the eigenvalue by

$$f_i = \frac{1}{2\pi}\sqrt{\lambda_i}, i = 1,2,\ldots \quad .. \quad .. \quad .. \quad .. \quad .. \quad \text{P.5}$$

Table P.4: Modal frequencies for the eigenproblem

Mode number	Eigenvalue	FEM frequency solution (Hz)	Analytical solution (Hz)
1	162.5	2.029	2.006
2	695.4	4.197	4.012
3	1739	6.637	6.018
4	3477	9.385	8.024
5	5722	12.039	10.03

The eigensolution was also obtained by increasing the number of elements (h-refinement) and by increasing the order of the shape functions (p-refinement). Tables P.5 and P.6 summarising the results are reproduced from [Burnett88].

Table P.5: Modal frequencies for the eigenproblem (h-refinement)

Mode	6 linear (7 DOF)	12 linear (13 DOF)	18 linear (19 DOF)	24 linear (25 DOF)	30 linear (31 DOF)
1	2.029	2.012	2.009	2.007	2.007
2	4.197	4.058	4.033	4.024	4.019
3	6.637	6.174	6.088	6.056	6.044
4	9.385	8.395	8.188	8.117	8.083
5	12.039	10.75	10.35	10.21	10.15

Table P.6: Modal frequencies for the eigenproblem (p-refinement)

Mode	6 linear (7 DOF)	6 quadratic (13 DOF)	6 cubic (19 DOF)	6 quartic (25 DOF)
1	2.029	2.006	2.006	2.006
2	4.197	4.015	4.012	4.012
3	6.637	6.042	6.018	6.018
4	9.385	8.112	8.027	8.024
5	12.039	10.27	10.04	10.03

P.4 Discussion - is 8 elements a better minimum?

For reasonably accurate results, it is commonly recommended by the sellers of acoustic packages that a minimum of 6 elements be used per wavelength. But is this sufficient to properly model the problem?

We note from mode 4 (Figure P.6) that a minimum of 3 elements per wavelength is sufficient to indicate the period of the wave under consideration. However, the wave shape is incorrect! Its amplitude exceeds that obtained by use of the analytical solution, which predicts a maximum value of 2.5. From mode 2 (Figure P.5), we observe that a minimum of 6 elements per wavelength gives both period and a rough outline of the wave's shape. However, the maximum amplitude of displacement that the wave could attain is not obtained!

To obtain the maximum value for mode 2, the least change would be to add one more element per half wavelength. With 8 elements per wavelength, we see that we could get both the period and a reasonably good outline of the wave's shape, including the maximum. This seems to indicate that a minimum of 8 elements is preferable to obtain all information regarding the wave.

An obvious counter-argument is that this would only be a valid supposition provided that zero displacement only occurred at the nodes. If it occurred elsewhere, then the possibility exists that the maximum would not be obtained by the use of 8 elements.

A solution to the latter problem is not readily obvious. However, in the event that the maximum amplitude is of importance, I would recommend a minimum of 8 elements per wavelength, and ignore the possibility that it may not apply to the problem under consideration. The marginally extra cost in terms of computation would be well worth the results. Further study may be required to generalise this concept to all problems.