Simulation Methods and Tissue Property Models for Non-Invasive Transcranial Focused Ultrasound Surgery

by

Christopher W. Connor

B.A., M.Eng, M.A. (Hon)
University of Cambridge, 1998.

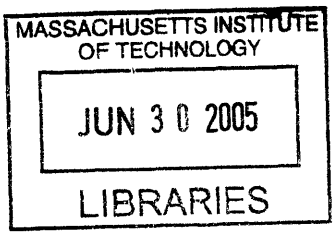
Submitted to the Harvard-MIT Division of Health Sciences & Technology in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2003. [June 2003]



© Massachusetts Institute of Technology, 2003. All Rights Reserved.

Author .....
Harvard-MIT Division of Health Sciences & Technology
January 30, 2003

Certified By .....
Kullervo Hynynen Ph.D
Associate Professor of Radiology, Harvard Medical School
Thesis Supervisor

Accepted By .....
Martha L. Gray Ph.D
Edward Hood Taplin Professor of Medical and Electrical Engineering
Co-director, Harvard-MIT Division of Health Sciences & Technology

ARCHIVES

Simulation Methods and Tissue Property Models for Non-Invasive Transcranial Focused Ultrasound Surgery

by

Christopher W. Connor

Submitted to the Harvard-MIT Division of Health Sciences and Technology
on January 14, 2003, in partial fulfillment of the requirements for
the degree of Doctor of Philosophy.

Abstract

Many brain tumors are localized deeply and are currently surgically inaccessible without causing severe damage to the overlying structures of the brain. The current spectrum of non-invasive methods for treating such tumors includes radiotherapy, which requires exposure to ionizing radiation, and chemotherapy, which is systemically toxic. However, these tumors may also potentially be attacked by focusing highly intense ultrasound onto them. Focused ultrasound surgery is without the side effects of radiotherapy and chemotherapy, and the therapeutic effect of ultrasound therapy can be monitored in real-time using the proton chemical shift MRI technique. However, in order for brain tumors to be treated non-invasively, the ultrasound must be focused onto the targeted brain tissue through the intact cranium.

Transcranial focusing of ultrasound is a longstanding and difficult problem as skull is a highly heterogeneous material. As the ultrasound field propagates through the bones of the skull, it undergoes substantatial distortion due to the variations in density and speed of sound therein. There is substantial individual variation in skull size, thickness and composition. Furthermore, the acoustic attenuation coefficient in bone is high, so the skull may also be heated by the ultrasound propagating through it.

This thesis contains novel simulation techniques for analyzing transcranial acoustic propagation and for analyzing the temperature changes so produced in the brain, skull and scalp. These techniques have also been applied to modeling non-invasive treatment of the liver, and to producing therapeutic ultrasound fields that harness non-linear acoustic effects advantageously. The thesis also contains unified models for the speed of sound and the acoustic attenuation coeffiecient in human skull. These models were generated by combining genetic optimization algorithms, acoustic propagation modeling and empirical measurement of intracranial ultrasound fields; they are valid across the full range of trabecular and cortical cranial bone.

Thesis Supervisor: Kullervo Hynynen
Title: Associate Professor of Radiology, Harvard Medical School

This one's for Mum, Dad and Aurora.

Table of Contents

CHAPTER 1 – INTRODUCTION 17

1.1 INTRODUCTION 18
1.2 THE PREPARATION AND ACQUISITION OF MEDICAL DATA 22
1.2.1 METHODS OF REGISTRATION AND DATA SEGMENTATION 24
1.2.2 CO-ORDINATE FRAMES ARE RELATED BY SOLID BODY TRANSFORMS. 24
1.2.3 RAY SEGMENTATION USES AN O(N) MAXIMAL SUB-ARRAY SEARCH 26
1.2.4 VOLUME SEGMENTATION USES HYSTERESIS AND RECURSION 27
1.3 ACOUSTIC SIMULATION IN HETEROGENEOUS MATERIALS 29
1.3.1 NUMERICAL MODELING WITH PARTIAL DIFFERENTIAL EQUATIONS 29
1.3.2 BOUNDARY CONDITIONS IN FDTD ACOUSTIC ANALYSIS 31
1.3.3 THE NUMERICAL STABILITY OF FDTD METHODS 33
1.3.4 STABILITY AND SIMULATION TIME IN THE CYLINDRICAL MESH 35
1.3.5 ACOUSTIC SIMULATION USING DISTRIBUTED COMPUTING METHODS 36
1.4 REFERENCES 37
1.5 FIGURES 39

CHAPTER 2 – BIOACOUSTIC THERMAL LENSING 46

2.1 ABSTRACT 47
2.2 INTRODUCTION 48
2.3 METHODS 49
2.3.1 CONFIGURATION OF SIMULATIONS FOR THIS PARAMETRIC STUDY 49
2.3.2 METHODS OF SIMULATION 50
2.4 RESULTS 58
2.4.1 THERMAL LENSING EFFECTS ARE ILLUSTRATED BY CASES 1 THROUGH 8 58
2.4.2 NON-LINEAR PROPAGATION EFFECTS ARE ILLUSTRATED BY CASE 9 59
2.5 DISCUSSION AND CONCLUSIONS 60
2.5.1 ON THERMAL LENSING 60

2.5.2 ON NON-LINEAR PROPAGATION EFFECTS 61
2.6 REFERENCES 64
2.7 TABLES 68
2.8 FIGURES 73

CHAPTER 3 – THERAPEUTIC EFFECT OF 2ND HARMONIC FOCAL SPOTS 85

3.1 ABSTRACT 86
3.2 INTRODUCTION 87
3.3 THEORY 88
3.3.1 ACOUSTIC SIMULATION 88
3.3.2 THERMAL SIMULATION 88
3.3.3 TISSUE AND TREATMENT MODELING 88
3.4 RESULTS 90
3.5 CONCLUSION 90
3.6 REFERENCES 91
3.7 TABLES 92
3.8 FIGURES 93

CHAPTER 4 – A UNIFIED MODEL FOR THE SPEED OF SOUND IN CRANIAL BONE 96

4.1 ABSTRACT 97
4.2 INTRODUCTION 98
4.3 METHOD 100
4.3.1 OVERVIEW 100
4.3.2 EMPIRICAL CONFIGURATION 101
4.3.3 GENETIC ALGORITHM REPRESENATION 102
4.3.4 SIMULATION OF ACOUSTIC PROPAGATION 105
4.3.5 THE COMPLETE METHOD BY ACTION 108
4.4 RESULTS 110
4.4.1 STATISTICAL SIGNIFICANCE IN PHASE CORRELATION MEASUREMENTS 110

4.4.2 COMPARISON OF THE GENETIC ALGORITHM SPEED OF SOUND MODEL TO A SPEED OF SOUND MODEL BASED ON THE HOMOGENIZATION OF SKULL. 111
4.5 DISCUSSION 114
4.5.1 TRABECULAR ORIENTATION 115
4.5.2 GENERALIZATION OF APPARENT DENSITY 116
4.5.3 LIMITS ON PHASE-BASED INFERENCE OF SPEED OF SOUND MODELS 117
4.5.4 COMPARISON TO OTHER PRIOR WORK 119
4.6 CONCLUSION 121
4.7 REFERENCES 122
4.8 TABLES 127
4.9 FIGURES 128

CHAPTER 5 – A UNIFIED MODEL FOR ATTENUATION COEFFICIENT IN CRANIAL BONE 139

5.1 ABSTRACT 140
5.2 INTRODUCTION 142
5.3 METHOD 143
5.3.1 OVERVIEW 143
5.3.2 EMPIRICAL CONFIGURATION 144
5.3.3 GENETIC ALGORITHM REPRESENATION 146
5.3.4 SIMULATION OF ACOUSTIC PROPAGATION 149
5.3.5 HOMOGENIZING CT DATA AND ATTENUATION COEFFICIENTS 151
5.3.6 ASSESSING THE PERFORMANCE OF ACOUSTIC ATTENUATION MODELS 152
5.4 RESULTS 154
5.4.1 ROBUSTNESS OF THE ACOUSTIC ATTENUATION MODELS. 154
5.4.2 COMPARISON OF THIS ALGORITHM TO ALGORITHMS BASED ON HOMOGENIZATION OF DENSITY AND ATTENUATION 155

5.5 DISCUSSION **156**
5.6 CONCLUSION **159**
5.7 REFERENCES **160**
5.8 TABLES **163**
5.9 FIGURES **169**

CHAPTER 6 – PATTERNS OF THERMAL DEPOSITION IN THE SKULL **180**

6.1 ABSTRACT **181**
6.2 INTRODUCTION **182**
6.3 METHOD **183**
6.3.1 STRUCTURE OF THE ACOUSTIC SIMULATIONS 184
6.3.2 FINITE DIFFERENCE-TIME DOMAIN (FDTD) SIMULATION OF TRANSCRANIAL PROPAGATION 185
6.3.3 THE ACOUSTIC PROPERTIES OF CRANIAL BONE 186
6.3.4 COMPLETING AND INITIATING SIMULATIONS WITH THE HUYGENS-FRESNEL INTEGRAL 187
6.3.5 HIERARCHICAL MULTIRESOLUTION MODELING OF SOURCES FOR ACOUSTIC SIMULATION 190
6.3.6 FOURIER CAPTURE IN FDTD ANALYSIS 191
6.3.7 FROM ACOUSTIC FIELDS TO THERMAL DEPOSITION 192
6.3.8 THERMAL CONDUCTIVITY IN CRANIAL BONE 193
6.3.9 MODELING THERMAL DAMAGE AND THERAPEUTIC VOLUME IN TISSUE 194
6.3.10 CONFIGURATION OF SIMULATIONS AS A PARAMETRIC STUDY 195

6.4 RESULTS 197
6.5 DISCUSSION 199
6.6 CONCLUSION 202
6.7 REFERENCES 203
6.8 TABLES 207
6.9 FIGURES 211

CHAPTER 7 – FUTURE DIRECTIONS 224

7.1 VALIDATING AND INCREASING THE SPEED OF ANALYSIS TOWARDS ON-LINE USE OF 3D MODELS. 225
7.2 THE VARIATION OF SPEED OF SOUND WITH FREQUENCY IN BONE. 226
7.3 THE VARIATION OF SPEED OF SOUND AND ACOUSTIC ATTENUATION COEFFICIENT WITH APPARENT DENSITY IN BONE WITH ORIENTATIONAL DEPENDENCY. 227
7.4 FORMULATING AN AMPLITUDE SHADING ALGORITHM FOR TRANSCRANIAL ULTRASOUND SURGERY 227
7.5 DETERMINING THERAPEUTIC LIMITS FOR TRANSCRANIAL FOCUSED ULTRASOUND SURGERY 229

CHAPTER 8 – ACOUSTIC SIMULATION SOURCE CODE 230

8.1 SOURCE CODE FOR A 3D ACOUSTIC SOLVER BASED UPON THE HUYGENS-FRESNEL INTEGRAL AND THE FDTD FORMULATION OF THE WESTERVELT EQUATION 231
8.2 A COMMAND LINE ROUTINE FOR TESTING RAINFALL APPLICATIONS ON A LONE WORKSTATION 297

List of Figures

FIGURE 1.1: A FRAME MOUNTED ONTO A CALVARIUM. 40

FIGURE 1.2: A CT SLICE OF A CALVARIUM MOUNTED IN A FRAME..... 41

FIGURE 1.3: THE 64 ELEMENT HEMISPHERICAL ARRAY. 42

FIGURE 1.4: A TYPICAL ONE-DIMENSTIONAL APPARENT DENSITY TRACE FROM THE SURFACE OF THE TRANSDUCER THROUGH THE SKULL TO THE FOCUS POINT. 43

FIGURE 1.5: EXTRACTION OF THE SEGMENTED SKULL TO A CYLINDRICAL MESH..... 44

FIGURE 1.6: THE RAINFALL SERVER USER INTERFACE..... 45

FIGURE 2.1: ARRANGEMENT OF TISSUES AND THE SECTOR VORTEX ARRAY..... 74

FIGURE 2.2: TEMPERATURE DEPENDENCE OF SOUND SPEED IN SOFT TISSUE..... 75

FIGURE 2.3: TEMPERATURE DEPENDENCE OF ABSORPTION IN SOFT TISSUE..... 76

FIGURE 2.4: GRID STRUCTURES FOR FINITE DIFFERENCE TIME DOMAIN (FDTD) SIMULATION. 77

FIGURE 2.5: THERMAL FIELDS UNDER SONICATION AT 666KHZ, 1MHZ AND 1.2MHZ 78

FIGURE 2.6: THERMAL FIELD UNDER SONICATION FROM TRANSDUCERS WITH FOCAL NUMBERS 0.8, 1.0, 1.2, 1.4 79

FIGURE 2.7: THERMAL FIELDS UNDER MODE 0, MODE 1 AND MODE 2 SONICATION..... 80

FIGURE 2.8: SIMULATION OF MODE 4 SONICATION AT 1MHZ WITH CLASSICAL LINEAR THEORY. 81

FIGURE 2.9: SIMULATION OF MODE 4 SONICATION AT 1MHZ WITH NONLINEAR THEORY. 82

FIGURE 2.10: THERMAL FIELD PRODUCED BY NON-LINEAR ACOUSTIC PROPAGATION AND LINEAR ACOUSTIC PROPAGATION WITHOUT THERMAL LENSING UNDER SONICATION FROM A TRANSDUCER WITH FOCAL NUMBER 1.0 AT 1 MHZ IN MODE 1. 83
FIGURE 2.11: FREQUENCY SEPARATION OF THE MODE 4 FIELD PRODUCED AT 1MHZ WITH NONLINEAR THEORY....84
FIGURE 3.1: THERMAL PROFILE IN LIVER AFTER 15S SONICATION AT 2.5 BAR USING LINEAR AND NONLINEAR SIMULATION METHODS. 85
FIGURE 3.2: COMPARISON OF THERAPEUTIC OUTCOME FOR MODE 0 AND MODE 4 TREATMENTS NOT EXCEEDING 80°C TEMPERATURE. 86
FIGURE 4.1: A HUMAN CALVARIUM MOUNTED IN A STEREOTACTIC POSITIONING FRAME....129
FIGURE 4.2: EXAMPLES FROM THE POPULATION OF POTENTIAL SPEED-OF-SOUND MODELS. 130
FIGURE 4.3: A COMPUTER RECONSTRUCTION OF A HUMAN CALVARIUM SHOWING ITS POSITIONING WITHIN A HEMISPHERICAL TRANSDUCER....131
FIGURE 4.4: BLOCK DIAGRAM OF THE GENETIC ALGORITHM OPTIMIZATION OF SPEED-OF-SOUND MODEL....132
FIGURE 4.5: THE FINAL MODEL FOR SPEED OF SOUND IN CRANIAL BONE BASED ON APPARENT DENSITY BY CT SCAN....133
FIGURE 4.6: THE PROBABILISTIC DISTRIBUTION OF THE PHASE CORRELATION METRIC FOR RANDOMLY ASSIGNED PHASES....134
FIGURE 4.7: STATISTICAL SIGNIFICANCE CURVE FOR THE PHASE CORRELATION METRIC. 135

FIGURE 4.8: COMPARISON OF THE PERFORMANCE OF THE LINEAR SPEED MODEL AND THE GENETIC ALGORITHM SPEED MODEL FOR A SKULL NOT USED IN THE OPTIMIZATION PROCESS....136
FIGURE 4.9: CT CROSS-SECTION SHOWING AN INSTANCE OF COMPLEX LACUNAR STRUCTURE....137
FIGURE 4.10: SPEED OF SOUND LIMITS IN PHASE BASED OPTIMIZATION....138
FIGURE 5.1: A HUMAN CALVARIUM MOUNTED IN A STEREOTACTIC POSITIONING FRAME....170
FIGURE 5.2: SCHEMATIC DIAGRAM OF THE EXPERIMENTAL APPARATUS....171
FIGURE 5.3: BLOCK DIAGRAM OF THE GENETIC ALGORITHM OPTIMIZATION OF THE AMPLITUDE ATTENUATION MODEL....172
FIGURE 5.4: STAGES OF EVOLUTION DURING GENETIC ALGORITHM OPTIMIZATION OF THE ATTENUATION COEFFICIENT IN CRANIAL BONE AT AN ULTRASOUND FREQUENCY OF 500 KHZ....173
FIGURE 5.5: FOCAL PLANE OF THE ULTRASOUND BEAM WITH HALF-POWER BEAM WIDTH....174
FIGURE 5.6: THE FINAL MODELS FOR AMPLITUDE ATTENUATION COEFFICIENT IN CRANIAL BONE BASED ON APPARENT DENSITY BY CT SCAN AND ULTRASOUND FREQUENCY....175
FIGURE 5.7: THE RELATIONSHIP BETWEEN APPARENT DENSITY OF CRANIAL BONE BY CT SCAN, ULTRASOUND FREQUENCY AND ATTENUATION COEFFICIENT AS A THREE-DIMENSIONAL SURFACE....176

FIGURE 5.8: COMPARISON OF THE PERFORMANCE OF THE GENETIC ALGORITHM ATTENUATION MODEL TO DERIVED HOMOGENIZED ATTENUATION MODELS.....177
FIGURE 5.9: GRAPH OF BULK ATTENUATION COEFFICIENT AGAINST ULTRASOUND FREQUENCY.....178
FIGURE 5.10: GRAPH OF DENSITY OF MINIMUM ATTENUATION IN CRANIAL BONE AGAINST ULTRASOUND FREQUENCY.....179
FIGURE 6.1: 3D RECONSTRUCTION OF CALVARIUM MOUNTED IN THE 500-ELEMENT HEMISPHERICAL ARRAY.....213
FIGURE 6.2: THE INTERSECTION OF ACOUSTIC SIMULATION VOLUMES, SHOWN FOR A 64-ELEMENT HEMISPHERICAL ARRAY.....214
FIGURE 6.3: DISTRIBUTION OF ACOUSTIC PRESSURE WITHIN THE SCALP, CALVARIUM AND BRAIN.....215
FIGURE 6.4: THERMAL DEPOSITION DURING TRANSCRANIAL FOCUSED ULTRASOUND WITH ACTIVE PRE-COOLING.....216
FIGURE 6.5: PATTERNS OF THERMAL DEPOSITION WITHIN THE SKULL WITH ACTIVE PRE-COOLING.....217
FIGURE 6.6: PATTERNS OF THERMAL DEPOSITION WITHIN THE DENUDED SKULL WITH ACTIVE PRE-COOLING.....218
FIGURE 6.7: PATTERNS OF THERMAL DEPOSITION WITHIN THE DENUDED SKULL WITHOUT PRE-COOLING.....219
FIGURE 6.8: THERMAL DISTRIBUTION AND TEMPERATURE PROFILE IMMEDIATELY AFTER SONICATION IN A SKULL OF INTERMEDIATE THICKNESS.....220
FIGURE 6.9: LOCATION AND HEATING EFFECT OF STANDING WAVES WITHIN THE SKULL.....221
FIGURE 6.10: DOSAGE RELATIONSHIPS FOR SKULL OF INTERMEDIATE THICKNESS ORIENTED CENTRALLY.....222

FIGURE 6.11: EFFECT OF ACTIVE AND PASSIVE COOLING ON TEMPERATURE WITHIN THE SKULL AGAINST TIME.....223

List of Tables

TABLE 2.1: THE PHASE SHIFTS FOR EACH ELEMENT WITH RESPECT TO MODE FOR AN 8 ELEMENT SECTOR VORTEX ARRAY. 69

TABLE 2.2: THE PARAMETRIC CONFIGURATION OF THE SIMULATIONS IN THIS STUDY. 70

TABLE 2.3: TYPICAL MATERIAL PROPERTIES OF RELEVANT HUMAN TISSUES 71

TABLE 2.4: NEGLECTING THERMAL LENSING INTRODUCES A DISCREPANCY IN THE POSITION OF THE ACOUSTIC FOCUS. THIS EFFECT IS MOST DEPENDENT OF THE FOCAL NUMBER OF THE TRANSDUCER. 72

TABLE 3.1: TYPICAL MATERIAL PROPERTIES OF RELEVANT HUMAN TISSUES 92

TABLE 4.1: SPEED OF SOUND IN CRANIAL BONE AS A FUNCTION OF APPARENT BONE DENSITY BY CT SCAN IN INTERVALS OF 100 KG M^{-3} 127

TABLE 5.1: THE GENETIC ALGORITHM FORM OF THE RELATIONSHIPS BETWEEN ACOUSTIC ATTENUATION COEFFICIENT, APPARENT BONE DENSITY AND ULTRASOUND FREQUENCY. 164

TABLE 5.2: THE ACOUSTIC ATTENUATION COEFFICIENT IN CRANIAL BONE AS A FUNCTION OF ULTRASOUND FREQUENCY AND APPARENT BONE DENSITY BY CT SCAN IN INTERVALS OF 100 KG M^{-3} 165

TABLE 5.3: COMPARISON OF GENETIC ALGORITHM TRAINING DATA TO GENETIC ALGORITHM TEST DATA. 166

TABLE 5.4: EQUIVALENT BULK ATTENUATION COEFFICIENT PARAMETERS FOR CRANIAL BONE BASED ON THE FUNCTIONS PRODUCED BY GENETIC ALGORITHM OPTIMIZATION. 167

TABLE 5.5: COMPARISON OF THE PERFORMANCE OF THE ALPHA MODEL BASED ON DENSITY TO HOMOGENIZED MODELS. 168
TABLE 6.1: ACOUSTIC PROPERTIES OF CRANIAL BONE BASED ON APPARENT CT DENSITY. 208
TABLE 6.2: MATERIAL AND ACOUSTIC PROPERTIES FOR WATER AND BRAIN. 209
TABLE 6.3: UNIFORM EXCITATION PRESSURE OVER THE 500-ELEMENT ARRAY TO PRODUCE A PEAK TEMPERATURE OF 70°C AFTER A 20 SECOND SONICATION. 210

Chapter 1

Introduction

1.1 INTRODUCTION

It has long been known that high-powered ultrasound, if focused appropriately, can be used to attack deep-seated tumors non-invasively; the tumor tissue can be killed within the patient without breaking the patient's skin. The energy of the ultrasound beam is deposited at the focus as heat. This causes the tissue temperature of the tumor to rise, and cell death by thermal coagulation results. Temperature can be monitored non-invasively and in real-time by using a proton chemical shift MRI scan [1]. This approach is now beginning to attain clinical use; present clinical trials include treatment of prostate [2] and breast tumors [3]. Application of this technique to inoperable brain tumors is highly desired.

Many brain tumors, such as those localized in the deep gray matter, brain stem, thalamus or motor areas, are currently surgically inaccessible without causing severe damage to the overlying structures of the brain. These tumors may be attacked by focusing high-powered ultrasound onto them, generated by a large hemispherical transducer placed over the patient's head [4]. This transducer is comprised of many individually drivable piezoelectric elements, arranged and driven in such a way that the transducer focuses the ultrasound through the patient's skull, into the brain and onto the tumor. This would be a non-invasive treatment technique for neoplastic growth within the brain.

Further applications of this technique may improve other current neurosurgical techniques. As an example, tremors caused by Parkinson's disease are treated by inserting a thermal ablation probe into the patient's basal ganglia to lesion the globus pallidus internus [5]. With focused ultrasound, this potentially could be achieved non-invasively. Additionally, focused ultrasound has been shown to increase uptake of therapeutic agents in targeted tissues [6] and its use may therefore be indicated as an adjunct to chemotherapy, increasing the action of a chemotherapeutic compound on a targeted tumor.

To create a sufficiently localized lesion in the brain, the ultrasound field from each of the transducer elements must coincide nearly in phase at the targeted point to interfere constructively and produce a high-intensity focus. This is of little concern in the breast and prostate since the overlying fatty tissue in the breast case and the trans-rectal tissue in the prostate case are largely uniform to ultrasound. However, the bones of the skull introduce large distortions into a trans-cranial ultrasound field [7], which make it decidedly non-trivial to produce a therapeutic effect within the brain. Nevertheless, ultrasound field configurations exist which produce a sufficiently powerful focus inside the brain to kill targeted tissue [8]. Ultrasound with a frequency between 0.5 MHz and 1 MHz is most appropriate for transcranial use since this frequency band undergoes much less absorption in the skull than higher frequencies [7].

The task, therefore, is to infer *a priori* what distortions are introduced into the ultrasound field and to determine what the relative phase shift at each transducer element should be, and to a lesser extent what their relative amplitudes should be. The solution will clearly vary from individual to individual, dependent on the geometry and composition of the skull in each case and the relative positions of the tumor, skull and transducer.

This is a long-standing problem within the field of focused ultrasound research, and one which has seen its fair share of false dawns, but the twin availabilities of increasingly high resolution medical imaging and economical computational power now allow novel approaches whereby the genesis and transmission of the focused ultrasound field may be completely simulated. The research presented here advances this goal in several steps, as follows.

A novel method of performing three-dimensional simulations using FDTD methods on a cylindrical mesh is introduced in Chapter 2. This method is used to analyze the effect of non-invasive sector-vortex therapeutic sonication of liver tissue. The biological tissues present in these simulations are relatively homogeneous. Consequently the work presented in Chapter 2 can be largely corroborated by analytical methods, verifying the numerical method. Chapter 2 also presents an interesting result in non-linear acoustics. If

an n-element sector-vortex array is driven with Mode n/2 sonication, then non-linear effects can cause an extra therapeutic focus to be produced within the liver. Straightforward linear acoustic methods fail to predict the existence of this focus.

Chapter 3 enlarges on the non-linear result in the previous chapter and demonstrates that this secondary non-linear focus can be harnessed to produce a useful therapeutic effect. The work in Chapter 3 also establishes that if a sonication is to be limited by the peak temperature that can be reached in the tissue, the use of non-linear therapeutic effects in a Mode 4 sonication can increase the volume of tissue treated per sonication by a factor of almost fifty over the use of a Mode 0 sonication. While one might attempt to achieve a equivalent effect therapeutically by superposing a Mode 4 sonication linearly upon a Mode 0 sonication, this configuration would then underestimate the heating caused by non-linear propagation of the Mode 4 component.

Chapter 4 uses a novel method of coupling genetic algorithm optimization to the simulation of acoustic propagation to produce a unified model for the speed of sound in cranial bone as it varies with apparent density. Research on the propagation of high intensity focused ultrasound through the intact human skull requires a detailed model for the acoustic velocity in cranial bone. The model presented in Chapter 4 is valid in cranial bone of both trabecular and cortical form, and has been shown to predict the speed of sound in transcranial propagation better than pre-existing homogenized speed of sound methods.

Chapter 5 uses an expanded version of the method developed in Chapter 4 to produce a model for the acoustic attenuation coefficient in cranial bone based upon the apparent density of bone determined by CT scan and also upon ultrasound frequency. This acoustic attenuation coefficient is very important for establishing dosimetry in focused ultrasound treatment of the brain through an intact skull. Firstly, the acoustic energy transmitted through the skull into the brain is dependent on the acoustic attenuation in the skull. Secondly, acoustic absorption in the skull causes skull heating; this places a

practical upper limit on the acoustic power that can be used for transcranial therapy in patients.

Chapter 6 brings together the simulation methods developed in Chapter 2 with the tissue property models for human skull developed in Chapter 4 and Chapter 5 to evaluate the patterns of thermal deposition in the skull, scalp and brain that a patient would experience during transcranial focused ultrasound surgery. The nature and extent of this heating has not previously been characterized as it is practically difficult to implant a sufficient number of thermocouples to obtain detailed temperature data directly, and bone is an unsuitable medium in which to perform non-invasive thermometry using proton chemical shift MRI.

Chapter 7 considers the results in the previous chapters, suggests future lines of academic enquiry and possible technical improvements based upon the research presented here, and attempts to foresee future developments in the field. Chapter 8 contains the complete source code for the acoustic propagation solver used in Chapter 6.

1.2 THE PREPARATION AND ACQUISITION OF MEDICAL DATA

The experimental model of the human skull used in this thesis is an axial section though the skull, taken anteriorly through the forehead above the eyes and proceeding posteriorly above the mandibular joint. This preparation is the *calvarium*, the roof of the skull. The anatomical gifts program at Harvard Medical School provided the human calvaria used in the experiments detailed herein. Real anatomical specimens are necessary as the shape and structure of the human calvarium cannot be satisfactorily modeled as homogeneous, regular geometry for the purposes of acoustical simulation; the human calvarium must be modeled by CT scanning real calvaria.

In preparation for being CT scanned [9], a calvarium is fixed into a rigid polycarbonate frame as shown in Figure 1.1. This frame has several points that will serve as fiducial markers in the CT data. It will be important later to be able to determine the position and orientation of the skull in the CT data set; the frame serves as both a support and a reference. This is similar to current practice in Radiation Oncology, where a joint CT and MRI study (a so-called fusion study) is performed on the patient; the patient may have a non-ferromagnetic reference frame screwed into her skull to serve as a reference frame and platform for radiation treatment. This frame is often referred to as a halo ring, for obvious esthetic reasons.

The calvarium and frame are placed in a large, water-filled Perspex trough for the CT scan. This ensures that the pores of the bone remain full of fluid, which is the most appropriate model for a living human. The CT scanning is performed at Brigham & Women's Hospital, using a Siemens Somatom Plus 4 Scanner, using the AH-82 bone kernel, which most accurately renders the structure of dense, mineralized tissues. The CT image shown in Figure 1.2 is a typical slice from a scan of a calvarium.

Using an axial slice separation of 1 mm, approximately 200 images are required to cover the calvarium and frame. This initial data set occupies around 100 MB and is stored in the DICOM medical image format. Further processing is required to convert these data

into a more usable form. Initially, the data is extracted to a text header file detailing the parameters of the CT scan and to a raw data number file.

CT machines do not produce measurements of density; they return a data number measured in somewhat arbitrary Hounsfield units (HU). However, a linear correlation can be formed between density and HU, achieved by sampling a block of water and a block of air in the CT image. If *water* and *air* are the mean values in HU in water and air respectively, then for any particular value in HU:

$$Density\ (\mathrm{g\,cm^{-3}}) = \frac{1}{water - air}(HU - air) \tag{1.1}$$

All the image slices are converted into density measurements and then combined into two three-dimensional matrices. The larger matrix contains the full data set and the smaller matrix contains a down-sampled version suitable as the basis for rendering in a user interface.

Once the CT scan is complete, the frame and calvarium are attached to the ultrasound transducer that will be used for the focused ultrasound procedure. Two transducers are available: one comprises 64 individual elements and was constructed in lab, the second comprises 500 individual elements and was constructed by Imasonic of Besancon, France [9]. Both transducers are hemispherical [10], and both have an internal radius of 150 mm. The 64-element transducer is shown in Figure 1.3.

1.2.1 METHODS OF REGISTRATION AND DATA SEGMENTATION

Two different segmentation methods are used for extracting the skull geometry and density from CT data. The first segmentation method is one-dimensional and deals solely with a single ray passing through the skull. The second method is used for the segmentation of volumes. However, before segmentation is performed, it is helpful to establish the mathematical relationship between the co-ordinate frame of the CT data and the co-ordinate frame of the ultrasound array.

1.2.2 CO-ORDINATE FRAMES ARE RELATED BY SOLID BODY TRANSFORMS.

The act of attaching the frame and calvarium to the ultrasound array creates a fixed relationship between the co-ordinate frame of the array and the co-ordinate frame of the CT data. By measuring the position of fiducial markers relative to the array and by locating them in the CT data set, a representation of the array can be mapped onto to the CT data. This is the very first step in calculating the acoustic propagation. The transformation between the co-ordinate frame of the array and that of the CT data is necessarily a Solid Body Transform (hereafter, SBT). It is clear that neither the skull nor the halo nor the array are physically distorted at any time. In the simplest case, 3 points that are not co-linear must be found in the array co-ordinate frame, and the same three points located in the CT data co-ordinate frame. Determining the position of three fiducial points in the frame of reference of the array is merely a matter of physical measurement. Determining the position of the three points in the CT data involves rendering the frame and calvarium in 3D and rotating the data interactively to pick out each fiducial marker in turn in two different projections. Selecting a point in a 3D projection in fact merely selects a ray that passes from the imaging camera through that point. Consequently, the point must be identified in two separate projections, which creates two rays that notionally intersect at that point of interest. In practice, the two rays usually pass closely by each other, but do not directly intersect. The point is therefore

determined by finding that point which lies closest to the two lines and minimizes the sum-squared error.

Given two rays which may or may not intersect:

$$\begin{aligned} \underline{y}_1 &= \underline{c}_1 + \lambda \underline{m}_1 \\ \underline{y}_2 &= \underline{c}_2 + \lambda \underline{m}_2 \end{aligned} \tag{1.2}$$

the selected point is that point which uniquely satisfies the intersection of these rays:

$$\begin{aligned} \tilde{\underline{y}}_1 &= (\underline{c}_1 + 0.5\mu\underline{e}) + \lambda \underline{m}_1 \\ \tilde{\underline{y}}_2 &= (\underline{c}_2 - 0.5\mu\underline{e}) + \lambda \underline{m}_2 \end{aligned} \quad \text{where} \quad \begin{aligned} \underline{e} &= \underline{m}_1 \times \underline{m}_2 \\ \mu &= (\underline{c}_2 - \underline{c}_1) \cdot \underline{e} \end{aligned} \tag{1.3}$$

Having located the three points in the frame of the array and in the frame of the CT data, the task becomes that of identifying a solid body transform such that:

$$\begin{bmatrix} \vdots & \vdots & \vdots \\ \underline{x}_1^{AR} & \underline{x}_2^{AR} & \underline{x}_3^{AR} \\ \vdots & \vdots & \vdots \\ \hline 1 & 1 & 1 \end{bmatrix} = \left[\begin{array}{ccc|c} \ddots & & \iddots & \vdots \\ & \mathbf{R} & & \mathbf{T} \\ \iddots & & \ddots & \vdots \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} \vdots & \vdots & \vdots \\ \underline{x}_1^{CT} & \underline{x}_2^{CT} & \underline{x}_3^{CT} \\ \vdots & \vdots & \vdots \\ \hline 1 & 1 & 1 \end{bmatrix} \tag{1.4}$$

where *AR* and *CT* indicate the array and CT data reference frames respectively, and the 3-by-3 sub-matrix **R** is constrained to be orthonormal. To attempt to calculate the appropriate matrix element-by-element would be challenging, given that the constraint on **R** depends upon the dot products of its rows and is therefore algebraically non-linear. Furthermore, there may very well be no correct solution to the problem. Any error in the position of any one of the points may produce a problem in which no SBT can map the given points precisely. Yet we know that the solution to the problem must be an SBT; no other solution has an appropriate physical meaning. The relationship between the two co-ordinate frames is determined by finding the values of orthonormal **R** and **T** that best satisfy this equation. Fortunately, this problem can be solved in closed form using Horn's

algorithm [11] by translating the problem into its equivalent form in quaternions; this algorithm always produces the solid body transform that is the least-squared-error solution.

1.2.3 RAY SEGMENTATION USES AN O(N) MAXIMAL SUB-ARRAY SEARCH

For each ray extending from the transducer to the focus point, the densities of the materials along that ray are extracted from the high resolution, density-corrected CT data. Figure 1.4 shows a typical trace. The area in which no density data is returned lies outside the range of the original CT scan; the density in this region is given the IEEE constant NaN (Not-A-Number). The area where the density is approximately zero is the air above the water in the trough in which the calvarium was scanned. Density values around 1.0 g cm^{-3} are obviously water. The density section through the skull shows the characteristic M shape caused by the presence of a lower density trabecular layer sandwiched between two higher density cortical layers. These will be the subjects of some discussion later on the material properties of the bones of the skull. The distance from transducer to focus on the trace is 150 mm because that is the internal radius of the hemispherical transducer.

The task is to recognize that part of the trace that describes bone, and define all the rest of the data as water. This task is frequently complicated by spikes of noise in the water data, and a potentially wide and deep drop in density in the center of the bone in the trabecular region. The cleanest way to segment the skull from the water is to subtract a density slightly more dense than water from the density data, and then locate the range of resulting shifted density data that has the largest algebraic sum. This approach proves to be very tolerant of both noise spikes in the water and for trabecular regions with very low density. A sharp spike is insufficient to distract the segmentation process, and a dip in the trabecular bone density below the threshold value is more than offset by the higher density of the cortical bones surrounding it.

Initially it appears that finding the maximal sub-array of a data set would be an $O(n^2)$ problem, requiring each possible sub-array to be searched. However, a more subtle algorithm allows this task to be accomplished in O(n). The basis of the algorithm is the maintenance of a current best answer and the realization that any range being scanned must be a part of any larger solution provided that its current algebraic sum is positive. This is obviously true in the converse case, since if the range being scanned were removed from some hypothetical larger solution, the sum of the larger solution would be decreased.

1.2.4 VOLUME SEGMENTATION USES HYSTERESIS AND RECURSION

The method of segmenting a volume of data is necessarily more computationally simple than the ray case given the number of points that need to be processed. A typical CT data set has a data volume of 512 by 512 by 200 points; a total of some 52.4 million points. One algorithm that produces acceptable results is scanning the data volume in a caudo-cranial direction and segmenting the data into bone or non-bone using a hysteresis method, by analogy to a Schmidt trigger. The advantage with this method is that it is computationally very light, being entirely expressible in Boolean logic. Given two Boolean values, *SchmidtUp* and *SchmidtDown*, set when the local density is respectively above the higher threshold value or below the lower threshold value, we define the logical variable *Bone* as:

$$Bone^n = SchmidtUp + Bone^{n-1} \cdot \overline{SchmidtDown} \tag{1.5}$$

While this process will extract the geometry of the skull from the density data produced by the CT scan, it will also tend to extract a few aberrant points as skull. Typically other materials present in the CT scan cause these points, such as parts of the frame or the bed of the CT scanner. These aberrant points can be removed from the segmentation by disallowing all points that cannot be recursively reached in the 3D segmentation dataset from a known skull point. Since typically all transducer elements will be active in

treatment, the entire skull volume is segmented, and then the regions of interest for each transducer are extracted. This removes the redundancy in performing segmentation multiple times on overlapping regions. The extraction of skull data to a cylindrical mesh for a particular element of a 64-element hemispherical array is shown in Figure 1.5.

1.3 ACOUSTIC SIMULATION IN HETEROGENEOUS MATERIALS

To a reasonable approximation, human tissues such as liver, fat and brain may be considered to be homogenous to acoustic propagation. Many directories exist detailing the acoustic properties of these tissues in a variety of species, over a range of temperatures, both preserved and freshly harvested [12-15].

However, a clearly demonstrated in Section 1.2.3 and Figure 1.4, the density of the bones of the skull varies over a wide range. The speed of sound in bone is also known to be highly variable. An acoustic simulation method is required that can model propagation through inhomogeneous materials in which the density, speed of sound, and amplitude attenuation coefficient all have local variation. These circumstances are not suited to treatment with standard analytical methods.

1.3.1 NUMERICAL MODELING WITH PARTIAL DIFFERENTIAL EQUATIONS

The acoustic simulations in this thesis are predominantly based on an explicit finite-difference time-domain (FDTD) [16] formulation of the Westervelt equation [17]. This equation is similar to the classical equation for a wave, but it conserves certain higher-order terms that model the behavior of a non-ideal fluid. The form of the Westervelt equation for a non-linear, absorbing, inhomogeneous fluid is:

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3} + \frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \nabla p \cdot \nabla\left(\ln \rho\right) = 0 \qquad (1.6)$$

where p is pressure, c is the local speed of sound, δ is acoustic diffusivity, α is the coefficient of absorption [18], β is the coefficient of non-linearity [19], and ρ is the local density and ω is the angular frequency of the sources of excitation. Furthermore, there are also two useful substitutions to this equation. The first replaces δ, the acoustic diffusivity.

Since $\delta = 2\alpha c^3/\omega^2$ [20], for harmonic excitation $\frac{2\alpha}{c\omega^2}\frac{\partial^3 p}{\partial t^3} \equiv -\frac{2\alpha}{c}\frac{\partial p}{\partial t}$, where α is the acoustic amplitude absorption coefficient and ω is the angular frequency. The second substitution removes the spatial differential of the natural logarithm of ρ by noting that $d(\ln \rho)/dz \equiv \rho^{-1}.d\rho/dz$.

All volumetric acoustic simulations in this thesis are performed on a cylindrical mesh. This geometry was chosen for two reasons: it provides a greater density of data points around the axis of the mesh which is typically the region of greatest interest, and it segues well from previous acoustic research that historically has used cylindrical symmetry to study acoustic propagation from axisymmetric transducers in cylindrically symmetrical media [21].

Recognizing that $d(\ln \rho)/dz \equiv \rho^{-1}.d\rho/dz$, in cylindrical coordinates the Westervelt equation becomes:

$$\begin{aligned} &\frac{\partial^2 p}{\partial r^2}+\frac{1}{r}\frac{\partial p}{\partial r}+\frac{1}{r^2}\frac{\partial^2 p}{\partial \theta^2}+\frac{\partial^2 p}{\partial z^2}-\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}+\frac{\delta}{\rho c^4}\frac{\partial^3 p}{\partial t^3}+\ldots \\ &\frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2}+\left(\frac{\partial p}{\partial t}\right)^2\right]-\frac{1}{\rho}\left[\frac{d\rho}{dr}\frac{dp}{dr}+\frac{1}{r^2}\frac{d\rho}{d\theta}\frac{dp}{d\theta}+\frac{d\rho}{dz}\frac{dp}{dz}\right]=0 \end{aligned} \tag{1.7}$$

The numerical discretization of this function in cylindrical co-ordinates has some interesting cases in the region of the central axis. These issues are treated in detail in Chapter 2. In an explicit numerical formulation, the values of the data points at the next incremental step are calculated directly from previous or known values. An implicit formulation is one in which the values of the data points at some future time are iterated until a convergence is reached. Implicit solutions tend to find their niche in problems with straightfoward boundary conditions, such as Neumann or Dirichlet boundaries, solved using Fourier representations of the solution by a matrix inversion method. Explicit solution methods tend to be more unstable than implicit methods; implicit methods are rarely unstable, but they may converge to an incorrect answer or never

converge. One advantage in using an explicit solution in a medical application is that the solution will either be germane or it will be unstable and generate infinities; the difference between the two outcomes is clear. Contrariwise, an implicit solution may produce a result that is merely incorrect.

1.3.2 BOUNDARY CONDITIONS IN FDTD ACOUSTIC ANALYSIS

Since it is impossible to have a numerical simulation that is infinite in extent, all numerical simulations require boundary conditions to constrain the simulation volume to a region of interest. Ideally, any sound wave that reaches the boundary of the simulation volume should behave as if it were carrying on and out into an infinitely large fluid-filled space. This is equivalent to stating that we would like a perfectly absorbing boundary around the periphery of the volume so that no reflections occur at the edges. This is an interesting problem and still the subject of much intensive research in computational simulation. The classical boundary conditions of Neumann and Dirichlet set either a fixed value around the periphery of the simulation volume or a fixed first derivative. Both of these are entirely inappropriate to this situation.

The appropriate boundary condition in one dimension is the Robin boundary condition, also known as the acoustic half space condition. Taking the case of a boundary at z=0 where the simulation volume is in positive z, the Robin condition is:

$$\left(\frac{\partial}{\partial z}-\frac{1}{c}\frac{\partial}{\partial t}\right)p_{z=0}=0 \tag{1.8}$$

which is a simple factorization of the plane wave equation, stating that at $z = 0$ there is no wave traveling in the positive z direction. Despite the apparent simplicity of this function, the first stable implementation [22] of this was not achieved until 1981 by Mur [23].

However, this equation is only perfectly absorbing for waves whose direction of propagation is exactly normal to the boundary; waves with glancing incidences are poorly

absorbed. In fact, it is not possible to formulate a perfectly absorbing boundary condition in three dimensions that is local in time and space. It is only possible to produce better approximations to this boundary condition. The Mei and Fang [24] super-absorbing boundary tackles this problem with the following approach:

If the outgoing wave was perfectly absorbed at the boundary, then there was no incoming wave at $z = 0$ and hence there was no incoming wave at $z = \delta z$ either. However, for waves at non-normal incidence, this will not have been true, and there will be some error at $z = \delta z$.

$$\left(\frac{\partial}{\partial z} - \frac{1}{c}\frac{\partial}{\partial t}\right) p_{z=\delta z}^{n-1} = E \tag{1.9}$$

The absorbing boundary condition can be adjusted for this error at this time step, so that:

$$\begin{aligned} & \left(\frac{\partial}{\partial z} - \frac{1}{c}\frac{\partial}{\partial t}\right) p_{z=0}^{n} - E = 0 \\ \Rightarrow \quad & \left(\frac{\partial}{\partial z} - \frac{1}{c}\frac{\partial}{\partial t}\right) p_{z=0}^{n} - \left(\frac{\partial}{\partial z} - \frac{1}{c}\frac{\partial}{\partial t}\right) p_{z=\delta z}^{n-1} = 0 \end{aligned} \tag{1.10}$$

Discretizing this, we obtain:

$$\frac{p_{z=\delta z}^{n} - p_{z=0}^{n}}{\Delta z} - \frac{p_{z=2\delta z}^{n} - p_{z=\delta z}^{n}}{\Delta z} = \frac{1}{c}\left(\frac{p_{z=0}^{n+1} - p_{z=0}^{n}}{\Delta t} - \frac{p_{z=\delta z}^{n} - p_{z=\delta z}^{n-1}}{\Delta t}\right) \tag{1.11}$$

And expressing it in array co-ordinates rather than in terms of z:

$$p_{1,j,k}^{n+1} = p_{1,j,k}^{n} + p_{2,j,k}^{n} - p_{2}^{n-1} + \frac{c\Delta t}{\Delta z}\left(p_{2,j,k}^{n} - p_{1,j,k}^{n} - p_{3,j,k}^{n-1} + p_{2}^{n-1}\right) \tag{1.12}$$

Similar analysis yields absorbing boundary conditions for the upper absorbing layer in z and for the absorbing cylindrical wall at the limit of r.

1.3.3 THE NUMERICAL STABILITY OF FDTD METHODS

Let us consider the modeling of a simple one-dimensional wave and the conditions that stability requirements impose upon the design of a finite difference simulation. Taking the most straightforward case, the classic differential equation for a plane pressure wave in a linear, nonabsorbing, nonscattering fluid is:

$$\frac{\partial^2 p}{\partial z^2} = c^2 \frac{\partial^2 p}{\partial t^2} \tag{1.13}$$

For a numerical simulation of the behavior of this equation, our method must not allow any solutions that become infinite in time. This can be analyzed in terms of time and space eigenvalues. The time eigenvalue problem is:

$$\frac{\partial^2}{\partial t^2} p_i^n = \Lambda p_i^n \tag{1.14}$$

which we approximate numerically, using a second order method, as:

$$\frac{p_i^{n+1} - 2p_i^n + p_i^{n-1}}{\Delta t^2} = \Lambda p_i^n \tag{1.15}$$

If we now define a growth factor, q_i, where:

$$q_i = \frac{p_i^{n+1}}{p_i^n} = \frac{p_i^n}{p_i^{n-1}} \tag{1.16}$$

then q_i must have a magnitude less than or equal to unity for all points and for all modes of propagation. Thus:

$$\frac{q_i u_i^n - 2u_i^n + \left(u_i^n / q_i^n\right)}{\Delta t^2} = \Lambda u_i^n$$

$$\Rightarrow \quad q_i^2 - \left(2 + \Lambda.\Delta t^2\right) q_i + 1 = 0$$

$$\Rightarrow \quad q_i = \frac{2 + \Lambda.\Delta t^2}{2} \pm \sqrt{\left(\frac{2 + \Lambda.\Delta t^2}{2}\right) - 1}$$

From this, it is reasonably clear that the limit on q_i is satisfied for $-4/\Delta t^2 \le \Lambda \le 0$. Now, turning to the eigenvalues in space, we can similarly begin with:

$$c^2 \frac{\partial^2 p_i^n}{\partial x^2} = \Lambda p_i^n \tag{1.17}$$

and, by numerical approximation,

$$c^2 \left(\frac{p_{i+1}^n - 2p_i^n + p_{i-1}^n}{\Delta x^2} \right) = \Lambda p_i^n \tag{1.18}$$

Expressing this in terms of the spatial frequency content, we have:

$$c^2 p_0 \left(\frac{e^{jk(i+1)\Delta x} - 2e^{jki\Delta x} + e^{jk(i-1)\Delta x}}{\Delta x^2} \right) = \Lambda p_0 e^{jki\Delta x}$$

$$\Rightarrow \quad \frac{c^2}{\Delta x^2}\left(e^{jk\Delta x} - 2 + e^{jk\Delta x}\right) = \Lambda$$

$$\Rightarrow \quad \frac{2c^2}{\Delta x^2}\left(\cos(k\Delta x) - 1\right) = \Lambda$$

Given that the range of the cosine is ±1 regardless of the wave number k, we have $-4c^2/\Delta x^2 \le \Lambda \le 0$. This gives us an upper bound on Λ, but two different lower bounds. Now, of these, it is reasonably obvious that the true lower bound is $-4/\Delta t^2 \le \Lambda \le 0$ since for Λ lower than this q_i is greater than 1, which is guaranteed unstable. Therefore:

$$\frac{-4c^2}{\Delta x^2} \geq \frac{-4}{\Delta t^2}$$

$$\Rightarrow \quad \frac{c\Delta t}{\Delta x} \leq 1 \tag{1.19}$$

This simple expression is the Courant-Friedrichs-Levy stability requirement, and the CFL number is defined as $\text{CFL} = c\Delta t/\Delta x$. An equivalent statement is to say that the FDTD problem must be defined such that no pulse can propagate more than one space step through the mesh in one time step. The CFL condition therefore has a strong physical meaning, although without the foregoing explanation it is not straightforward to understand why this should be a constraint. $\text{CFL} \leq 1$ is a necessary but not sufficient condition for stability. In practice, most simulations are performed with $\text{CFL} \approx 0.5$.

1.3.4 Stability and Simulation Time in the Cylindrical Mesh

While the cylindrical mesh has its geometrical benefits as stated previously, it brings with it some special problems regarding the stability of the FDTD method and the time required to complete the simulation. At first glance it might appear that the simulation would run the risk of being unstable around the periphery of the volume where the points are widely spaced around the circumference and the spatial resolution is lowest. In fact, the converse is true; the problem is most unstable around the axis where the density of data points is greatest. From the previous section, it is clear that this is where the CFL number will be largest, and it is this region that sets the minimum time-step size for the simulation. However, it is possible to run different parts of the simulation mesh at time-steps that are integer multiples of this minimum time-step size to increase the efficiency of the simulation as a whole. The technique of using multiple time-steps is explored in detail in Chapter 2.

1.3.5 ACOUSTIC SIMULATION USING DISTRIBUTED COMPUTING METHODS

Rainfall is a distributed compute-server package that allows the simulation methods described previously to be distributed across a heterogeneous network of desktop computers, allowing all of them to contribute to the parallel solution of the problem [25]. Each computer on the network runs one instance of the Rainfall Server program, as shown in Figure 1.6. The server is a small program that resides unobtrusively in the system tray. It is designed to run as an idle process so that any spare processing cycles are used by the server application for simulations without impacting the performance of the machine from the local user's perspective. Rainfall is based upon the Remote Procedure Calling (RPC) mechanism [26]. Each Rainfall Server registers its existence with the local network name server and provides information about its protocol and network port number. Client applications can dynamically locate and communicate with all Rainfall Servers on the local network without having to be informed by the user which machines these servers are located on or what their network configuration might be. Simulation code can be compiled as a dynamically-linked library, allowing Rainfall servers to retrieve whatever routines are necessary to solve the problem at hand. This also allows distributed simulation code to be easily modified and updated for all Rainfall servers.

1.4 REFERENCES

[1] B. Quesson, J.A. de Zwart and C.T. Moonen, Magnetic resonance temperature imaging for guidance of thermotherapy, *Journal of Magnetic Resonance Imaging*, vol. 12, pp.525-533, 2000.

[2] N.B. Smith, N.K. Merrilees, M. Dahleh and K. Hynynen, Control system for an MRI compatible intracavitary ultrasound array for thermal treatment of prostate disease, *International Journal of Hyperthermia*, vol. 17, pp.271-282, 2001.

[3] K. Hynynen, O. Pomeroy, D.N. Smith, P.E. Huber, N. McDannold, J. Kettenbach, J. Baum, S. Singer and F.A. Jolesz, MR imaging-guided focused ultrasound surgery of fibroadenomas in the breast: a feasibility study, *Radiology*, vol. 219, pp. 176-185, 2001.

[4] J. Sun and K.H. Hynynen, The potential of transskull ultrasound therapy and surgery using the maximum available skull surface area *Journal of the Acoustical Society of America*, vol. 105, pp. 2519-2527, 1999.

[5] L. Laitinen, A. Bergenheim, and M. Hariz, Ventroposterolateral pallidotomy can abolish all Parkinsonian symptoms *Stereotact Funct Neurosurg*, vol. 58, pp. 14-21, 1992.

[6] P.E. Huber and P.Pfisterer, In vitro and in vivo transfection of plasmid DNA in the Dunning prostate tumor R3327-AT1 is enhanced by focused ultrasound, *Gene Therapy*, vol. 7, pp. 1516-1525, 2000.

[7] F. Fry, Transskull transmission of axisymmetric focused ultrasonic beams in the 0.5 to 1 MHz frequency range: implications for brain tissue visualization, interrogation, and therapy *Ultrasonic Tissue Characterization*, vol. 2, pp. 203-208, 1979.

[8] K. Hynynen and F.A. Jolesz, Demonstration of potential noninvasive ultrasound brain therapy through an intact skull *Ultrasound in Medicine and Biology*, vol. 24, pp. 275-283, 1998.

[9] K. Hynynen and J. Sun, Trans-skull ultrasound therapy: the feasibility of using image-derived skull thickness information to correct the phase distortion *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, pp. 752-755, 1999.

[10] J. Sun and K.H. Hynynen, Focusing of therapeutic ultrasound through a human skull: a numerical study *Journal of the Acoustical Society of America*, vol. 104, pp. 1705-1715, 1998.

[11] B.K.P. Horn, Closed-form solution of absolute orientation using unit quaternions *Journal of the Optical Society of America A*, vol. 4, pp. 629-642, 1986.

[12] F. Duck. Thermal properties of tissue. In: *Physical properties of tissue: a comprehensive reference book*, Academic Press, London, 1990.

[13] F. Duck. Acoustic properties of tissue at ultrasonic frequencies. In: *Physical properties of tissue: a comprehensive reference book*, Academic Press, London, 1990.

[14] S.A. Goss, R. Johnston, and F. Dunn, Comprehensive compilation of empirical ultrasonic properties of mammalian tissues *Journal of the Acoustical Society of America*, vol. 64, pp. 423-457, 1978.

[15] A.R. Selfridge, Approximate material properties in isotropic materials *IEEE Transactions on Sonics and Ultrasonics*, vol. SU-32, pp. 381-394, 1985.

[16] A. Taflove. *Computational Electromagnetics: The finite-difference time-domain method*, Artech House, 1995.

[17] P.J. Westervelt, Parametric Acoustic Array *Journal of the Acoustical Society of America*, vol. 35, pp. 535-537, 1963.

[18] D.T. Blackstock, Thermoviscous attenuation of plane, periodic, finite-amplitude sound waves *Journal of the Acoustical Society of America*, vol. 36, pp. 534-542, 1964.

[19] L. Bjorno and P.A. Lewin. Measurement of nonlinear acoustic parameters in tissue. In: *Tissue characterization with ultrasound, volume 1*, ed. J.F. Greenleaf. CRC Press, 1986.pp. 141-163.

[20] M.F. Hamilton and C.L. Morfey. Model Equations. In: *Nonlinear Acoustics*, eds. M.F. Hamilton and D.T. Blackstock. Academic Press, 1997.pp. 41-63.

[21] I.M. Hallaj and R.O. Cleveland, FDTD simulation of finite-amplitude pressure and temperature fields for biomedical ultrasound *Acoust.Res.Lett.Online*, vol. 1, pp. 7-12, 1999.

[22] O.M. Ramahi, Stability of absorbing boundary conditions *IEEE Transactions on Antennas and Propagation*, vol. 47, pp. 593-599, 1999.

[23] G. Mur, Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-23, pp. 377-382, 1981.

[24] K.K. Mei and J. Fang, Superabsorption - a method to improve absorbing boundary conditions *IEEE Transactions on Antennas and Propagation*, vol. 40, pp. 1001-1010, 1992.

[25] G.W. Sabot. *High performance computing - problem solving with parallel and vector architectures*, Addison-Wesley, 1995.

[26] J. Bloomer. *Power Programming with RPC*, O'Reilly & Associates, 1992.

1.5 FIGURES

Figure 1.1: A frame mounted onto a calvarium.
(a) Rostral aspect.
(b) Anterior aspect.
(c) Caudal aspect.

Figure 1.2: A CT slice of a calvarium mounted in a frame

Figure 1.3: The 64 element hemispherical array.
(a) Viewed from the side.
(b) Viewed looking onto active surface.

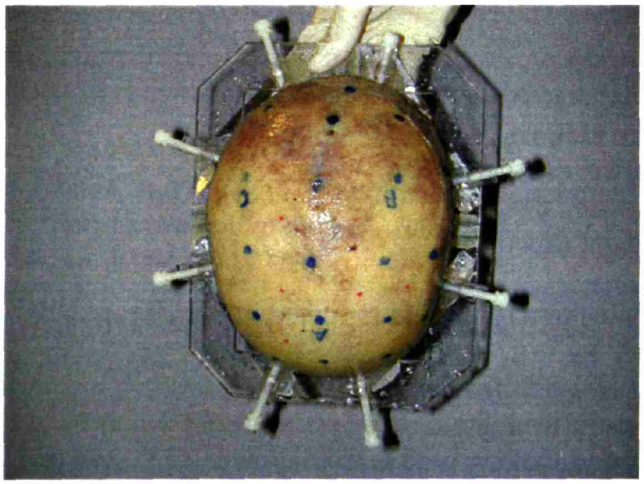
Figure 1.4: A typical one-dimenstional apparent density trace from the surface of the transducer through the skull to the focus point.

Figure 1.5: Extraction of the segmented skull to a cylindrical mesh.
(a) Rostral aspect.
(b) Caudal aspect.

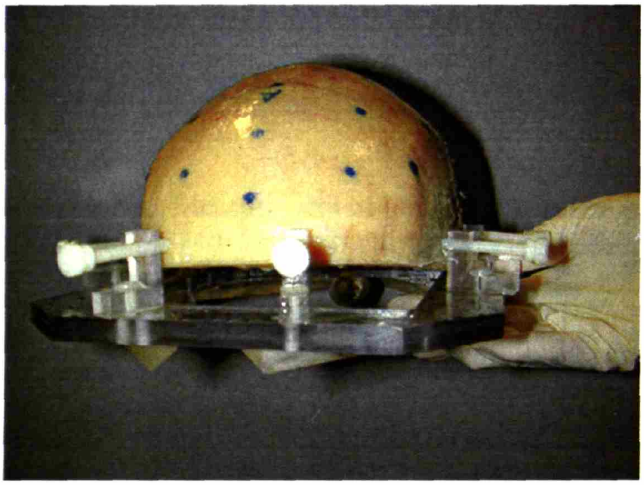
Figure 1.6: The Rainfall Server user interface.

Figure 1.1

(a)



(b)



(c)

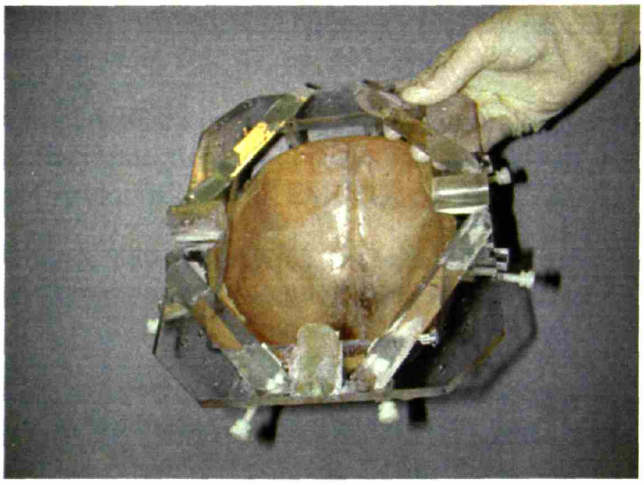


Figure 1.2

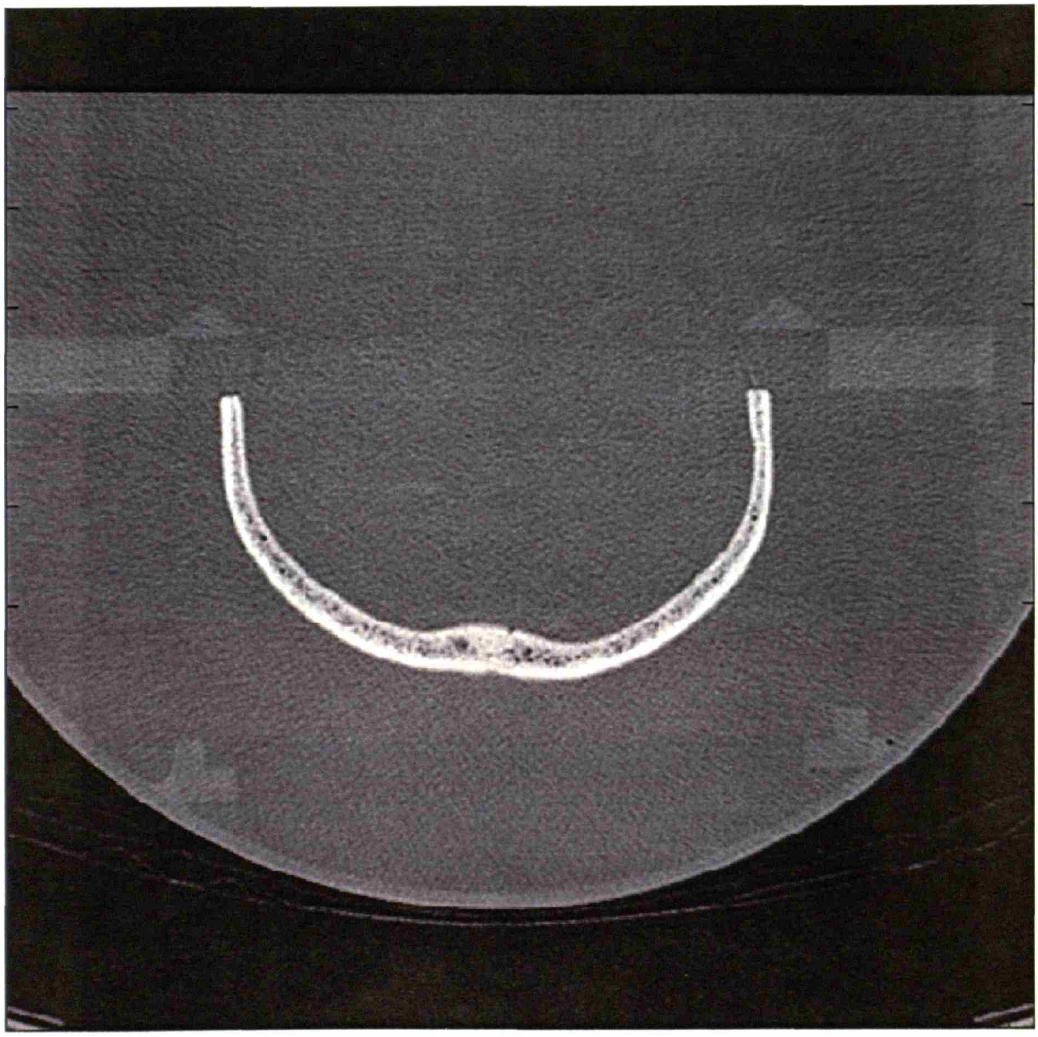
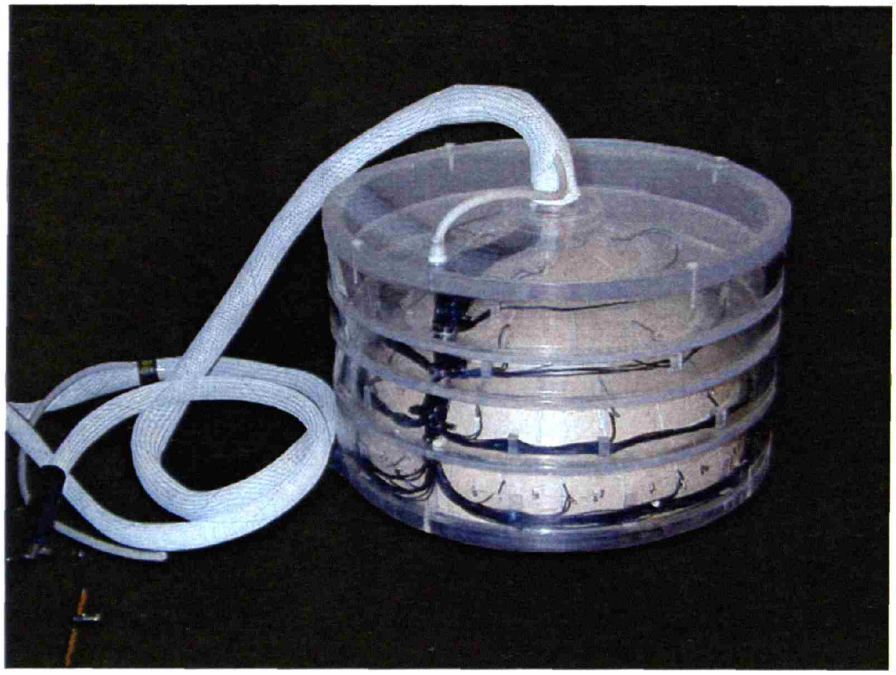


Figure 1.3

(a)



(b)



Figure 1.4

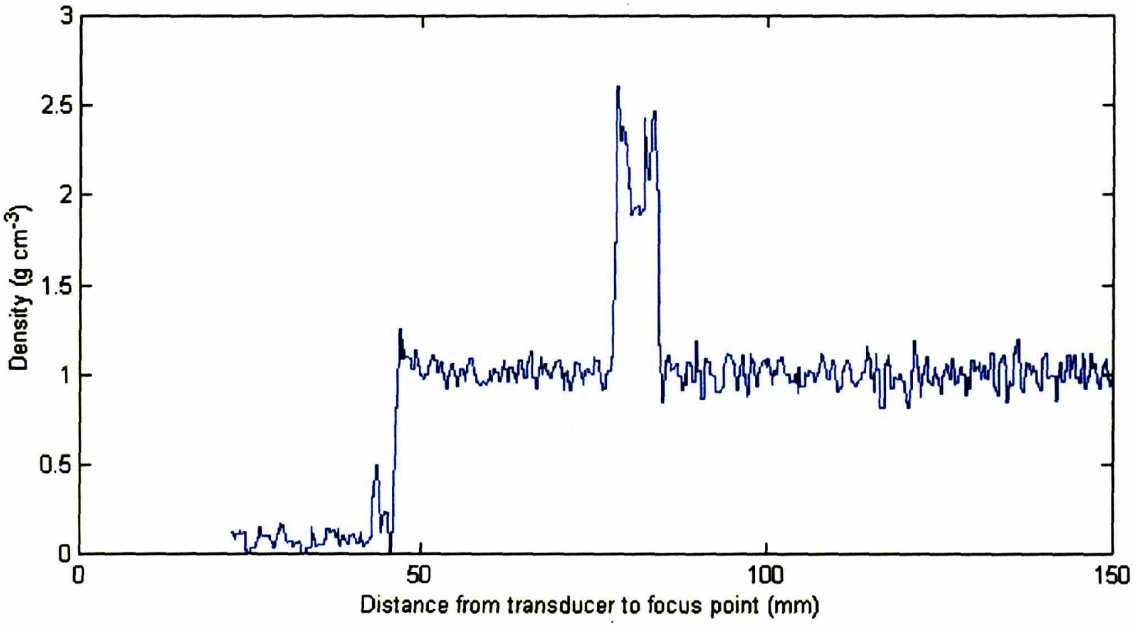
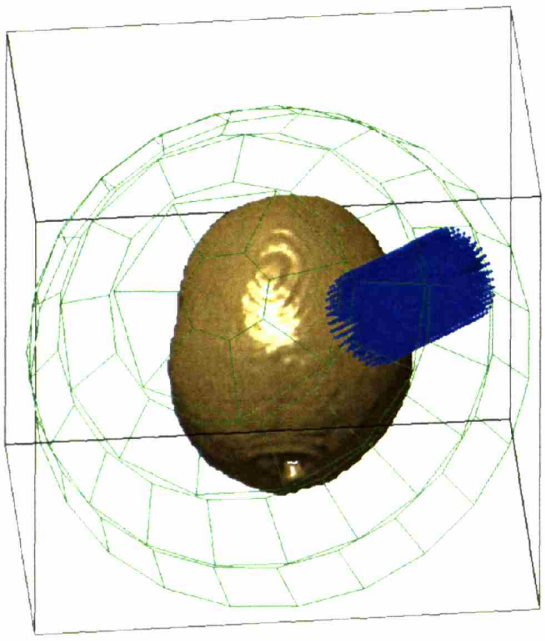

Density (g cm-3)
Distance from transducer to focus point (mm)
0
0.5
1
1.5
2
2.5
3
0
50
100
150

Figure 1.5

(a)



(b)

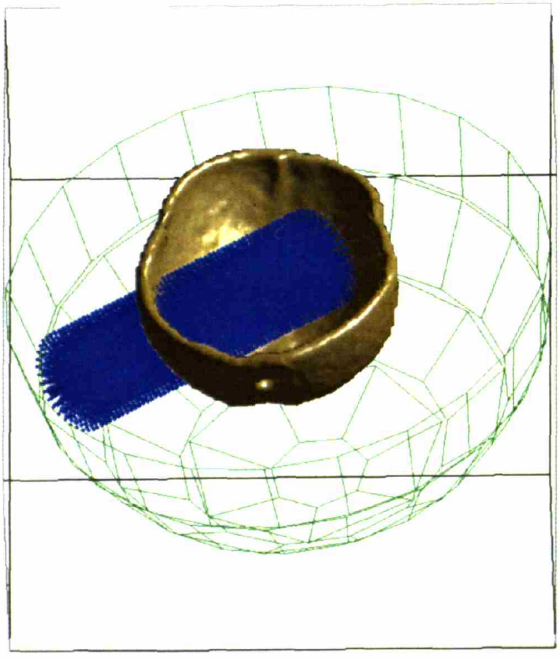
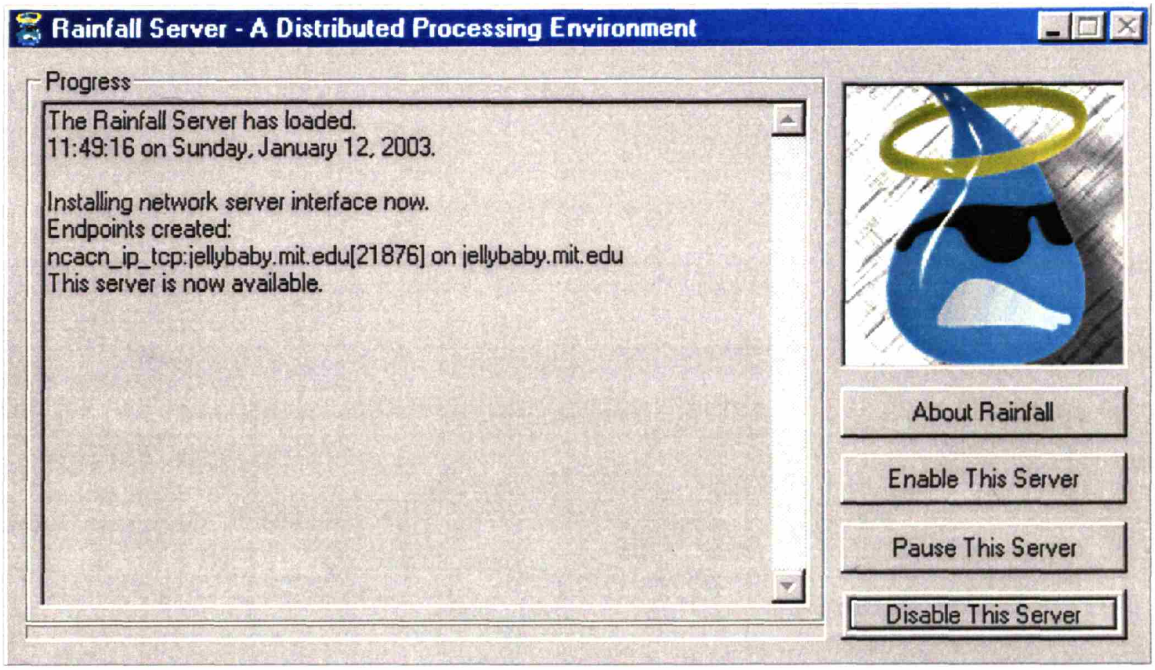


Figure 1.6


Rainfall Server - A Distributed Processing Environment
Progress
The Rainfall Server has loaded.
11:49:16 on Sunday, January 12, 2003.
Installing network server interface now.
Endpoints created:
ncacn_ip_tcp:jellybaby.mit.edu[21876] on jellybaby.mit.edu
This server is now available.
About Rainfall
Enable This Server
Pause This Server
Disable This Server

Chapter 2

Bio-Acoustic Thermal Lensing And Non-Linear Propagation In Focused Ultrasound Surgery Using Large Focal Spots: A Parametric Study.

The work presented in this chapter has been published as:

Connor and Hynynen, "Bio-Acoustic Thermal Lensing And Non-Linear Propagation In Focused Ultrasound Surgery Using Large Focal Spots: A Parametric Study", *Physics in Medicine and Biology* (47), pp. 1191-1128.

2.1 ABSTRACT

It is well known that the acoustic properties of soft tissue have a dependence on tissue temperature. This is of particular interest in focused ultrasound surgery since the mechanism of action of focused ultrasound surgery is to kill targeted tissue by inducing localized heating by ultrasound absorption, and hence cautery of that tissue. However, the act of localized heating induces a change in the acoustic properties of the targeted tissue and tissue surrounding it. This phenomenon distorts the incoming acoustic wavefront, and has been termed the thermal lens effect for this reason. Furthermore, non-linear effects in acoustic propagation become non-negligible at the ultrasound intensities required for therapeutic action.

This paper examines the importance of the thermal lens effect and non-linear tissue properties by simulating a variety of clinically applicable phased array transducer configurations that have not yet been appropriately analyzed using a full three-dimensional non-linear treatment of acoustic propagation. The significance of the thermal lens effect is characterized by comparing the simulation of coupled acoustic and thermal propagation with an uncoupled treatment; neglecting thermal lensing typically produces a movement of 1 to 2 millimeters in the predicted position of the focus towards the transducer.

The results also show that the classical methods of acoustic propagation can produce grossly erroneous results under certain clinically relevant transducer configurations and that an acoustic field scan with a hydrophone may not accurately predict therapeutic effect.

2.2 INTRODUCTION

Focused ultrasound provides a non-invasive method for inducing temperature elevations in tumors, resulting in the coagulation of the tumor [1] and apoptotic cell death [2]. This focused ultrasound surgery has been demonstrated experimentally [3], explained theoretically [4,5] and practiced clinically [6-10].

The design of focused ultrasound equipment and the planning of focused ultrasound treatments demand the ability to model focused ultrasound therapy and predict *in vivo* outcomes [11]. The phenomenon of thermal lensing [12-15] is particularly interesting here as it has the potential to cause the movement of the acoustic focus in otherwise entirely homogenous media; the local increase in temperature at the focus causes a change in local tissue properties and leads to an acoustically inhomogeneous medium. Furthermore, at the acoustic intensities used for focused ultrasound surgery, the infinitesimal wave approximation ceases to be valid and non-linear material properties become apparent. Classical acoustic methods are ill suited to solving this problem; a complete treatment of this behavior requires inhomogeneous equations to quantify the time-evolution of the acoustic and thermal fields, and a mechanism of coupling these fields.

Previous studies have examined solely axisymmetric tissue volumes and excitations using linear acoustic behavior [16] and non-linear behavior [17,18]. Hallaj *et al.* [15] studied the effects of including or neglecting thermal lensing calculations in acoustic simulations. The largest discrepancy the authors found in the position of the acoustic focus due to thermal lensing was about 2 mm. However, single focused transducers suitable only for small tumor ablation were investigated. To treat large tumors, phased array applicators capable of producing multiple focal spots simultaneously are needed [19-21]. These arrays use high power and thus thermal lensing and nonlinear propagation effects may be different from the single focused transducer case studied so far.

This paper extends the simulations into a fully three-dimensional approach allowing arbitrary transducer designs and tissue volumes to be analyzed. This model is used to study the role of nonlinear propagation and thermal lensing in high-intensity focused ultrasound treatment using phased arrays with large focal spots.

2.3 METHODS

The acoustic model in this paper is based on the one used in [15,22], but extended into 3D. It incorporates inhomogeneity, absorption and non-linearity and is coordinated with Pennes Bioheat [23] equation to model the effects of the ultrasound on the temperature field, and consequently the effects of temperature on the propagation of ultrasound. Tissue coagulation with thermal dose is modeled by the Sapareto and Dewey function[24]. The models of variation of tissue parameters with temperature used in the acoustic time-evolution simulations are based on published experimental studies. In addition to the previously mentioned *in vivo* uses of these models, they have also been verified in phantom studies [25,26].

2.3.1 CONFIGURATION OF SIMULATIONS FOR THIS PARAMETRIC STUDY

This study examines the effect of varying three parameters in the design of a sector vortex focused ultrasound transducer [27,28]. These are:

1. The f-number of the transducer, which is the ratio of the spherical radius of curvature of the transducer to the aperture of the device. This study covers f-number values from 0.8 to 1.4, suitable for a range of shallow and deeply penetrating focused ultrasound devices.
2. The mode of excitation. Each sector of a sector vortex array may be individually driven, and the mode represents the relationship between the phases of the periodic waves driving the sectors. The mode is the number of complete rotations in the phase shift over the sectors. See Table 2.1 for the relative phases of the

elements of an 8 element sector vortex array with respect to the mode of excitation. Note that Mode 0 is the axisymmetric excitation case, and this study covers all modes.

3. The frequency of excitation of the array. This parametric study examines excitation at 0.667 MHz, 1.0 MHz and 1.2 MHz, suitable for deep sonications into soft tissues such as liver or breast.

The study is centered on the case of 1.0 MHz excitation in Mode 1 for a transducer with f-number of 1.0, and simulations are performed for all single-parameter variations as show in Table 2.2.

The sector vortex transducer is simulated to be in water with a gap of 5mm between the rim of the transducer and the tissue. Each simulation uses the same simplified tissue model; the tissue comprises 2cm of fat, below which is liver, as shown in Figure 2.1. The transducers are driven such that a peak temperature of 80°C is produced within 10 seconds, requiring a peak pressure at the transducer surface of between 0.17 MPa and 0.45 MPa depending on the parametric configuration of the simulation. In Section 2.4.2, comparison is made between thermal fields resulting from simulations based on non-linear acoustic propagation and from simulations based on linear methods; interesting distinct focal regions arise in these simulations that are more clearly seen when a peak temperature of 70°C is shown. Figure 8 and 9 are shown with this slightly lower peak temperature to allow clearer comparisons to be made between the linear and non-linear cases.

2.3.2 METHODS OF SIMULATION

The propagation of ultrasound into tissue is modeled using the Westervelt equation[29].

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3} + \frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \nabla p \cdot \nabla(\ln \rho) = 0 \qquad (2.1)$$

where p is pressure, c is the speed of sound, δ is acoustic diffusivity, β is the coefficient of non-linearity and ρ is the density. The simulation is performed on a cylindrical mesh.

This geometry provides a greater density of data points around the axis of the mesh which is typically the region of greatest interest; this provides an advantage over a uniform Cartesian mesh structure [30]. A cylindrical mesh also segues well from previous acoustic research that historically has used polar formulations to study acoustic propagation from axisymmetric transducers in cylindrically symmetrical media.

Recognizing that $d(\ln \rho)/dz \equiv \rho^{-1}.d\rho/dz$, in cylindrical coordinates, the Westervelt equation becomes:

$$\frac{\partial^2 p}{\partial r^2}+\frac{1}{r}\frac{\partial p}{\partial r}+\frac{1}{r^2}\frac{\partial^2 p}{\partial \theta^2}+\frac{\partial^2 p}{\partial z^2}-\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}+\frac{\delta}{\rho c^4}\frac{\partial^3 p}{\partial t^3}+\ldots$$
$$\frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2}+\left(\frac{\partial p}{\partial t}\right)^2\right]-\frac{1}{\rho}\left[\frac{d\rho}{dr}\frac{dp}{dr}+\frac{1}{r^2}\frac{d\rho}{d\theta}\frac{dp}{d\theta}+\frac{d\rho}{dz}\frac{dp}{dz}\right]=0 \qquad (2.2)$$

Since $\delta = 2\alpha c^3/\omega^2$, for harmonic excitation $\frac{2\alpha}{c\omega^2}\frac{\partial^3 p}{\partial t^3} \equiv -\frac{2\alpha}{c}\frac{\partial p}{\partial t}$, where α is the acoustic amplitude absorption coefficient and ω is the angular frequency.

Temperature distribution throughout the tissue under sonication is modeled using the Bioheat Equation [23]:

$$\rho_t C_t \frac{\partial T}{\partial t} = k_t \nabla^2 T - \rho_b C_b W (T - T_a) + Q \qquad (2.3)$$

where C is specific heat capacity, k is thermal conductivity and W is the perfusion rate. This is similarly expressed in cylindrical polar co-ordinates. The variable Q varies spatially and represents the acoustic energy absorbed by the tissue at a given point, and is given by[31]:

$$Q = \frac{1}{n\tau}\int_{t'}^{t'+n\tau} \frac{\delta}{\rho c^4}.\left(\frac{\partial p}{\partial t}\right)^2 dt \qquad (2.4)$$

where n is a positive integer, τ is the period of the ultrasound excitation, and t' is sufficient time from the beginning of the simulation such that a steady state has been reached. δ, ρ and c are as defined above.

The equation models thermal diffusion with the $k_t \nabla^2 T$ term, where k_t is the thermal conductivity of the tissue. Removal of heat by blood circulation is modeled by the $\rho_b C_b W (T - T_a)$ term, where ρ_b and C_b are the density and specific heat capacity of blood respectively and W is the volumetric perfusion rate of the tissue measured in milliliters of blood per milliliter of tissue per second. T_a is the ambient body temperature, which is 37°C. Typical values for material properties for thermal simulations are shown in Table 2.3, where ρ, c, α_0, β, k, W, and C are as defined above. These values are for human tissue at a frequency of 1MHz. For frequencies of excitation greater or lesser than 1MHz, linear scaling of a with frequency is used such that $a = a_0 f$, where f is in MHz [32].

For thermal lensing calculations, the constant values of c and α described in Table 2.3 are replaced with temperature-varying functions. The variation of speed of sound in water with temperature is as described by Bilaniuk [33], and the variation in speed of sound and acoustic amplitude absorption coefficient is as described in Hallaj *et al.* [15]. The speed of sound data and absorption data used by Hallaj *et al.* are based in part on empirical data in human liver tissue and bovine peritoneal fat [34] and also informed by canine tissue models [35]. The measurement technique used in the original literature is estimated to have an error range of $\pm 3.45 \times 10^{-2}$ Np, which based on tissue sample thicknesses of 3cm and bidirectional passage of ultrasound through these samples gives an estimated error of ± 0.58Np m^{-1}.

$$c_{water} = 1402.39 + 5.0371T - 5.8085\times10^{-2}T^2 + 3.3420\times10^{-4}T^3 - 1.4780\times10^{-6}T^4 + 3.1464\times10^{-9}T^5$$
$$c_{fat} = 1746.1 - 3.9308T - 4.1282\times10^{-1}T^2 + 1.1373\times10^{-2}T^3 - 1.1010\times10^{-4}T^4 + 3.7308\times10^{-7}T^5$$
$$c_{liver} = 1529.3 + 1.6856T + 6.1131\times10^{-2}T^2 - 2.2967\times10^{-3}T^3 + 2.2657\times10^{-5}T^4 - 7.1795\times10^{-8}T^5$$

$$\alpha_{fat} = 40.5 - 2.1054T + 4.505\times10^{-2}T^2 - 0.3880\times10^{-4}T^3 + 1.2238\times10^{-6}T^4$$
$$\alpha_{liver} = 5.5367 - 2.9950\times10^{-1}T + 3.3357\times10^{-2}T^2 - 1.6058\times10^{-3}T^3 + 3.4382\times10^{-5}T^4 \ldots$$
$$-3.2486\times10^{-7}T^5 + 1.1181\times10^{-9}T^6$$

These α values are also specified at a frequency of 1MHz, and linear scaling is likewise introduced. T is the temperature in centigrade. These functions are stated to have interpolatable validity over the range of 30°C to 90°C; in this paper their use is restricted to the range of 37°C to 80°C. These functions are shown in Figure 2.2 and Figure 2.3.

The Finite Difference Time Domain (FDTD) pressure and temperature simulations are coupled by simulating the acoustic propagation until a continuous wave case is reached, at which point the values of Q throughout the volume are calculated from the pressure data. This Q field is used to simulate heating with a duration of one second, after which time the resulting temperature data is used to recalculate the material properties of the sonicated tissue using the functions above. These new material properties are used to recalculate the acoustic propagation until a new continuous wave is reached. This iterative method allows thermal lensing calculations to be performed for long durations of sonication. This coupling method is based on the fact that the rise in tissue temperature with time is many orders of magnitude slower than the time constants involved in simulating acoustic propagation. To perform a simulation in which thermal lensing effects are neglected, clearly all one need do is neglect to modify the tissue properties as temperature changes.[15]

2.3.2.1 The cylindrical mesh hides a central singularity.

Previous works on the axisymmetric case have defined the mesh in such a way that one border of it is coincident with the axis of symmetry as shown in Figure 2.4a [15]. The second term of the Westervelt equation in cylindrical coordinates depends on the reciprocal of r, which clearly gives rise to a singularity at the axis where $r = 0$. However, this singularity disappears in an axisymmetric case - the first differential with respect to r is guaranteed to be zero at $r = 0$ as the pressure field is necessarily a local minimum or maximum with respect to r. Other terms in the Westervelt equation that are dependent on the reciprocal of r^2 are also dependent on the derivatives of certain quantities with respect to θ. In the axisymmetric case, all derivatives with respect to θ have a value of zero everywhere by definition so these terms also pose no problem. This is fortuitous, but it

does not apply when p is allowed to vary arbitrarily with r, z, and θ. There is no reason why the first differential of pressure with r should be zero at the axis in a non-symmetric case, and in general it is not.

This problem can be avoided by redefining the structure of the mesh. Let [I,J,K] be the co-ordinate system of our mesh, running from [1,1,1] to [I_{max}, J_{max}, K_{max}], where the I co-ordinate lies along the z axis, the J co-ordinate represents the radial distance r from the cylindrical axis, and K represents an angle of rotation θ about the cylindrical axis. The singularity is avoided if the mapping from the coordinate J to the radius r is allowed to be $r = (J - 0.5).\Delta r$ instead of the more traditional $r = (J - 1).\Delta r$. This produces the mesh shown in Figure 2.4b.

The advantage here is that now no points are coincident upon each other at the axis, and that the distance between two neighboring points on any particular 'spoke' is conserved as Δr.

2.3.2.2 Stability and Simulation Time in the Cylindrical Mesh

While the cylindrical mesh has its geometrical benefits, as stated previously, it brings with it some special problems regarding the stability of the FDTD method and the time required to complete the simulation. The simulation is most unstable around the axis where the density of data points is greatest, and it is therefore this region that will set the time-step size for the simulation.

A well known necessary condition for the stability of FDTD solutions is that the Courant-Friedrichs-Levy (CFL) number be less than or equal to unity. The CFL number is defined as $CFL = c\Delta t/\Delta x$ where c is the local wave propagation speed. Δt is the time step of the FDTD simulation, and Δx is the distance separating adjacent grid points. In practice, most simulations are performed with CFL ≈ 0.5 and Δx set such that there are around 8 mesh points per wavelength. Therefore for a frequency of 1MHz, setting $\Delta x = 1.875 \times 10^{-4}$ and $\Delta t = 5 \times 10^{-8}$ s would be appropriate.

Let us consider a standard geometry for a typical problem. If the sources of excitation in the simulation volume are sinusoidal with a frequency of 1 MHz, and the volume is full of water ($c \approx 1500\ ms^{-1}$), then we might set the point density in the mesh to be 10 points per spatial wavelength in both the z and r directions with 48 spokes around the axis of rotation. If we then set the time-step to give us a CFL number of 0.5 in the z or r direction, we obtain:

$$\Delta t = \frac{\mathrm{CFL}.\Delta z}{c} = \frac{0.5 \times \frac{(1500/1\times10^6)}{10}}{1500} = 50\,\mathrm{ns}$$

Now, at the points closest around the axis, we find that:

$$\mathrm{CFL} = \frac{c\Delta t}{0.5\Delta r \Delta\theta} = \frac{1500\times50\times10^{-9}}{0.5\times\frac{(1500/1\times10^6)}{10}\times\frac{2\pi}{48}} = 7.64$$

To obtain a stable simulation, we shall have to reduce the time-step down to 5 ns, giving a CFL of 0.76 around the axis. However, this increases the time to complete the simulation by an order of magnitude. Furthermore, it is easily shown that it is only in the innermost n rings, where

$$n = \mathrm{floor}\left((\Delta\theta)^{-1} + 0.5\right) \tag{2.5}$$

that this instability arises; here n is 8. Floor is the rounding-down function such that floor(3.4) = 3. Thus, for a simulation that may contain several hundred radial points, the simulation time is increased by an order of magnitude to provide stability to a small fraction of the simulation volume.

A solution to this problem is *gearing*. The essence of the solution is that different parts of the volumetric mesh should be iterated forward with different time steps appropriate to the stability requirements local to that region. The current implementation of this divides the simulation into two volumes, called the inner and outer regions, and defines two

further parameters. The *gearRadius* is the outermost concentric ring contained in the inner volume, and the *gearRatio* is defined as:

$$gearRatio = \frac{\Delta t_{outer}}{\Delta t_{inner}} \tag{2.6}$$

The simulation begins with all points in the volume at the same simulation time t. First, the values of the outer points are calculated one time step ahead, at a future time of $t + \Delta t_{outer}$. Next, the inner points are repeatedly stepped forward in increments of Δt_{inner}, requiring *gearRatio* steps until both the outer and inner volumes represent the same simulation time. For a fourth-order spatial FDTD method, the difficulty arises in that points in the *gearRadius* and *gearRadius* – 1 ring require values at points in the outer volume at times between t and $t + \Delta t_{outer}$. We obtain this information by interpolating into the outer volume between these two times. Thus it is important that the outer volume be stepped forward first so that the future-time values are available. The interpolation in time is cubic and calculated directly by Lagrange's formula. Thus, in terms of ε, where ε is a fractional step between p^n and p^{n+1}, we obtain:

$$p^{n+\varepsilon} = \frac{1}{6}\left[\varepsilon(\varepsilon+1)(\varepsilon+2)p^{n+1} - 3(\varepsilon-1)(\varepsilon+1)(\varepsilon+2)p^{n} + 3\varepsilon(\varepsilon-1)(\varepsilon+2)p^{n-1} - \varepsilon(\varepsilon-1)(\varepsilon+1)p^{n-2}\right]$$

This allows the centermost *gearRadius* points to be run using the fine increment Δt_{inner} for stability, while using the larger step size for the outer volume. This brings with it an improvement in simulation time of almost an order of magnitude. The penalty for this is the additional complexity in the code design to allow for different step sizes in time in the mesh, and in maintaining the coherency of the data in the volume, especially when the simulation is being run as a distributed calculation across a network of computers.

2.3.2.3 Modeling tissue death and coagulation

Tissue death is the outcome of thermal treatment, and the ultimate goal of the treatment simulation is to determine the pattern and extent of this tissue death. The effect of a given temperature profile on tissue can be determined using Sapareto and Dewey's function [24]. This function maps a time course of thermal exposure to an equivalent duration of exposure at 43°C, as follows:

$$T_{43} = \int_0^t R^{T_{ref}-T(t)} dt \qquad R = \left\{ \begin{matrix} 0.5, T \geq T_{ref} \\ 0.25, T \leq T_{ref} \end{matrix} ; T_{ref} = 43^{\circ}C \right\} \tag{2.7}$$

where T(t) is the temperature at a particular point in the tissue at time t.

The conservative limit for tissue death is passed when T_{43} is greater than 240 minutes. This value is tracked throughout the simulation since once tissue death has occurred, that portion of tissue is considered to be coagulated and its volumetric blood perfusion rate W is set to zero in the Pennes Bioheat equation; this approach is applicable to the tissue model we have defined for this study. However, hyperthermia treatments targeted to areas close to large blood vessels may require adaptation to model the local tissue thermal response accurately [36].

Tissue coagulation margins are drawn using contours of the Sapareto and Dewey function: the inner contour corresponds to an equivalent exposure of 240 minutes at 43°C within which all tissue has died, the outer contour corresponds to an equivalent exposure of 30 minutes at 43°C outside of which no tissue has been damaged.

2.4 RESULTS

2.4.1 THERMAL LENSING EFFECTS ARE ILLUSTRATED BY CASES 1 THROUGH 8

Figure 2.5b shows the thermal field in the liver under mode 1 sonication at 1 MHz at the point at which the thermally-lensed simulation reached a peak temperature of 80°C, which is the central case of the parametric study. The left half-plane shows the simulation performed without thermal lensing and the right half-plane shows the simulation with thermal lensing. The black contour lines show the predicted tissue coagulation margins caused by this treatment as described in 2.3.2.3 above. The arrangement and thickness of the tissue layers is as described in 2.3.1 above and as illustrated in Figure 2.1; this is equivalent in orientation to the transducer being placed on the abdomen of a supine patient. The vertical axis shows the vertical distance from the back of the transducer in centimeters and so increases downwards. The horizontal axis shows the distance in centimeters from the central axis of rotation of the sector vortex transducer.

Figure 2.5a, b and c show the results of mode 1 sonication from a transducer with focal number of 1.0 at 1 MHz, 0.666 MHz and 1.2 MHz respectively. In each case, neglecting to include thermal lensing effects causes the focus to appear 1.4mm closer to the transducer. Neglecting thermal lensing produces no noticeable change in the width of the focus. As the frequency of the sonication increases the size of the focal spot becomes smaller, as is clearly the case with constructive interference.

Figure 2.6a, b, c and d show the thermal field in liver under mode 1 sonication at 1 MHz from transducers with focal numbers 0.8, 1.0, 1.2 and 1.4 respectively. As the focal number increases, it is clear that the focal spot becomes longer regardless of thermal lensing effects. More importantly, as the focal number increases, the discrepancy between the simulations neglecting and including thermal lensing also increases; these discrepancies are shown in Table 2.4.

Figure 2.7a, b and c show the thermal field in the liver under sonication at 1MHz from a transducer with focal number 1.0 in modes 0, 1 and 2 respectively. The mode 4 case is substantially more complex and the results from this case are given in 2.4.2 below. Clearly across these cases, the focus becomes broader as the mode number is increased. In each case the discrepancy between neglecting and including thermal lensing in the position of the focus is 1.4mm.

2.4.2 NON-LINEAR PROPAGATION EFFECTS ARE ILLUSTRATED BY CASE 9

The results for mode 4 sonication at 1MHz using a transducer with focal number 1.0 are considered separately here due to a remarkable difference in the acoustic and temperature fields in the liver caused by the neglecting or implementing of non-linear acoustic propagation in the simulation. Classical methods for calculating acoustic fields assume linear behavior: the acoustic pressure field and thermal field generated under this assumption are shown in Figure 2.8a and b respectively. The relative locations of the sector vortex array and soft tissues are still as in Figure 2.1. Once again, the black contour lines in the thermal simulation delimit margins of predicted tissue coagulation. Comparison with Figure 2.7a, b and c here shows that these field patterns would fit the trend established by the mode 0, 1 and 2 simulations at 1 MHz and focal number 1.0.

However, once the non-linear properties of the tissues under sonication are considered, the acoustic and thermal fields take on a strikingly different form as shown in Figure 2.9a and b. The absorption of the acoustic field produces the thermal field structure as shown in cross-section in which the central lobe contains the temperature peak. As stated earlier, for clarity the temperature fields in Figure 8b and 9b are shown at the point at which a peak temperature of 70°C is produced with non-linear propagation. At this time the peak temperature produced with linear propagation reaches 58°C. The resulting tissue death margins are notable for the fact that in the linear propagation study there is no tissue death in the center of the region of heating. The non-linear study shows substantial tissue death at the center; the margins here indicate a region of coagulation 7.9mm wide.

2.5 Discussion and Conclusions

2.5.1 On Thermal Lensing

Measuring the movement of the thermal focus reveals that neglecting thermal lensing considerations produces a predicted thermal focus between 1mm and 2mm nearer the transducer. The positioning error is related to the focal number of the transducer; the effect is largely independent of mode and frequency. Table 2.4 shows that as the focal number of the transducer is increased, the discrepancy in the simulated position of the focus between simulations without thermal lensing and those with thermal lensing increases. This result is in agreement with Hallaj *et al.* [15], which commented that the effect of thermal lensing appeared to increase as focal number was increased.

The effect of thermal lensing produces an increase in peak temperature at the focus due to the increase in absorption coefficient in the tissue with temperature. However, thermal dosage calculations revealed that this did not affect the width of the region of tissue coagulation. This is due to the slowness of thermal diffusion in these soft tissues; consequently the elevated tissue temperature at the focus does not significantly increase the tissue temperature at the coagulation margins and so those margins are not noticeably broadened.

Thermal lensing occurs because of the cumulative effect along the acoustic propagation path of changes in tissue properties due to heating. It is therefore the heating in the prefocal region that controls where the focus appears. Hallaj *et al.* studied small transducers with apertures of 4cm, which resulted in an acoustic focus being formed very close to the surface of the body. The simulations in this paper are based upon the larger 10cm aperture configuration, which is necessary for deeper tissue treatment. Consequently Hallaj *et al.* found greater prefocal heating in subcutaneous fat than was noted in these simulations. With reference to the speed of sound curves in Figure 2.2, it is clear that the effect of prefocal heating in subcutaneous fat is to decrease the speed of

sound in fat and that the predominant effect of prefocal heating in liver is to increase the speed of sound in liver. As detailed above, all simulations in this study produced the result that neglecting thermal lensing resulted in the focus appearing closer to the transducer. Hallaj *et al.* found that neglecting thermal lensing would produce either no noticeable change or cause the focus to appear further away from the transducer. These results are nicely predicted by using the ray approximation to acoustic propagation and considering the refraction effect of the prefocal heating.

Thermal lensing calculations are therefore required if the accuracy of the placement of the focal spot is within the discrepancies given above. However, this is unlikely to be the case for most clinical applications; the displacement of the liver over a respiratory cycle is 10.8mm on average [37] and tissue surrounding the hepatic vein has a peak velocity of 4.2mm/s over the cardiac cycle [38]. Constant tissue property models are therefore likely to be sufficient especially when online temperature monitoring is used [6].

2.5.2 On non-linear propagation effects

The mode 4 case presents a very interesting heating pattern in the liver. The acoustic field produced by the distribution of phases in mode 4 excitation would be expected to result in field cancellation along the central axis. Nevertheless strong acoustic absorption occurs at the geometric focus of the sector vortex array leading to tissue coagulation. This is an unusual result and is not predicted by classical acoustic simulation methods.

The reason for the presence of this central focus becomes apparent on analyzing the acoustic pressure field due to this mode 4 excitation of the sector vortex transducer. The majority of the pressure field is composed of the expected ultrasound waves of frequency f=1 MHz. However, around the site of the central focus, we find the field contains only ultrasound with frequency $2f$. The original frequency f field induces its own second harmonic $2f$ due to the non-linear properties of the liver. Furthermore, the mode 4 field in frequency f produces a mode 0 field in $2f$. This is because the phase offsets for the mode

4 excitation are [0,180,0,180,0,180,0,180]; for a frequency of *2f* these phase shifts appear as [0,360,0,360,0,360,0,360] which is obviously mode 0.

Therefore, at a point where we would predict no heating due to the expected field cancellation of the mode 4 field, and hence no coagulation at all, we in fact find strong heating from the induced second harmonic field. Simulation methods that do not model the non-linear properties of the tissue will fail to predict the presence of this locus of heating.

It should be noted that we do not find substantial heating due to nonlinear effects in the mode 1 and mode 2 sonications. In these cases, the second harmonic field is induced in mode 2 and mode 4 respectively, so there is no constructive interference of this second harmonic at the geometric focus. Figure 2.10 shows the thermal fields produced by non-linear and linear acoustic propagation respectively; these fields are only subtly different with a slightly elongated focal spot as the second harmonic does not appear as an axisymmetric mode and so is not appreciably amplified against the fundamental. In the mode 0 case, there is constructive interference of the second harmonic at the geometric focus but this effect is largely obscured by the constructive interference of ultrasound at the fundamental frequency in the same location, so the pattern of tissue death is not remarkably modified.

Since the induction of the second harmonic field is a cumulative path effect dependent on the nonlinearity of the tissues in the prefocal region, it is predicted that nonlinear heating will be increased with focal number. Furthermore, as frequency is increased, non-linear heating will also increase due both to increased induction of the second harmonic field and also due to the increased absorption of higher frequency ultrasound in soft tissues.

Since the *2f* field is produced by non-linear effects, it is also clear that the heating pattern at therapeutic intensities cannot be accurately predicted by scanning the acoustic field experimentally at lower intensities with a hydrophone. At low intensities, the non-linear effects are not sufficiently strong to induce the second harmonic field and so this central

peak in pressure cannot be appreciated. This phenomenon will have an impact on the heating fields generated with other phased array geometries if phase rotation is used. It may also complicate the optimization [19,39,40] and control [41] of multifocus patterns used for therapy.

It may be possible to purposely enhance this nonlinear contribution to enhance focal heating by using low duty cycle high-pressure amplitude pulses. This was proposed earlier for enhancement of ultrasound hyperthermia [42,43] and may have value for ultrasound surgery also. Using a discrete Fourier technique in the time domain, we can separate the field shown in Figure 2.9a into the portions of the field contributed by ultrasound at frequencies *f* and *2f* respectively. This result is shown in Figure 2.11. This technique of generating a sharp *2f* focal spot with the phase rotation method may be useful for harmonic ultrasound imaging [44].

2.6 References

[1] J. Overgaard, Effect of hyperthermia on malignant cells in vivo. A review and a hypothesis. *Cancer*, vol. 39, pp. 2637-2646, 1977.

[2] N.I. Vykhodtseva, N. McDannold, H. Martin, R.T. Bronson, and K. Hynynen, Apoptosis in ultrasound-produced threshold lesions in the rabbit brain *Ultrasound in Medicine and Biology*, vol. 27, pp. 111-117, 2001.

[3] F. Fry and L. Johnson, Tumor irradiation with intense ultrasound *Ultrasound in Medicine and Biology*, vol. 4, pp. 337-411, 1978.

[4] T.C. Robinson and P.P. Lele, An analysis of lesion development in the brain and plastics by high-intensity focused ultrasound at low-megahertz frequencies. *Journal of the Acoustical Society of America*, vol. 51, pp. 1333-1351, 1972.

[5] P.P. Lele and A.D. Pierce, The thermal hypothesis of the mechanism of ultrasonic focal destruction in organised tissues. Interaction of ultrasound and biological tissues. FDA 73-8008 BRH/DBE, pp. 121-128, 1973. Bureau of Radiological Health. Washington, DC.

[6] K. Hynynen, O. Pomeroy, D.N. Smith, P.E. Huber, N. McDannold, J. Kettenbach, J. Baum, S. Singer, and F.A. Jolesz, MR imaging-guided focused ultrasound surgery of fibroadenomas in the breast: a feasibility study. *Radiology*, vol. 219, pp. 176-185, 2001.

[7] N.T. Sanghvi, F. Fry, R. Bihrle, F.S. Foster, M.H. Phillips, J. Syrus, A.V. Zaitsev, and C.W. Hennige, Noninvasive surgery of prostate tissue by high-intensity focused ultrasound. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 43, pp. 1099-1110, 1996.

[8] A.G. Visioli, I.H. Rivens, G.R. ter Haar, A. Horwich, R.A. Huddart, E. Moskovic, A. Padhani, and J. Glees, Preliminary results of a phase I dose escalation clinical trial using focused ultrasound in the treatment of localised tumours. *European Journal of Ultrasound* , vol. 9, pp. 11-18, 1999.

[9] G. Vallancien, E. Chartier-Kastler, N. Bataille, D. Chopin, M. Harouni, and J. Bougaran, Focused extracorporeal pyrotherapy *European Urology*, vol. 23, pp. 48-52, 1993.

[10] A. Gelet, J.Y. Chapelon, R. Bouvrier, O. Rouviere, Y. Lasne, D. Lyonnet, and J.M. Dubernard, Transrectal high-intensity focused ultrasound: minimally invasive

therapy of localized prostate cancer *Journal of Endourology*, vol. 14, pp. 519-528, 2000.

[11] K. Mahoney, T. Fjield, N. McDannold, G.T. Clement, and K.H. Hynynen, Comparison of modelled and observed in vivo temperature elevations induced by focused ultrasound: implications for treatment planning *Phys Med Biol*, vol. 46, pp. 1785-1798, 2001.

[12] R. Seip and E.S. Ebbini, Noninvasive estimation of tissue temperature response to heating fields using diagnostic ultrasound. *IEEE Transactions on Biomedical Engineering*, vol. 42, pp. 828-839, 1995.

[13] N.A. Watkin, G.A. ter Haar, and I.H. Rivens, The intensity dependence of the sute of maximal energy deposition in focused ultrasound surgery *Ultrasound in Medicine and Biology*, vol. 22, pp. 483-491, 1996.

[14] C. Le Floch, M. Tanter, and M. Fink, Self-defocusing in ultrasonic hyperthermia: Experiment and simulation *Applied Physics Letters*, vol. 74, pp. 3062-3064, 1999.

[15] I.M. Hallaj, R.O. Cleveland, and K. Hynynen, Simulations of the thermo-acoustic lens effect during focused ultrasound surgery *Journal of the Acoustical Society of America*, vol. 109, pp. 2245-2253, 2001.

[16] P.M. Meaney, R.L. Clarke, G.A. ter Haar, and I.H. Rivens, A 3D finite element model for computation of temperature profiles and regions of thermal damage during focused ultrasound surgery exposures *Ultrasound in Medicine and Biology*, vol. 24, pp. 1489-1499, 1998.

[17] W. Swindell, A theoretical study of non-linear effects with focused ultrasound in tissues: an acoustic 'Bragg' peak *Ultrasound in Medicine and Biology*, vol. 11, pp. 121-129, 1985.

[18] V.A. Khokhlova and L.A. Crum, Non-linear enhancement and saturation phenomena in high intensity focused ultrasound beams Proceedings of the 17th International Congress on Acoustics, pp. September 2-7, 2001, 2001.

[19] D.R. Daum and K. Hynynen, A 256-element ultrasound phased array for the treatment of large volumes of deep seated tissue *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, pp. 1254-1268, 1999.

[20] X. Fan and K. Hynynen, A study of various parameters of spherically curved phased arrays for noninvasive ultrasound surgery. *Phys Med Biol*, vol. 41, pp. 591-608, 1996.

[21] H. Wan, P. Van Baren, E.S. Ebbini, and C.A. Cain, Ultrasound Surgery: Comparison of strategies using phased array systems. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 43, pp. 1085-1098, 1996.

[22] I.M. Hallaj and R.O. Cleveland, FDTD simulation of finite-amplitude pressure and temperature fields for biomedical ultrasound *Acoust.Res.Lett.Online*, vol. 1, pp. 7-12, 1999.

[23] H.H. Pennes, Analysis of tissue and arterial blood temperatures in the resting human forearm *Journal of Applied Physiology*, vol. 1, pp. 93-122, 1948.

[24] S. Sapareto and W. Dewey, Thermal dose determination in cancer therapy *Int J Radiation Oncology Biol.Phys.*, vol. 10, pp. 787-800, 1984.

[25] J. Wu, J. Chase, and T. Holzapfel, Temperature rise in a tissue-mimicking material generated by unfocused and focused ultrasonic transducers. *Ultrasound in Medicine and Biology*, vol. 5, pp. 495-512, 1992.

[26] M.C. Kolios, A.E. Worthington, M.D. Sherar, and J.W. Hunt, Experimental evaluation of two simple thermal models using transient temperature analysis. *Phys Med Biol*, vol. 43, pp. 3325-3340, 1998.

[27] C.A. Cain and S. Umemura, Concentric-ring and sector-vortex phased array applicators for ultrasound hyperthermia *IEEE Transactions on microwave theory and techniques*, vol. MTT-34, pp. 542-551, 1986.

[28] T. Fjield and K. Hynynen, The Combined Concentric-Ring and Sector-Vortex Phased Array for MRI Guided Ultrasound Surgery *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 44, pp. 1157-1167, 1997.

[29] P.J. Westervelt, Parametric Acoustic Array *Journal of the Acoustical Society of America*, vol. 35, pp. 535-537, 1963.

[30] P.P. Delsanto, R.S. Schechter, and R.B. Mignogna, Connection machine simulation of ultrasonic wave propagation in materials III: The three-dimensional case *Wave Motion*, vol. 26, pp. 329-339, 1997.

[31] A.D. Pierce. *Acoustics: an introduction to its physical principles and applications*, 1989.

[32] F. Duck. Acoustic properties of tissue at ultrasonic frequencies. London: Academic Press, 1990.

[33] N. Bilaniuk and G.S.K. Wong, Speed of sound in pure water as a function of temperature *Journal of the Acoustical Society of America*, vol. 93, pp. 1609-1612, 2001.

[34] J.C. Bamber and C.R. Hill, Ultrasonic attenuation and propagation speed in mammalian tissues as a function of temperature *Ultrasound in Medicine and Biology*, vol. 5, pp. 149-157, 1979.

[35] C.A. Damianou, N.T. Sanghvi, F. Fry, and R. Maass-Moreno, Dependence of ultrasonic attenuation and absorption in dog soft tissues on temperature and thermal dose *Journal of the Acoustical Society of America*, vol. 102, pp. 628-634, 1997.

[36] M.C. Kolios, M.D. Sherar, and J.W. Hunt, Large blood vessel cooling in heated tissues: a numerical study *Phys Med Biol*, vol. 40, pp. 477-494, 1995.

[37] S.C. Davies, A.L. Hill, R.B. Holmes, M. Halliwell, and P.C. Jackson, Ultrasound quantitation of respiratory organ motion in the upper abdomen *British Journal of Radiology*, vol. 67, pp. 1096-1102, 1994.

[38] M. Schlaikjer, S. Torp-Pedersen, J.A. Jensen, and P.F. Stetson, Tissue motion in blood velocity estimation and its simulation IEEE Ultrasonics Symposium Proceedings, October 1998, 1998.

[39] E.S. Ebbini and C.A. Cain, Optimization of the intensity gain of multiple-focus phased-array heating patterns. *International Journal of Hyperthermia*, vol. 7, pp. 953-973, 1991.

[40] E.S. Ebbini and C.A. Cain, Multiple-focus ultrasound phased-array pattern synthesis: Optimal driving-signal distributions for hyperthermia. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 36, pp. 540-548, 1989.

[41] P. Van Baren and E.S. Ebbini, Multipoint temperature control during hyperthermia treatments: theory and simulation. *IEEE Transactions on Biomedical Engineering*, vol. 42, pp. 828-839, 1995.

[42] K. Hynynen, Demonstration of enhanced temperature elevation due to nonlinear propagation of focussed ultrasound in dog's thigh in vivo. *Ultrasound in Medicine and Biology*, vol. 13, pp. 85-91, 1987.

[43] K. Hynynen, The role of nonlinear ultrasound propagation during hyperthermia treatments *Med Phys*, vol. 18, pp. 1156-1163, 1991.

[44] B. Ward, A.C. Baker, and V.F. Humphrey, Nonlinear propagation applied to the improvement of resolution in diagnostic medical ultrasound. *Journal of the Acoustical Society of America*, vol. 101, pp. 143-154, 1997.

[45] F. Duck. Thermal properties of tissue. In: *Physical properties of tissue: a comprehensive reference book*, Academic Press, London, 1990.

[46] L. Bjorno and P.A. Lewin. Measurement of nonlinear acoustic parameters in tissue. In: *Tissue characterization with ultrasound, volume 1*, ed. J.F. Greenleaf. CRC Press, 1986.pp. 141-163.

[47] H.F. Bowman, Heat transfer and thermal dosimetry *Journal of Microwave Power*, vol. 16, pp. 121-133, 1981.

2.7 TABLES

Table 2.1: The phase shifts for each element with respect to mode for an 8 element sector vortex array.

Table 2.2: The parametric configuration of the simulations in this study.

Table 2.3: Typical material properties of relevant human tissues [32,45-47].

Table 2.4: Neglecting thermal lensing introduces a discrepancy in the position of the acoustic focus. This effect is most dependent of the focal number of the transducer.

Table 2.1

Element	Phase Shift of the Driving Waveform (degrees)			
	Mode 0	Mode 1	Mode 2	Mode 4
0	0	0	0	0
1	0	45	90	180
2	0	90	180	0
3	0	135	270	180
4	0	180	0	0
5	0	225	90	180
6	0	270	180	0
7	0	315	270	180

Table 2.2

Simulation	F-number	Mode	Frequency (MHz)
1	1.0	1	1.0
2	1.0	1	0.667
3	1.0	1	1.2
4	0.8	1	1.0
5	1.2	1	1.0
6	1.4	1	1.0
7	1.0	0	1.0
8	1.0	2	1.0
9	1.0	4	1.0

Table 2.3

	ρ [kg m^{-3}]	c [ms^{-1}]	α_0 [Np m^{-1} MHz^{-1}]	β [kg^3 m^{-4} s^{-2}]	k [W m^{-1} K^{-1}]	C [J kg^{-1} K^{-1}]	W [ml ml^{-1} s^{-1}]
Water	1000	1500	$2.88\text{x}10^{-4}$	3.5	0.615	4180	
Fat	910	1430	9	10.5	0.248	2490	$5\text{x}10^{-4}$
Liver	1050	1596	4.5	6	0.512	3600	$1.33\text{x}10^{-2}$
Blood	1030					3620	

Table 2.4

F-number	Thermal Lensing Discrepancy
0.8	1.1mm
1.0	1.4mm
1.2	1.7mm
1.4	2.1mm

2.8 FIGURES

Figure 2.1: Arrangement of tissues and the sector vortex array.

Figure 2.2: Temperature dependence of sound speed in soft tissue.

Figure 2.3: Temperature dependence of absorption in soft tissue.

Figure 2.4: Grid structures for Finite Difference Time Domain (FDTD) simulation.

Figure 2.5: Thermal fields under sonication at 666kHz, 1MHz and 1.2MHz
(a) Mode 1, 666 kHz, f-number 1.0
(b) Mode 1, 1MHz, f-number 1.0
(c) Mode 1, 1.2MHz, f-number 1.0

Figure 2.6: Thermal field under sonication from transducers with focal numbers 0.8, 1.0, 1.2, 1.4
(a) Mode 1, 1MHz, f-number 0.8
(b) Mode 1, 1MHz, f-number 1.0
(c) Mode 1, 1MHz, f-number 1.2
(d) Mode 1, 1MHz, f-number 1.4

Figure 2.7: Thermal fields under mode 0, mode 1 and mode 2 sonication.
(a) Mode 0, 1MHz, f-number 1.0
(b) Mode 1, 1MHz, f-number 1.0
(c) Mode 2, 1MHz, f-number 1.0

Figure 2.8: Simulation of Mode 4 sonication at 1MHz with classical linear theory.
(a) Pressure Field Produced By Classical Methods
(b) Temperature Field By Classical Methods

Figure 2.9: Simulation of Mode 4 sonication at 1MHz with nonlinear theory.
(a) Pressure Field Produced By Non-Linear Methods
(b) Non-Linear Temperature Field By Non-Linear Methods

Figure 2.10: Thermal field produced by non-linear acoustic propagation and linear acoustic propagation without thermal lensing under sonication from a transducer with focal number 1.0 at 1 MHz in mode 1.

Figure 2.11: Frequency separation of the Mode 4 field produced at 1MHz with nonlinear theory.
(a) Fundamental component at 1 MHz
(b) Harmonic component at 2 MHz

Figure 2.1

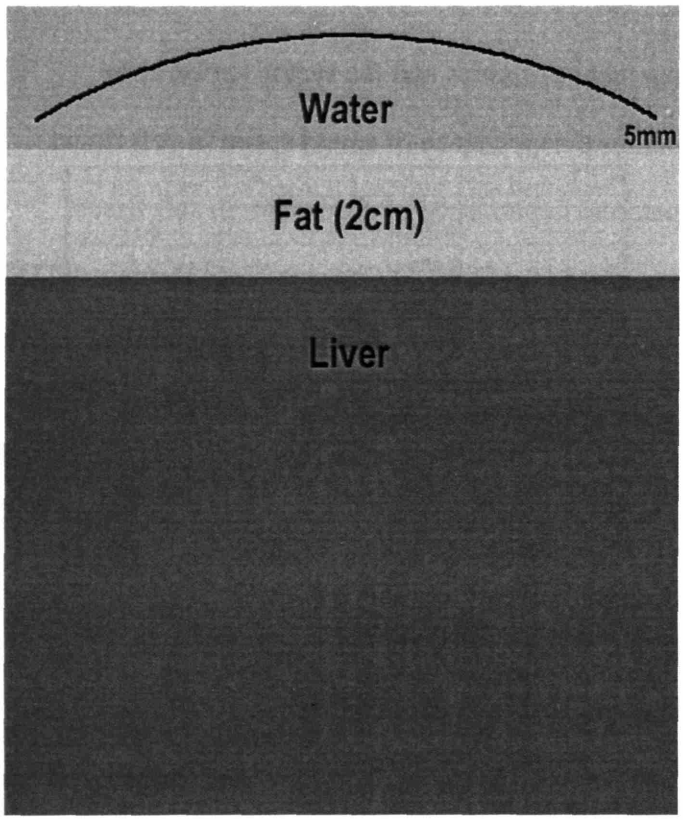


Figure 2.2

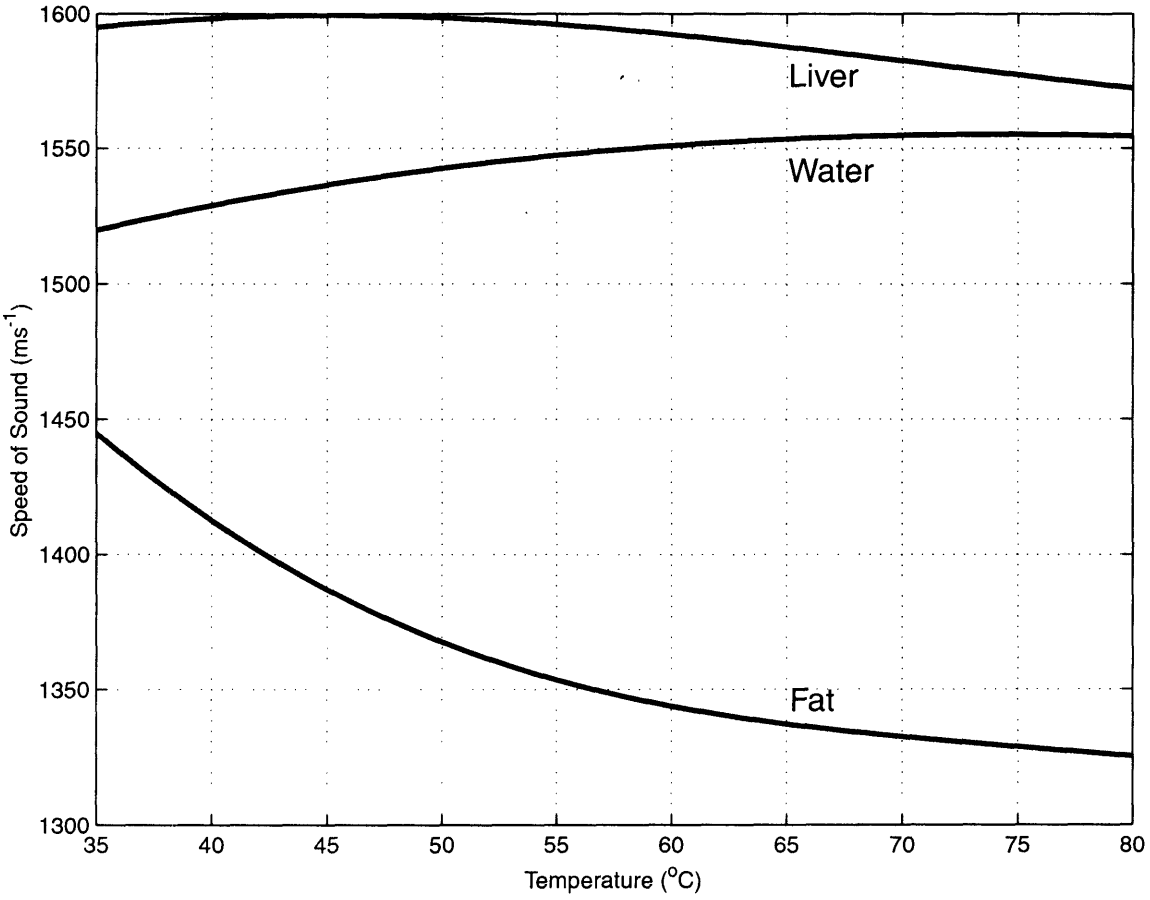

1600
1550
1500
1450
1400
1350
1300
35
40
45
50
55
60
65
70
75
80
Liver
Water
Fat
Speed of Sound (ms-1)
Temperature (°C)

Figure 2.3

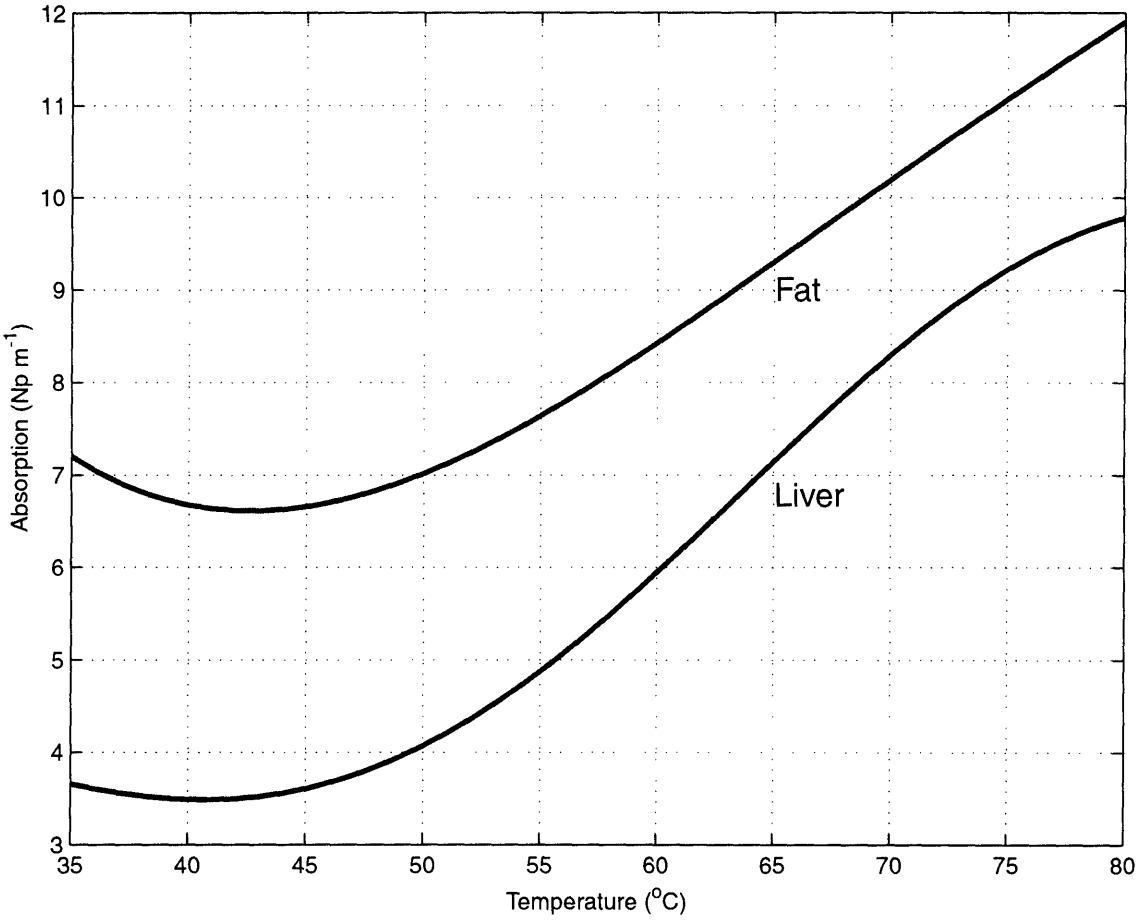
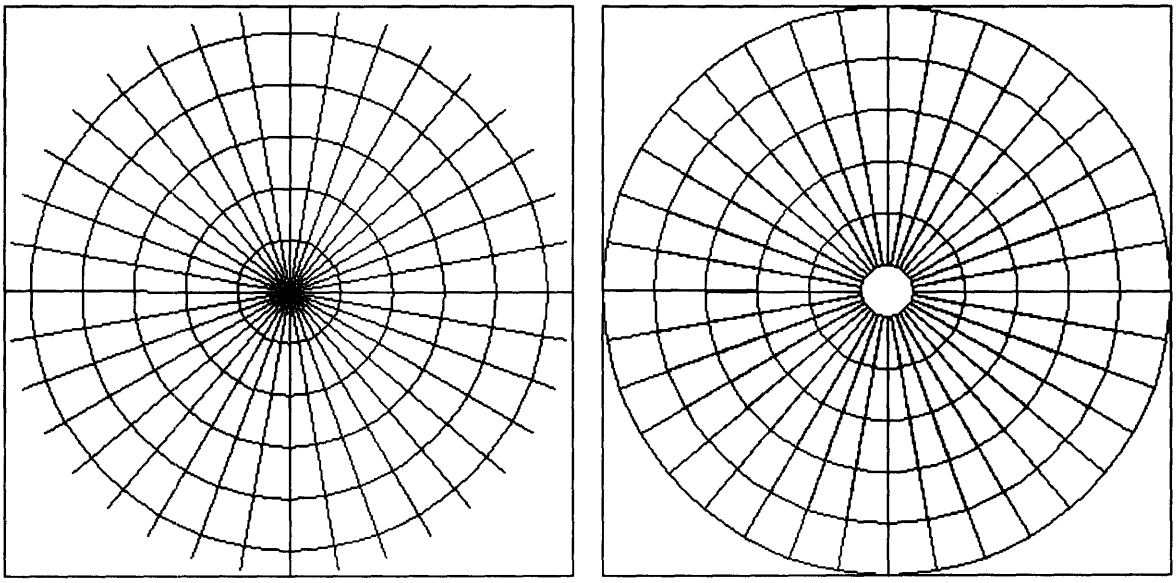

12
11
10
9
8
7
6
5
4
3
35
40
45
50
55
60
65
70
75
80
Absorption (Np m^{-1})
Temperature (^{o}C)
Fat
Liver

Figure 2.4



(a) The classical cylindrical mesh

(b) The FDTD3D mesh, without singularity

Figure 2.5

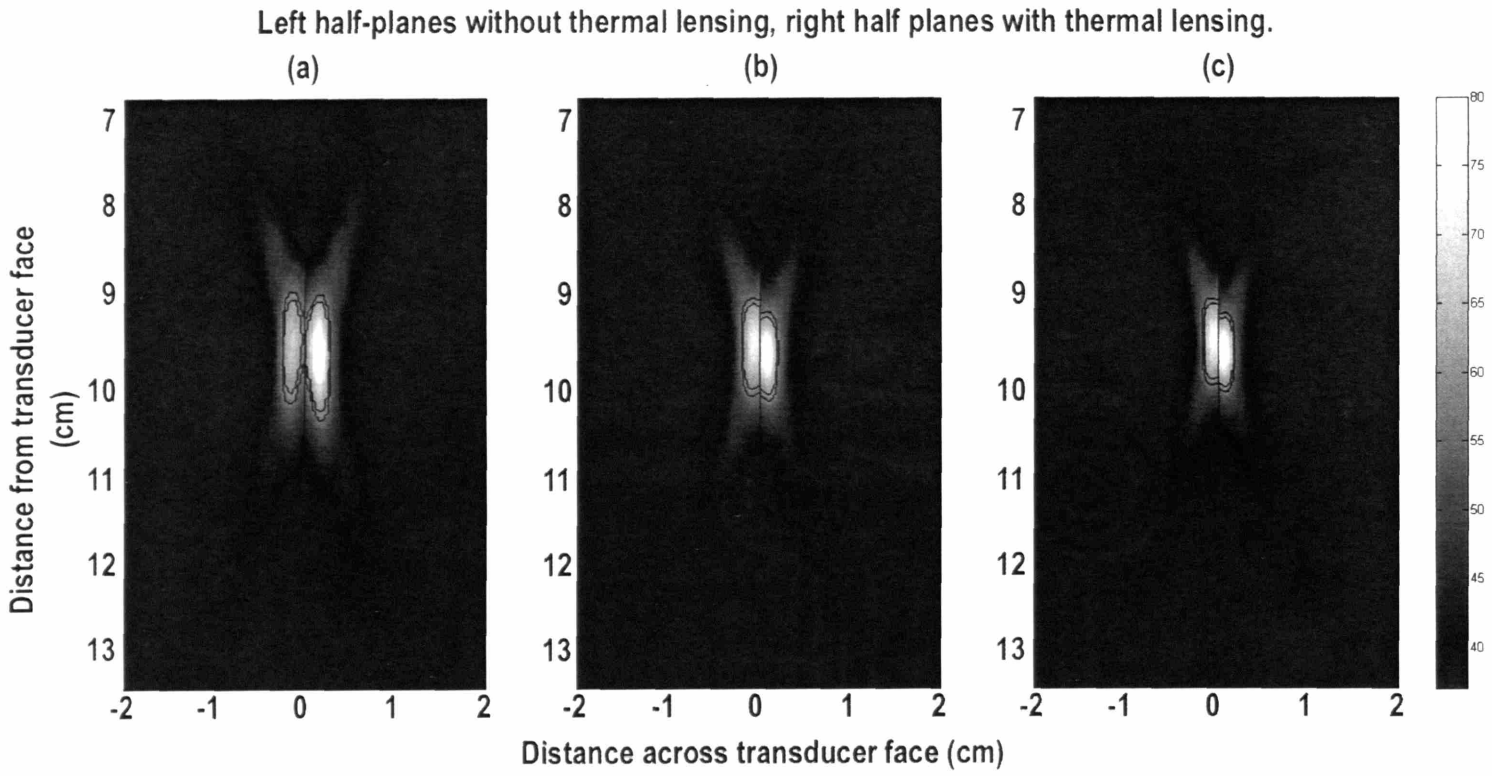

Left half-planes without thermal lensing, right half planes with thermal lensing.
(a)
(b)
(c)
Distance from transducer face (cm)
7
8
9
10
11
12
13
-2
-1
0
1
2
Distance across transducer face (cm)
80
75
70
65
60
55
50
45
40

Figure 2.6

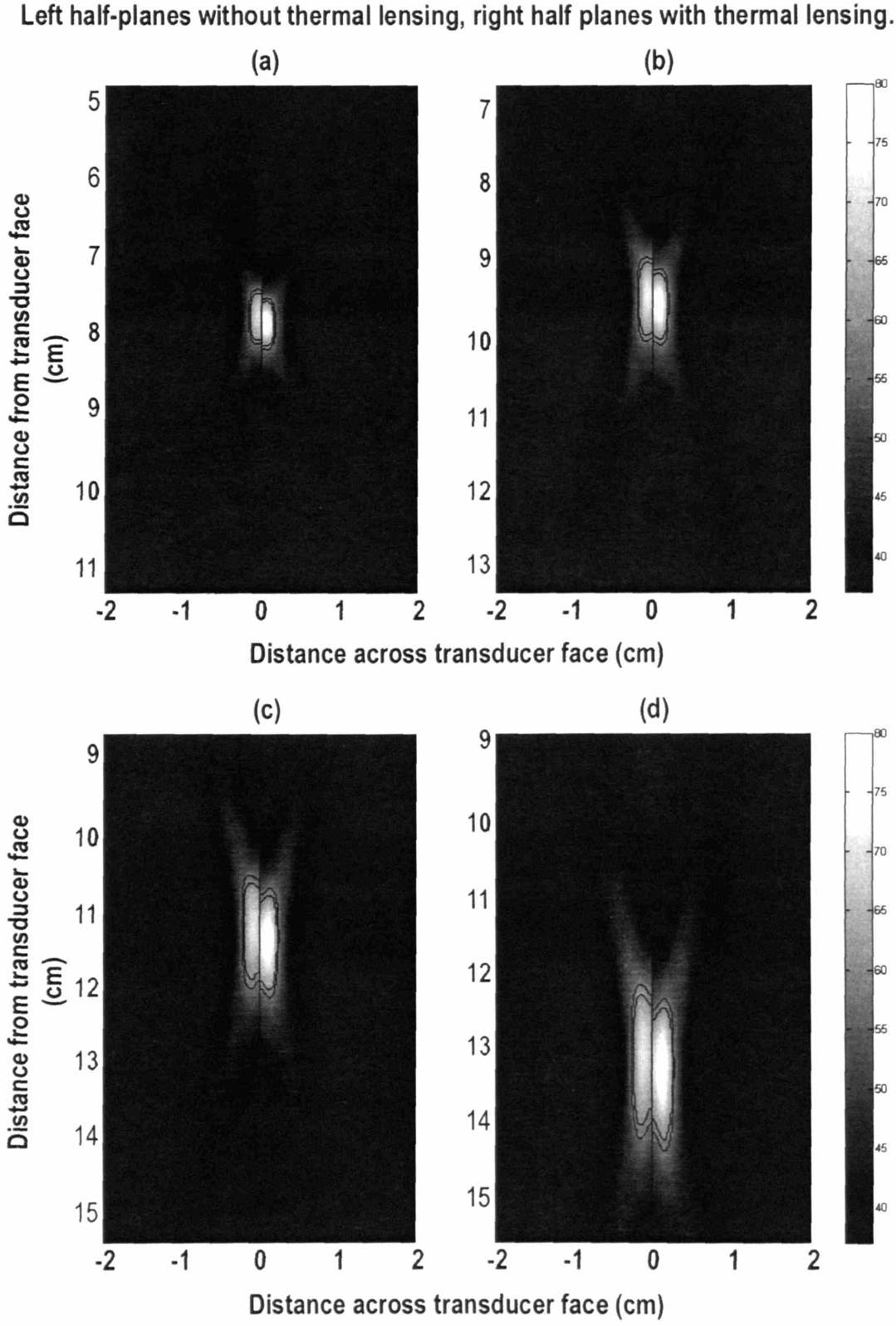

Left half-planes without thermal lensing, right half planes with thermal lensing.
(a)
(b)
(c)
(d)
Distance from transducer face (cm)
Distance across transducer face (cm)
5
6
7
8
9
10
11
12
13
14
15
-2
-1
0
1
2
80
75
70
65
60
55
50
45
40

Figure 2.7

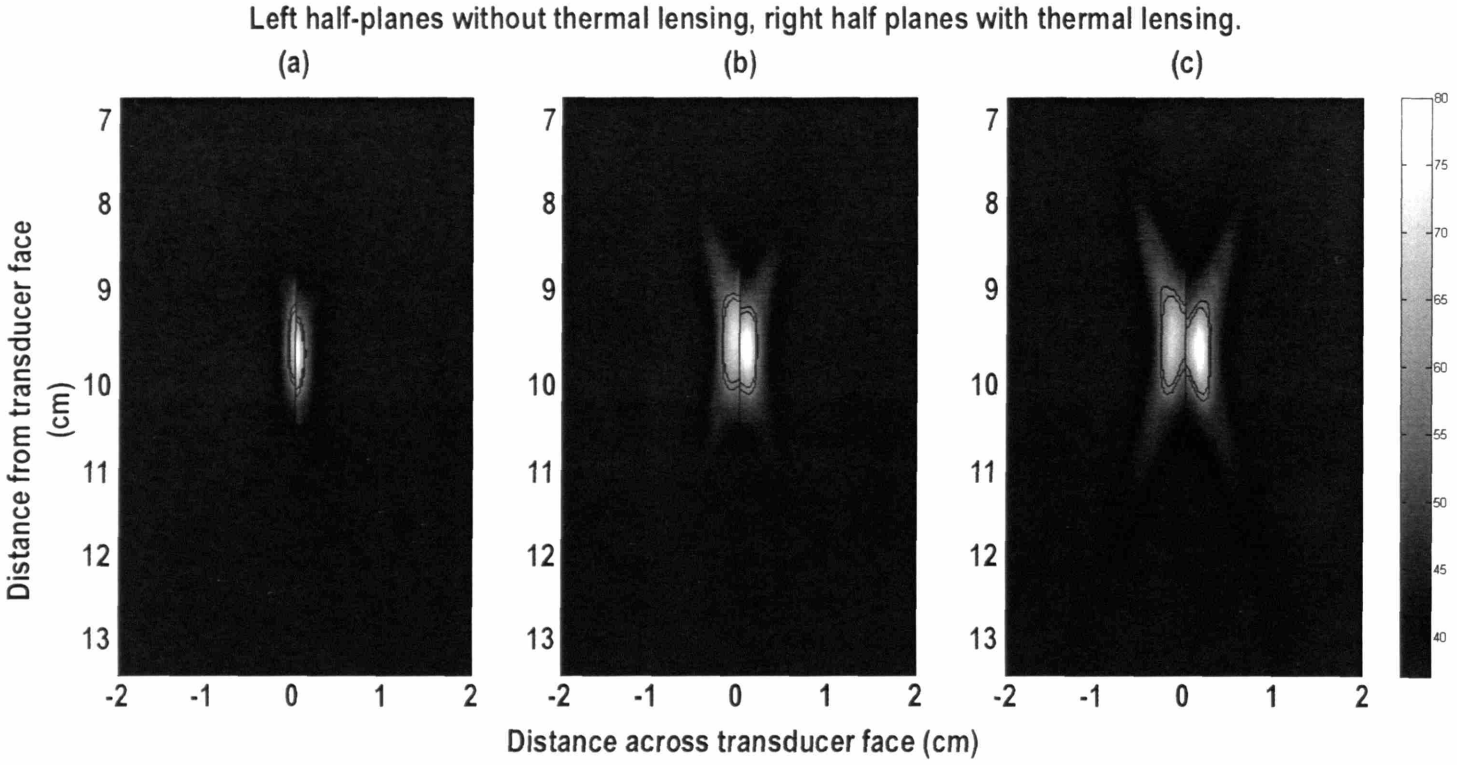

Left half-planes without thermal lensing, right half planes with thermal lensing.
(a)
(b)
(c)
Distance from transducer face (cm)
7
8
9
10
11
12
13
-2
-1
0
1
2
Distance across transducer face (cm)
80
75
70
65
60
55
50
45
40

Figure 2.8

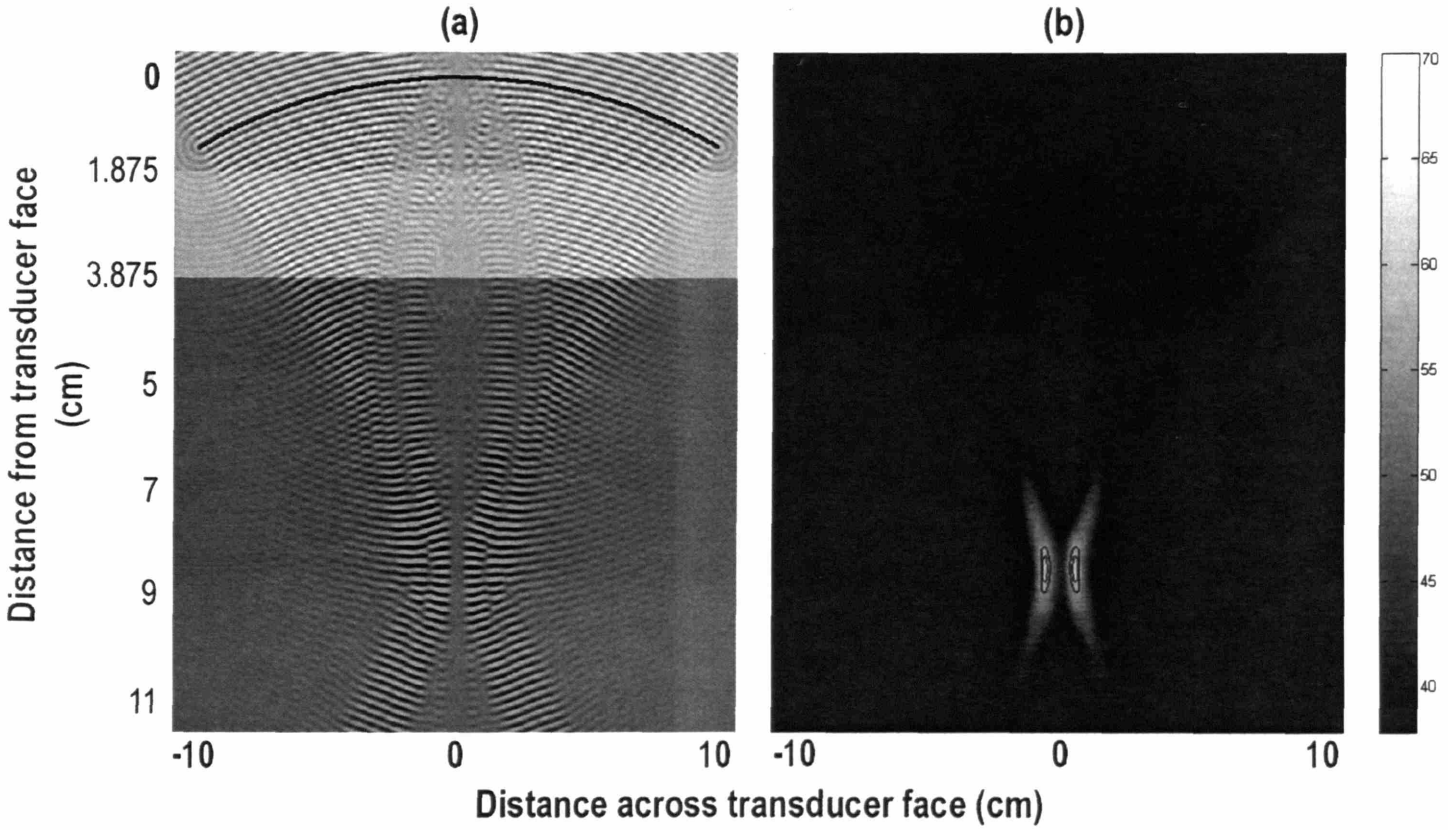

(a)
(b)
0
1.875
3.875
5
7
9
11
Distance from transducer face (cm)
-10
0
10
-10
0
10
Distance across transducer face (cm)
70
65
60
55
50
45
40

Figure 2.9

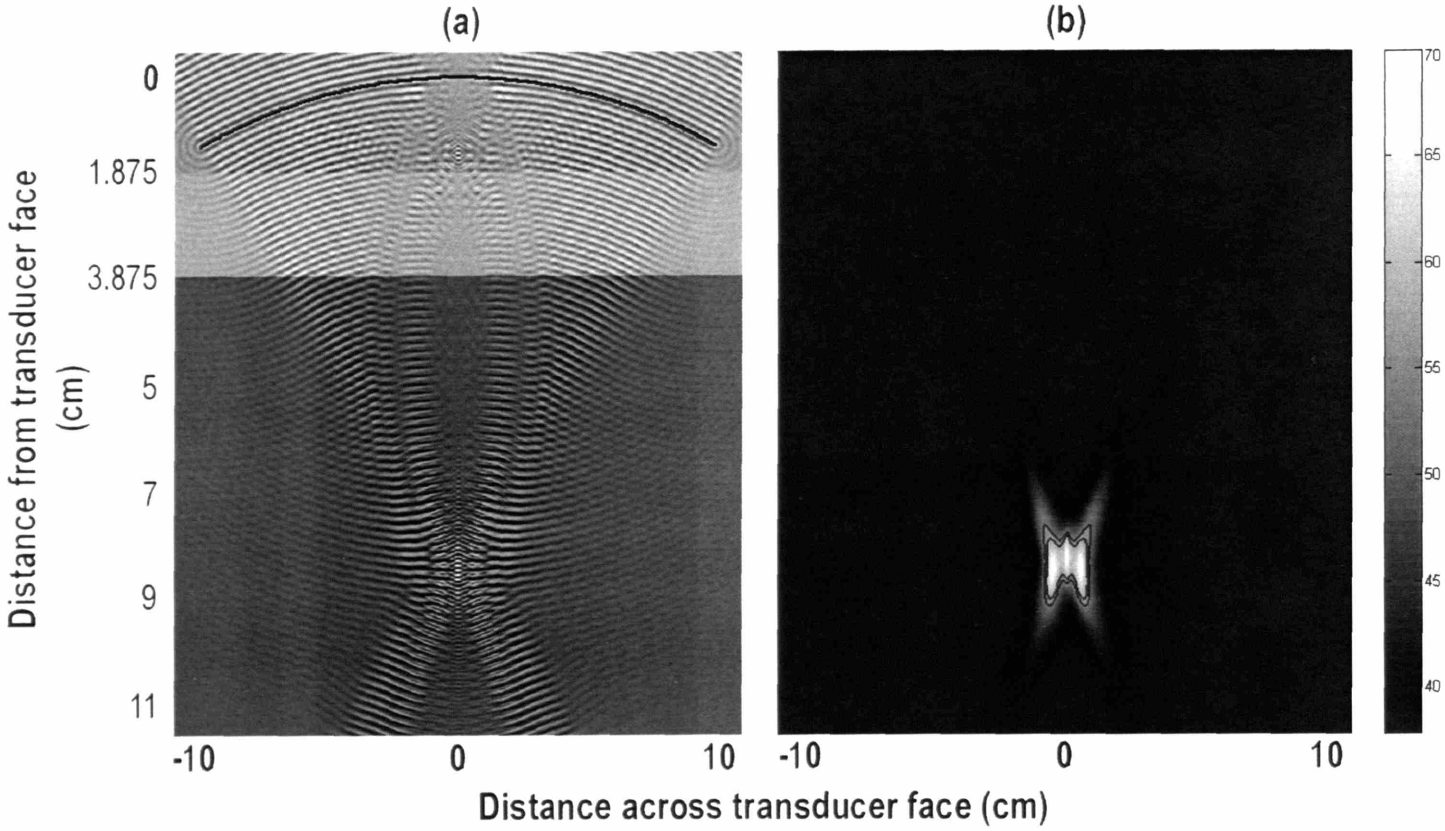

(a)
(b)
0
1.875
3.875
5
7
9
11
Distance from transducer face (cm)
-10
0
10
-10
0
10
Distance across transducer face (cm)
70
65
60
55
50
45
40

Figure 2.10

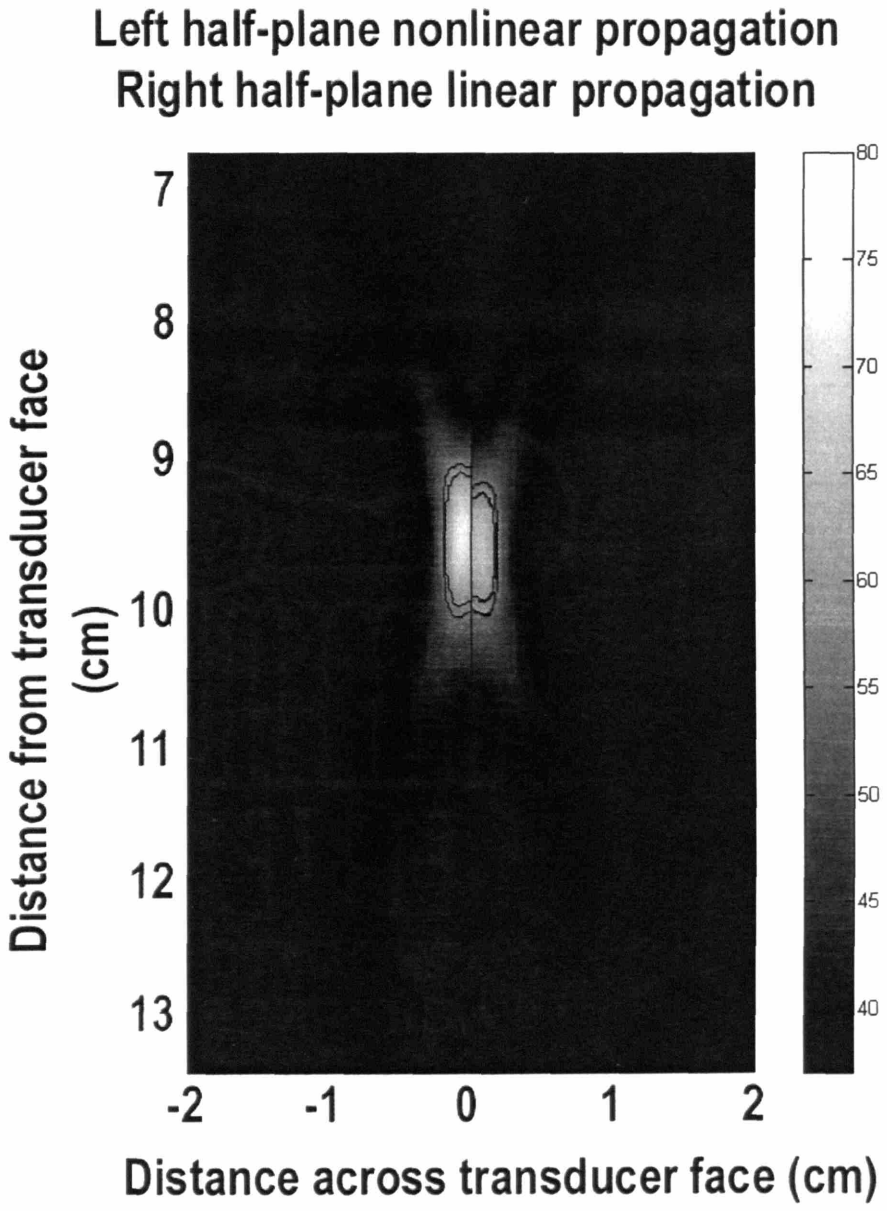
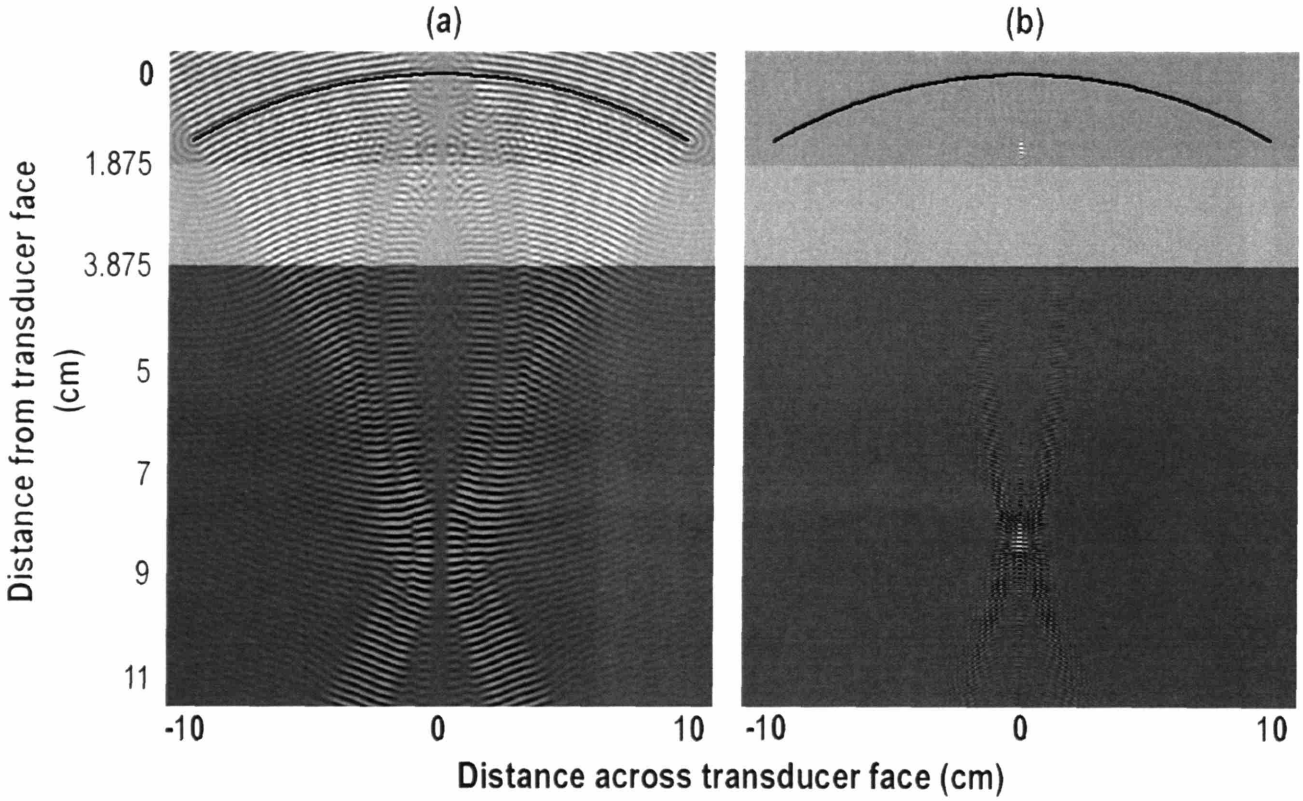

Left half-plane nonlinear propagation
Right half-plane linear propagation
Distance from transducer face (cm)
7
8
9
10
11
12
13
-2
-1
0
1
2
Distance across transducer face (cm)
80
75
70
65
60
55
50
45
40

Figure 2.11


(a)
(b)
0
1.875
3.875
5
7
9
11
Distance from transducer face (cm)
-10
0
10
-10
0
10
Distance across transducer face (cm)

Chapter 3

Therapeutic Effect of Second Harmonic Focal Spots in High Intensity Focused Ultrasound Treatment of Tissues with Nonlinear Acoustic Properties

---

The work presented in this chapter has been published as:

Connor and Hynynen, "Therapeutic Effect of Second Harmonic Focal Spots in High Intensity Focused Ultrasound Treatment of Tissues with Nonlinear Acoustic Properties", *16th Conference of the International Society for Nonlinear Acoustics*, Moscow State University, Moscow, August 19th-23rd 2003.

3.1 ABSTRACT

It has long been known that the ultrasound intensities used in high intensity focused ultrasound (HIFU) are sufficient to cause nonlinear acoustic effects. These nonlinear effects have mostly been disregarded in treatment planning since it has generally been considered that the hyperthermia induced by the focus of the fundamental ultrasound wave substantially outweighs the heating from any co-incident second harmonic waves. However, we have recently shown that certain non-axisymmetric excitation patterns of sector vortex arrays may create regions of intense heating due to the induced second harmonic field; these regions are co-incident with areas of destructive interference of the fundamental wave. Consequently, the hyperthermia induced in these regions is due solely to the second harmonic field induced by the nonlinear acoustic behavior of the tissue. As a corollary, we note that treatment prediction algorithms that neglect nonlinear acoustic behavior would produce markedly erroneous results for these acoustic configurations. These acoustic fields are produced within tissues having absorption and nonlinear characteristics equivalent to human liver, and at pressures suitable for high intensity focused ultrasound therapy. The potential therapeutic effects of these second harmonic focal spots are therefore of interest in the development of abdominal HIFU procedures.

3.2 INTRODUCTION

High intensity focused ultrasound (HIFU) procedures have been proposed for use throughout the body at tissue sites as diverse as liver [1], prostate [2], breast [3] and brain [4,5]. However, the pressure fields used in high intensity focused ultrasound surgery are sufficiently large that finite amplitude nonlinear propagation effects are produced. These effects have generally been neglected in the planning of focused ultrasound treatments since the focus of the harmonics produced by nonlinear propagation are generally co-incident with the focus of the fundamental frequency. When these foci are co-incident, the result is merely an increase in pressure at the focus over what would be predicted by linear theory and hence a faster rise in the temperature at the focus than would be expected. Using MRI-based temperature monitoring during therapy can circumscribe this problem.

However, it was recently observed [6] that certain transducer geometries could produce a harmonic focus of therapeutic intensity distinct from the fundamental focus. In cases such as these, linear acoustic analysis fails to predict the existence of this harmonic focus and so strikingly misjudges the therapeutic effect of the acoustic field. The particular case observed was the mode 4 excitation of an 8 element sector vortex array; while no therapeutic effect might be expected on the central axis of the transducer due to the destructive interference of the mode 4 field, in fact a sharply defined focus is found that is entirely composed of the second harmonic frequency.

This paper features a more detailed study of this field structure and demonstrates that the effects of nonlinear propagation may be used to increase the volume of tissue that may be treated with single sonication. Harnessing the therapeutic effect of the second harmonic focus allows an almost fifty-fold increase in thermally treated tissue volume without extending treatment time, exceeding peak temperature limits, requiring mode-switching of the transducer excitation and without creating untreated voids within the volume of treated tissue.

3.3 THEORY

3.3.1 ACOUSTIC SIMULATION

The propagation of ultrasound into tissue is modeled using the Westervelt equation. The simulation is performed on a cylindrical mesh. This geometry provides a greater density of data points around the axis of the mesh, which is typically the region of greatest interest, and it also provides a very natural treatment of the geometry of sector vortex transducers.

3.3.2 THERMAL SIMULATION

Temperature distribution throughout the tissue under sonication is modeled using the Bioheat Equation. The induced thermal load Q from absorbed acoustic power is given by:

$$Q = \frac{1}{n\tau} \int_{t'}^{t'+n\tau} \frac{\delta}{\rho c^4} \cdot \left(\frac{\partial p}{\partial t} \right)^2 dt \tag{3.1}$$

where n is a positive integer, τ is the period of the ultrasound excitation, and t' is sufficient time from the beginning of the simulation such that a steady state has been reached. δ, ρ and c are the acoustic diffusivity, local density and local speed of sound.

3.3.3 TISSUE AND TREATMENT MODELING

The effect of a given temperature profile on tissue can be determined using Sapareto and Dewey's function. This function maps a time course of thermal exposure to an equivalent duration of exposure at 43°C, as follows:

$$T_{43} = \int_0^t R^{T_{ref} - T(t)} dt \qquad R = \begin{Bmatrix} 0.5, T \geq T_{ref} \\ 0.25, T \leq T_{ref} \end{Bmatrix} ; T_{ref} = 43^\circ C \tag{3.2}$$

where T(t) is the temperature at a particular point in the tissue at time t. The limit for tissue death is T_{43} greater than 240 minutes. The tissue is simulated to be under sonication from an 8-element sector vortex array of diameter 10cm with radius of curvature of 10cm immersed in water. There is a 5mm gap between the transducer and the tissue. The tissue consists of a 2cm layer of fat overlaying liver, both of which are appropriately perfused. The relevant model parameters for this tissue model are shown in Table 3.1.

3.4 Results

By varying the pressure of the acoustic wave at the transducer, the extent of nonlinear acoustic effects can be judged against the linear simulation of the acoustic field at equivalent pressure. An acoustic pressure amplitude of 2.5 Bar at the transducer demonstrated a focus at the mode 0 focal position for the second harmonic that was appropriately intense to compensate for the destructive interference of the fundamental frequency at this position, as shown in Figure 3.1. Thus a sufficiently large nonlinear mode zero second harmonic focus can be created to balance the absence of a similarly located fundamental focus, allowing a mode 4 sonication to be used without the risk of allowing an untreated region within the mode 4 focus.

This balanced field allows a greater volume of tissue to be treated with a single sonication than is possible with a standard mode zero field, given the constraint that a peak desired temperature not be exceeded within the treatment volume. Considering the treatment constraint of a peak temperature within tissue of 80°C after 15 seconds of heating and allowing 15 seconds of post-treatment cooling, the tissue coagulation patterns shown in Figure 3.2 are produced for mode zero and mode 4 sonication respectively. Using a balanced nonlinear field increases the treatment volume almost fifty-fold for the same treatment duration.

3.5 Conclusion

Under certain transducer geometries, the nonlinear properties of soft tissue can produce entirely distinct, therapeutically active foci that are not properly modeled with linear acoustic theory. However, when appropriately harnessed, these nonlinear foci can contribute to the useful therapeutic effect of a HIFU treatment profile. The pressures required to produce such foci are within the scope of HIFU procedures.

3.6 REFERENCES

[1] G.R. ter Haar, D. Sinnett, and I. Rivens, High intensity focused ultrasound - a surgical technique for the treatment of discrete liver tumors *Phys Med Biol.*, vol. 34, pp. 1743-1750, 1989.

[2] N.T. Sanghvi, *et al.*, Noninvasive surgery of prostate tissue by high-intensity focused ultrasound. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 43, pp. 1099-1110, 1996.

[3] K. Hynynen, *et al.*, MR imaging-guided focused ultrasound surgery of fibroadenomas in the breast: a feasibility study. *Radiology*, vol. 219, pp. 176-185, 2001.

[4] M. Tanter, J.L. Thomas, and M. Fink, Focusing and steering through absorbing and aberrating layers: application to ultrasonic propagation through the skull *Journal of the Acoustical Society of America*, vol. 103, pp. 2403-2410, 1998.

[5] K. Hynynen and F.A. Jolesz, Demonstration of potential noninvasive ultrasound brain therapy through an intact skull *Ultrasound in Medicine and Biology*, vol. 24, pp. 275-283, 1998.

[6] C.W. Connor and K. Hynynen, Bio-acoustic thermal lensing and nonlinear propagation in focused ultrasound surgery using large focal spots: a parametric study *Phys Med Biol*, vol. 47, pp. 1911-1928, 2002.

3.7 TABLES

Table 3.1: Typical material properties of relevant human tissues, where ρ is density, c is speed of sound, α_0 is the acoustic amplitude absorption, β is the acoustic nonlinearity parameter, k is thermal conductivity, C is specific heat capacity and W is the volumetric blood perfusion rate.

	ρ [kg m^{-3}]	c [ms^{-1}]	α_0 [Np m^{-1} MHz^{-1}]	β [kg^3 m^{-4} s^{-2}]	k [W m^{-1} K^{-1}]	C [J kg^{-1} K^{-1}]	W [ml cm^{-3} s^{-1}]
Water	1000	1500	2.88×10^{-4}	3.5	0.615	4180	
Fat	910	1430	9	10.5	0.248	2490	5×10^{-4}
Liver	1050	1596	4.5	6	0.512	3600	1.33×10^{-2}
Blood	1030					3620	

3.8 FIGURES

Figure 3.1: Thermal profile in liver after 15s sonication at 2.5 Bar using linear and nonlinear simulation methods.

Figure 3.2: Comparison of therapeutic outcome for mode 0 and mode 4 treatments not exceeding 80°C temperature.

Figure 3.1

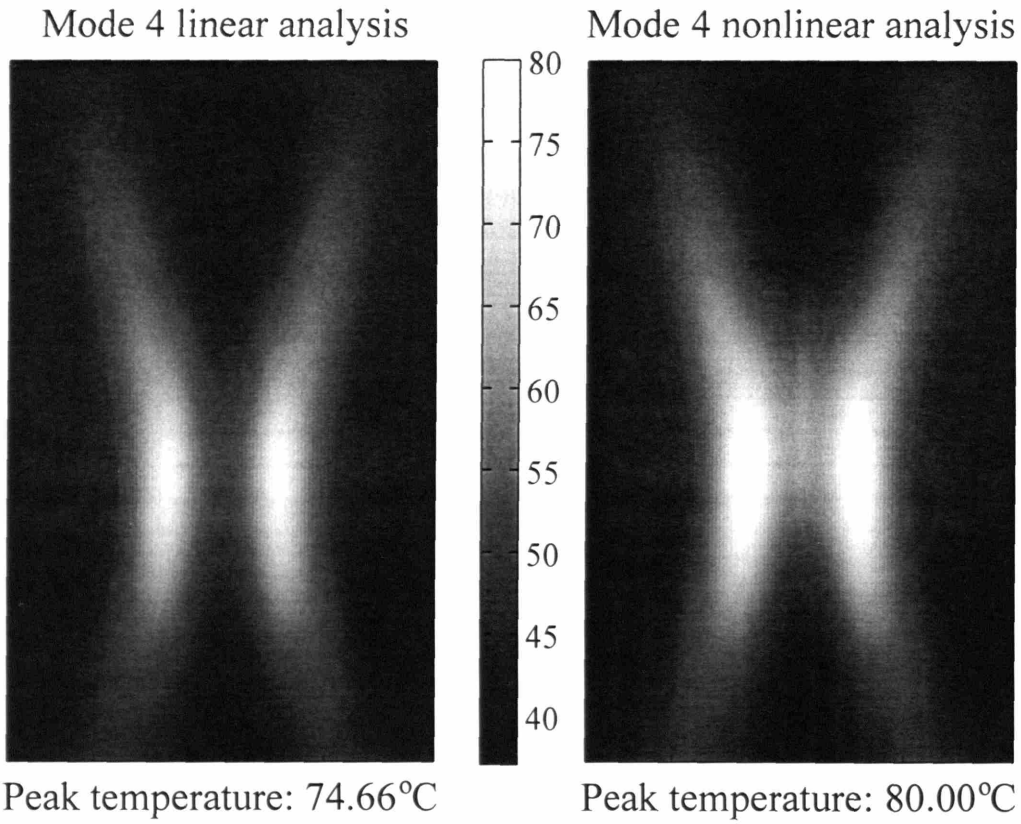
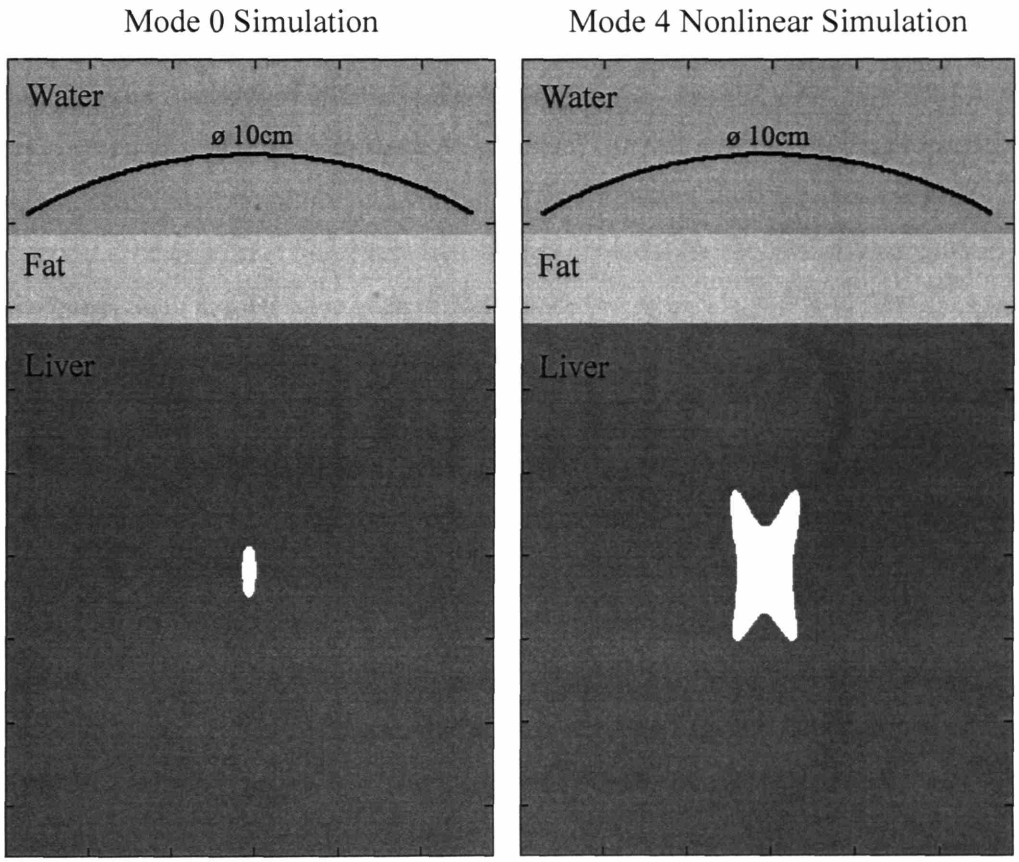

Mode 4 linear analysis
Mode 4 nonlinear analysis
80
75
70
65
60
55
50
45
40
Peak temperature: 74.66°C
Peak temperature: 80.00°C

Figure 3.2


Mode 0 Simulation
Mode 4 Nonlinear Simulation
Water
ø 10cm
Fat
Liver
Water
ø 10cm
Fat
Liver
Treatment Volume: 53mm3
Treatment Volume: 2603mm3

Chapter 4

A Unified Model For The Speed Of Sound In Cranial Bone Based On Genetic Algorithm Optimization.

The work presented in this chapter has been published as:

Connor, Clement and Hynynen, "A Unified Model For The Speed Of Sound In Cranial Bone Based On Genetic Algorithm Optimization", *Physics in Medicine and Biology* (47), pp. 3925-3944.
The empirical measurements of transcranial acoustic phase shift were performed by Greg Clement.

4.1 Abstract

The density and structure of bone is highly heterogeneous, causing wide variations in the reported speed of sound for ultrasound propagation. Current research on the propagation of high intensity focused ultrasound through an intact human skull for non-invasive therapeutic action on brain tissue requires a detailed model for the acoustic velocity in cranial bone. Such models have been difficult to derive empirically due to the aforementioned heterogeneity of bone itself.

We propose a single unified model for the speed of sound in cranial bone based upon the apparent density of bone by CT scan. This model is based upon the coupling of empirical measurement, theoretical acoustic simulation and genetic algorithm optimization. The phase distortion caused by the presence of skull in an acoustic path is empirically measured. The ability of a theoretical acoustic simulation coupled with a particular speed of sound model to predict this phase distortion is compared against the empirical data, thus providing the fitness function needed to perform genetic algorithm optimization. By performing genetic algorithm optimization over an initial population of candidate speed-of-sound models, an ultimate single unified model for the speed of sound in both the cortical and trabecular regions of cranial bone is produced.

The final model produced by genetic algorithm optimization has a non-linear dependency of speed of sound upon local bone density. This model is shown by statistical significance to be a suitable model of the speed of sound in bone. Furthermore, using a skull that was not part of the optimization process, this model is also tested against a published homogeneous speed of sound model and shown to return an improved prediction of transcranial ultrasound propagation.

4.2 INTRODUCTION

Focused ultrasound provides a non-invasive method for inducing temperature elevations in tumors [1-6], resulting in the coagulation of the tumor. It has long been recognized that the ability to produce a high intensity ultrasound focus within the brain, and hence non-invasive cautery at that focus due to acoustic absorption by brain tissue, would provide a minimally invasive means of treating deep-seated brain tumors without disturbing overlying structures [7,8]. It is clearly desirable to produce this focus through an intact skull rather than requiring craniotomy to establish an acoustic window, which has led to interest in the study of the propagation of ultrasound through the bones of the skull [9-11]. Although bone in general was first considered to be an impassible barrier to therapeutic ultrasound, and then was considered to induce too much distortion into the acoustic field to allow a therapeutic focus to be formed [12,13], simulation and experimental studies have since shown that a therapeutic focus may be formed inside an intact skull if suitable phase compensations are applied over the ensonicating field by means of a segmented transducer [10,14].

An entirely non-invasive therapeutic ultrasound system using transcranial focusing clearly requires that the appropriate phase corrections be determined non-invasively [15]. Acoustic simulation can be utilized to determine the distortion an intact skull would introduce into an incident uniform ultrasound field, and hence to determine the appropriate phase compensations to reverse this distortion [16]. Potentially, many different acoustic simulation techniques may be applied to this problem, including acoustic time reversal, inversion of linear filters, inversion of Helmholtz operators, Rayleigh-Sommerfield analysis, and formulations such as KZK and linear or non-linear finite-difference time-domain (FDTD) [17-20]. These simulations are based on the reconstruction of CT data to provide the geometry and composition of each individual skull [16]. CT data in Hounsfield units may be readily normalized to apparent density by ensuring that materials of known density are included in the scan.

The field correction calculations under all these simulation methods are dependent on the velocity with which the ultrasound is propagated through the skull. However, skull is a heterogeneous material, so it does not have a single speed of sound throughout. Furthermore, its heterogeneity makes it difficult to determine empirically what the variation in speed of sound throughout a sample of skull might be. This difficulty is reflected in the literature on the material properties of the skull: estimates of the speed of sound in skull are given as speed of sound through some bulk sample of skull bone, referred to as the plesiovelocity [21]. The consensus across the literature is not especially convincing. A common approach to attempt to improve on these circumstances is to divide the skull into a three-layer model [22], comprising an inner and outer table of cortical bone between which trabecular bone is sandwiched. This distinction into layers is somewhat arbitrary since much skull bone is of an intermediate porosity and hence is not robustly assignable to either category. Sharp divisions between the three layers are similarly seen to be lacking in anatomical specimens.

Hence we find ourselves in the situation of requiring a functional model of local speed of sound in bone based on the apparent bone density at a particular point, but seemingly confounded from producing such a model empirically due to the very variability of the material under study. We propose that this problem may be made tractable by using an empirical acoustic approach combined with the genetic algorithm optimization [23] of a population of candidate functions relating apparent bone density to local speed of sound.

4.3 METHOD

4.3.1 OVERVIEW

The acoustic properties of a material in bulk may be measured by examining the transmission of ultrasound through a sample of that material, and this technique may be used to examine the bulk properties of a heterogeneous material such as bone [24]. In this study, we concentrate on the phase difference between the received wave and the transmitted wave in continuous wave sonication. In particular we examine the change in this phase difference between the case where only degassed water is present between the transducer and receiver and the case where a human calvarium is placed between the transducer and the receiver.

The human calvaria used empirically above are first CT scanned at high clinical resolution. Once the overall geometric structure and the density distribution of the calvaria have been acquired, the experiment described above can be reproduced as a computational simulation of acoustic propagation. The change in phase difference measurement described above is calculated for the simulation. This measurement is acutely sensitive to the speed of sound in the bone. The problem then becomes one of optimizing the model used for speed of sound in bone in the simulation.

This optimization problem is ideally suited to treatment with a genetic algorithm since there is a well-defined, quantifiable success function, which is the total deviation of the simulated phases from the empirical phases. Furthermore the genetic algorithm optimization technique does not depend on gradient information that may be unacceptably noisy since it would only be available by numerical approximation in this case. This optimization method is also relatively insensitive to local minima and maxima. The genetic algorithm optimization technique can be easily tailored to known physical constraints such that only physically realistic models are permitted [25].

4.3.2 EMPIRICAL CONFIGURATION

The calvaria were maintained in wet condition so that their acoustic properties remain comparable to *in vivo* bone. The CT scan was performed using a Siemens Somatom Plus 4 with deconvolution kernel AH-82. The calvaria were placed within stereotactic frames as shown in Figure 4.1 to allow their accurate positioning during both the experimental procedure and CT scanning. Fiducial markers on the frame allow the position and orientation of the skull during the experimental procedure to be accurately matched to a solid body geometric transformation of the reconstructed data.

Ultrasound sonication was performed in degassed, deionized, room-temperature water at 0.74MHz, which is within the acoustic window for skull bone [26] while being of an appropriate frequency for the accurate measurement of the phase component. While an increase in temperature in cranial bone may produce changes in the phase of the received ultrasound wave [27], the amplitude and duration of the sonication used here are such that no appreciable change in skull temperature occurs during measurement.

A 500-element hemispherical ultrasound transducer was used to provide ultrasound transmitters. Forty transducer elements were selected at random from the 500 total elements. These elements were activated in turn while a 0.2 mm-diameter Polyvinylidene difluoride (PVDF) hydrophone (Precision Acoustics, UK) located at the geometric center of the hemispherical transducer by a three-dimensional stepper motor positioning system (Parker, Hannifin, Pennsylvania) was used as the receiver. This allowed the rapid assaying of forty different acoustic paths through a particular calvarium. The phase information for each acoustic path was acquired automatically using a pre-amp (Precision Acoustics, UK) recorded by digital GPIB IEEE-488 compliant oscilloscope (Textronix, Oregon, Model 380). The empirical experimental design here is similar to that used in Clement *et al.* [16].

4.3.3 GENETIC ALGORITHM REPRESENATION

A model for speed of sound in bone can be determined by admitting an initial large pool of candidate speed of sound models, and ascertaining how well these simulations predict the phase changes as established by empirical study. These speed of sound models are then used to produce a new generation of speed of sound models that are similarly studied. However, it is required that these speed of sound models be stated in a form amenable to manipulation by genetic algorithms.

Bone may be considered to be a composite of inorganic mineral within an organic framework. The mineral component is predominantly hydroxyapatite, which may be present in a crystalline form with a density of up to 3170 kg m^{-3} [28]. Type I collagen contributes 90% of the organic material and, like most organic materials, has a density similar to that of water [29]. The speed of sound in skull in bulk has not been reported to be greater than 3500ms^{-1}. The fastest reported speed of sound for bulk ultrasonic propagation in wet non-dental bone is 4180ms^{-1} [30], which was seen in cow tibia. Faster speeds of sound are possible in dry bone as the collagen matrix stiffens, but this is not representative of the *in vivo* case.

The genetic algorithm representation of the speed function is as five "knots" arranged on a graph with densities between 1000 and 3400 kg m^{-3} and speeds of sound between 1500ms^{-1} and 5500ms^{-1}. The speed of sound and density of water establish the lower bounds. The upper bound is set slightly above the maximum possible density of 3170 kg m^{-3}; optimization techniques may perform poorly if the maximum of the optimization is located in a corner of the search space, and this small margin potential alleviates that risk. The maximum speed of sound is set higher than the reported maximum speed of sound not only to aid the optimization process but also because the reported maximum speed of sound above is given as a bulk property and we must allow for the possibility that the speed of sound in some small portion of the tissue may be much greater.

The continuous model of the speed of sound can now be represented in the discrete allelic form required for this method of optimization [25,31]. The function $c(\rho)$, the model speed of sound dependent on the local apparent bone density, is generated from these allelic knots by minimizing [32]:

$$\upsilon.\sum_{i=1}^{5}\left[c_i - c(\rho_i)\right]^2 + (1-\upsilon).\int_{1000}^{3400}\left(\frac{d^2c}{d\rho^2}\right)^2 d\rho \qquad (4.1)$$

where c_i and ρ_i are the allelic knots. υ is a numeric value between 0 and 1 that balances the relative importance of fitting the knots closely with the requirement that the function $c(\rho)$ be reasonably smooth. This operation is part of the class of cubic smoothing splines. This avoids the well-known problem with polynomial function fitting where the knots are perfectly fit but the function displays unacceptable hilliness in between the knots. Selecting a value of $\upsilon = 0.99$ places strong emphasis on fitting the knots (the first part of the equation) while providing acceptably smooth and responsive fitting in the regions between.

Functions that are not monotonically increasing or nearly so are discarded as unacceptable genetic forms: all functions must satisfy the admissibility equation (4.2) where $c_{tolerance}$ is set to 200 ms^{-1}.

$$c(\rho_0) + c_{tolerance} \geq \max_{1000<\rho<\rho_0}\left(c(\rho)\right) \; ; \; 1000 < \rho_0 \leq 3400 \qquad (4.2)$$

In optimization by genetic algorithm, it is important to allow a certain level of tolerance in enforcing limits such as these so that potentially useful genetic alleles are not entirely discarded out of hand. Curves such as those shown in Figure 4.2 are representative of the types of speed of sound models simulated.

A quantitative success function is required to measure the fitness of each particular speed of sound model. A parameter θ_e is calculated for each acoustic path; this is the error between the phase change as calculated by the simulation and the phase change as measured empirically. The success function is then defined as $\overline{\cos\theta_e} \times 100\%$, i.e. the arithmetic mean of the cosines of the θ_e distribution as a percentage. The success function

has a range of 100% for a perfect correlation down to –100% for a perfect anti-correlation; the success function is a phase correlation metric.

The alleles are stored as floating point numbers as shown in Figure 4.2, hence the genetic algorithm is of the real-coded type. Most original work on the robustness of genetic algorithms was performed on binary coded algorithms. However, by extension of the demonstration of the robustness of genetic algorithms over non-binary alphabets [33], real-coded algorithms have also been shown to perform in accordance with Holland's schemata [23,34].

Each generation of models is evolved from the previous one by the three steps of competitive tournament selection, breeding based on allelic crossover, and the application of a low rate of spontaneous mutation [35].

Firstly, the tournament selection step is performed by selecting two models at random from the current population. Of these two models, the one that produced the best score using the success function is allowed to form part of the breeding pool that produces the next generation. Models chosen in this way are not removed from the current population. This produces a breeding pool that is guaranteed not to contain the worst performing model from the current population, and also should favor the inclusion of better performing models. This selection method has the two further useful properties. It has limited susceptibility to being overwhelmed early in the optimization process by a single highly successful model, which can cause problems in genetic algorithm optimization due to loss of population diversity. This selection method also is not dependent on the absolute value returned by the success function and simulation method, but only on the relative value of the success function between models: thus in this case, the only requirement is that a better speed of sound model should generally produce a better result in the success function. In practice, this means that the success function can exhibit non-linear or noisy behavior without adversely disturbing the optimization process.

Secondly, two parent models are selected at random from the breeding pool, and a new child model is produced from this pair and becomes a member of the next generation. Eighty percent of the time, this new child model is produced by selecting a random point in its alleles and including the alleles of the first parent model up to that point, and the alleles of the second parent model after that point; known as allelic crossover. Twenty percent of the time, the child model is a direct copy of a parent. This tends to allow successful models to propagate through to the next generation.

Thirdly, ten percent of the time a random mutation is added to the alleles of a particular model, with a permissible mutation range of ± 200 kg m^{-3} and ± 200 ms^{-1}. While this mutation rate would be high for binary-coded algorithms where a single bit mutation can produce a huge variation in the numerical value it represents, it is appropriate for real-coded genetic algorithms, which are more robust in this sense.

Any models in this new generation that are not physically valid by the criteria given above are deleted, and the second and third steps above are repeated until a complete new generation is produced. Once this is achieved, parallel processing is used to increase the speed of the genetic algorithm optimization; many simulations are performed concurrently over a network of computers [36].

4.3.4 SIMULATION OF ACOUSTIC PROPAGATION

It is vitally important that the acoustic path that is simulated match the acoustic path studied empirically. Each experimental calvaria is mounted in a stereotactic frame to ensure that it can be rigidly and reproducibly positioned in the hemispherical array described in section 4.3.2. The positions of the fiducial markers on the frames can be measured relative to the hemispherical ultrasound transducer using a three-dimensional Cartesian calibrated positioning system. By identifying these same fiducial markers in the CT data, it is then possible to determine the geometric solid body transformation from the orientation of the skull in the CT data set to the orientation of the skull in the

experimental apparatus. This transformation can be represented in the classical augmented matrix form:

$$\left[\begin{array}{ccc|c} & \vdots & & \vdots \\ \cdots & \mathbf{R} & \cdots & \mathbf{T} \\ & \vdots & & \vdots \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{4.3}$$

where **R** is a 3x3 matrix describing rotation of the skull and **T** is a 3-element vector describing translation of the skull. The determinant of **R** is necessarily unity as we require a solid body transformation since the skull does not change in size. A minimum of three fiducial markers that are not co-linear is required, although the fidelity of this transformation is improved by using more markers. Determining the optimum values of **R** and **T** for arbitrarily many fiducial markers is an over-determined problem; fortunately Horn [37] developed an elegant closed-form solution in quaternions for determining orientation in visual systems that is mathematically equivalent to this problem. There is a unique correspondence between a solid body transformation and its quaternion. A typical 3D computer reconstruction of the skull CT data coupled with a 500 element hemispherical array is shown in Figure 4.3.

As the bones of the skull are heterogeneous, an algorithm is required that is capable of simulating acoustic propagation in a material with rapidly changing local acoustic properties. Acoustic propagation is simulated using a finite difference time domain (FDTD) algorithm based on the following formulation of the Westervelt [38] equation:

$$\frac{\partial^2 p}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3} + \frac{2\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \frac{\partial p}{\partial z}\cdot\frac{\partial(\ln \rho)}{\partial z} = 0 \tag{4.4}$$

where p is the pressure of the acoustic wave, ρ is the local density, δ is the local acoustic diffusivity, β is the co-efficient of non-linearity, *and* c is the local speed of sound.

The local density at a particular point is generated from the CT scan of the calvarium. CT scans return data as Hounsfield Unit (*HU*) rather than as density. This problem can be overcome by ensuring that a sample of both water and air appear in the CT scan. Since

the densities of water and air can be reasonably taken to be 1000 kg m^{-3} and 0.0 kg m^{-3} respectively, a linear normalization relation can be produced where:

$$\kappa_1 = \frac{1}{H_{water} - H_{air}} \quad ; \quad \kappa_0 = \frac{-H_{air}}{H_{water} - H_{air}} \tag{4.5}$$

and thus for any particular point we state that:

$$\rho_{ct} = \kappa_1 HU + \kappa_0 \tag{4.6}$$

where ρ_{ct} is the apparent local density based on the CT scan in kg m^{-3}, and κ_1 and κ_0 are constants with units of kg m^{-3} HU^{-1} and kg m^{-3} respectively.

For harmonic excitation, $\delta = \frac{2\alpha c^3}{(2\pi f)^2}$ where α is the acoustic amplitude attenuation co-efficient and f is the frequency of the ultrasound wave. As stated in section 4.3.2, here f = 0.74MHz to harness the skull acoustic window. While a precise consensus on the amplitude attenuation co-efficient is not available from the literature, a value of $\alpha = 167 \times \frac{f}{10^6}$ Np m^{-1} for cortical bone and $\alpha = 300 \times \frac{f}{10^6}$ Np m^{-1} for trabecular bone is consistent [22,39,40]. In order to decide which α value should be used, we use the guideline that trabecular bone may be considered to be that bone structure which is composed of 70% or less bone matrix [41]. Cortical bone in bulk has a density of approximately 1950 kg m^{-3} [22,42], so for the purposes of absorption we assign the higher value of α to bone with $\rho_{ct} < (0.7 \times 1950 + 0.3 \times 1000)$ kg m^{-3}, i.e. $\rho_{ct} < 1665$ kg m^{-3}. The value of the non-linear parameter β is set to zero, effectively simulating linear acoustic propagation. The pressure amplitude of the ultrasound used empirically is sufficiently low that non-linear acoustic effects are negligible; the Goldberg number Γ is also expected to be small due to the high attenuation α[43]. Furthermore, the value of β in bone has not yet been satisfactorily determined.

Discretization parameters Δt and Δz are chosen to ensure numerical stability in the simulation. This requirement is enforced by requiring the Courant-Friedrichs-Levy (CFL) number to be less than or equal to unity at all points, where $\mathrm{CFL} = c\Delta t/\Delta z$ [44].

4.3.5 THE COMPLETE METHOD BY ACTION

A genetic algorithm optimization structured as the block diagram in Figure 4.4 provides a robust method for determining an appropriate speed of sound model. The method is precisely defined as the following steps:

1. The calvarium of a human skull is placed within a stereotactic positioning frame and scanned at high resolution using a clinical CT scanner (Siemens Somatom Plus 4, Kernel AH-82).
2. A 500-element hemispherical ultrasound transducer is placed in a baffled tank filled with degassed and deionized water. A needle hydrophone is placed at the geometric center of the transducer.
3. A random selection of 40 elements is made from the 500 elements of the hemispherical ultrasound transducer.
4. Each selected element is activated in turn and the phase of the ultrasound received at the hydrophone is recorded, giving the phase shift between the transducer and the hydrophone in water alone.
5. The calvarium is mounted between the transducer and the hydrophone using the stereotactic apparatus so that the relative alignment of the transducer, skull and hydrophone can be determined and repeatedly established.
6. The 40 selected elements are again activated in turn and the phase of the ultrasound received at the hydrophone is recorded. The differences between the phase measurements without and with skull present are calculated.
7. A three dimensional model of the skull is reconstructed computationally, and the CT data in Hounsfield Units is normalized to apparent density. The alignment between the transducer and stereotactically mounted skull is calculated.

8. The acoustic propagation is computationally simulated given a certain model of acoustic velocity in skull bone. The phase changes predicted by simulation are compared against the empirically measured phase changes.
9. This process is repeated across 4 skulls and for 50 initial speed of sound models.
10. The next generation of 50 speed of sound models are evolved using the principles of genetic algorithm optimization, namely selection, mutation and crossover as elaborated in section 4.3.3. [23,35]
11. The process is repeated for 50 generations.
12. The best speed of sound model is chosen by selecting the model that has produced the best phase correlation metric as defined in section 4.3.3.

4.4 RESULTS

Ultimately, after 50 generations, the model shown in Figure 4.5 was produced for the speed of sound against apparent bone density by CT scan. This model is stated in numerical form in Table 4.1.

4.4.1 STATISTICAL SIGNIFICANCE IN PHASE CORRELATION MEASUREMENTS

Since the speed of sound model generated in this paper was produced by its ability to generate correctly model phase shifts across skull, the null hypothesis for this study is that the model produces no better correlation between simulation and empirical data than would be produced by choosing the modeled phase shifts randomly.

To determine whether the speed of sound model is statistically significant, it is first necessary to characterize the distribution of randomly chosen phase shifts. This distribution is shown in Figure 4.6, and is based upon 100000 trials of sets of 40 randomly generated phase shifts.

A statistical significance curve as shown in Figure 4.7 can then be calculated. For a given phase correlation value, this curve gives the proportion of random phase assignations that would be expected to produce a better phase correlation. Only 5% of random phase assignations would be expected to produce a better phase correlation value than 27.3%, and thus a model that is capable of producing phase correlations better than 27.3% is statistically significant at $p<0.05$ from the point of the view of the null hypothesis described above.

In fact the best speed of sound model generated by the genetic algorithm produced a phase correlation value of 53.4%, and is therefore very highly significant.
Previous work in the literature of speed of sound of models for bone includes an interesting model proposed by *Clement et al.* [16] based upon assigning an effective

speed of sound to a region of skull based on an average of the densities in that region obtained by CT scan [16]. The relationship between effective speed of sound and density in *Clement et al.* is:

$$c_{eff} = 2.06\overline{\rho} - 1540 \tag{4.7}$$

where c_{eff} is in ms^{-1} and ρ is in kg m^{-3}. When applied to the data in this study, this model produced a phase correlation value of 49.2%; it is therefore outperformed here by the speed of sound model produced by genetic algorithm. The result of 49.2% is also very highly statistically significant by the method described above.

4.4.2 COMPARISON OF THE GENETIC ALGORITHM SPEED OF SOUND MODEL TO A SPEED OF SOUND MODEL BASED ON THE HOMOGENIZATION OF SKULL.

This paper describes the production of a general model for the speed of sound in skull, based upon the analysis of a finite number of experimental calvaria. This problem is therefore of the class of problems where an optimization process is used to produce a generally applicable result from a pool of empirical data. Therefore, it must be established that the genetic algorithm has general applicability by using it in the analysis of ultrasound propagation through a calvarium that was not used in the optimization procedure. This is necessary to show that the optimization is not over-trained; over-training occurs when an algorithm performs well on members of the original dataset against which it was formed, but performs very badly on novel datasets.

An unused calvarium was placed in the same experimental apparatus as described in section 4.3.2 above, and the phase shifts due to transcranial propagation of the ultrasound were measured in 116 elements corresponding to the top 6 layers of the 500 element hemispherical array as shown in Figure 4.3. The top 6 layers were chosen because elements chosen near the rim of the transducer in this experimental configuration may have greater accessory pathways to the focus point through water without impinging on skull, as may be inferred from Figure 4.3. Choosing the elements that comprise the top 6

layers of the transducer gives a representative number of elements over which to compare the two models while also ensuring transcranial propagation to the focus. Ultrasound propagation was modeled using the FDTD algorithm described in Equation (4.4). However, two simulations were run: the first simulation used the effective density speed of sound model [16] as shown in Equation (4.7), and the second simulation used the speed of sound model derived in this paper by genetic algorithm optimization as shown in Figure 4.5 and Table 4.1.

The distribution of θ_e (as defined in section 4.3.3) for each of the two speed of sound models is shown for this novel calvarium in Figure 4.8. These curves show the proportion of elements that have a θ_e greater than a particular error margin; all curves must begin at 100% at $\theta_e = 0°$ and end at 0% at $\theta_e = 180°$. Better algorithms have traces that run lower on the graph. The genetic algorithm speed of sound model outperforms the effective density speed of sound model across the range of θ_e. In particular, it is apparent that the effective density model suffered by having over 30% elements with $\theta_e > 90°$, which counts poorly in the phase correlation metric $\overline{\cos\theta_e} \times 100\%$ as defined in section 4.3.3.

The effective density model produced a phase correlation metric of 28.36%. The genetic algorithm speed of sound model, however, produced a phase correlation metric of 44.8%, which is markedly better when judged in light of the statistical analysis given above in section 4.4.1. Furthermore, this result is of a broadly equivalent level of performance to that seen with the calvaria used in the optimization procedure, and as such refutes the presence of substantial over-training in the speed of sound model.

The presence of elements with $\theta_e > 90°$ with the genetic algorithm speed of sound model appears to be due primarily to anomalies in the material geometry of the skull that are not well analyzed by the comparatively simple one dimensional FDTD model described in Equation (4.4). In particular, these errors tend to arise when complex lacunae are present in the simulation; Figure 4.9 shows a portion of the CT data for such a case where a substantial lacuna is present in the inner table of the calvarium along the simulation path. Of the elements with $\theta_e > 90°$ with the genetic algorithm speed of sound model, 73% had

readily apparent lacunae on the simulation path on inspection. Ideally the genetic algorithm optimization would be conducted with a full three-dimensional simulation of acoustic propagation. However, even using suitable distributed computing techniques, to perform the optimization in this way with an equivalent population of speed models and number of generations might be expected to have taken several years with current computing power. Consequently, a full three-dimensional treatment is not possible at this time.

4.5 Discussion

This paper describes a novel method for establishing a model of the speed of sound in cranial bone. An important difference from previous work is that this method produces a model for the speed of sound at any particular point in the skull; the model is not dependent on dividing the cranial bone into regions or layers or on ascribing homogenized properties to any part of the skull. This model is valid for cranial bone of both cortical and trabecular structure, and is based on a coupled strategy of genetic algorithm optimization of CT-based acoustic simulation and empirical measurement of the change in phase of an ultrasound wave on transmission through human calvaria.

The paper also details the results of this novel method in the form of a model for the speed of sound in cranial bone dependent on local density as shown in Figure 4.5 and as described in Section 4.4 above, and further establishes the statistical significance of this result and compares its performance to another speed of sound model in the literature [16]. As indicated in the introduction, this improvement in the characterization of the acoustic properties of cranial bone is necessary for the development of non-invasive transcranial focused ultrasound surgery.

Non-invasive transcranial focused ultrasound surgery requires precise *a priori* prediction of the distortion introduced into the acoustic field by the presence of the skull; this distortion cannot be determined without a thorough knowledge of the geometry of the skull and the acoustic properties of that skull. The distribution of the speed of sound in the skull is the most important of these acoustic properties since it most affects the phase of the transmitted ultrasound wave.

The modeling process described in this paper makes certain assumptions about the nature of trabecular orientation in cranial bone, the use of CT data to provide apparent density measurements, and the use of phase measurements as a basis for optimization; these assumptions and the justifications for their use are discussed below. The speed of sound

model produced in this paper is then compared to prior notable work on entirely homogeneous models for the speed of sound in skull bone [22,39,40,45].

4.5.1 TRABECULAR ORIENTATION

The speed of sound model derived in this paper is expressed in the form $c(\rho_{ct})$, which implicitly disregards the direction of propagation of the ultrasound wave. Some studies [30,46,47] have found marked anisotropy in the speed of sound of ultrasound in bone, which would require a model of the form $c(\rho_{ct},\underline{u})$ with $\underline{u}$ as the vectorial direction of propagation. However these studies have been performed in such bones as the calcaneus, long bones of the leg or vertebral bodies – in these cases the bone is clearly placed under a predominant mechanical loading which gives rise to an equivalent orientation in the trabecular lattice. Faster acoustic propagation is found where $\underline{u}$ is parallel to the direction of mechanical loading; this faster propagation velocity is accounted for by acoustic bar wave propagation directly down the trabecular framework and is not strongly correlated to the density of the bone [46]. In contrast, cranial bone is self-evidently not placed under routine directional loading, and so there is little mechanical stimulus towards any particular trabecular orientation [48]. Furthermore, transcranial acoustic propagation places stringent requirements on the range of permissible propagation directions; the high level of acoustic absorption in skull requires that the direction of propagation be close to normal to the surface of the skull to shorten the path in bone and hence permit a sufficiently intense intracranial acoustic field to be developed. As detailed above, the direct wave strongly dominates in transmission through the skull for angles of incidence within 20° of normal [49], and therefore the effect of $\underline{u}$ on the production of the shear mode of acoustic propagation may be neglected.

The model used in the genetic algorithm optimization process also does not include any special treatment of potential ultrasound scattering in trabecular bone. However, Chaffaï et al.[50] have already treated this matter, in which it was found that "at frequencies less than 1MHz (the clinically useful frequencies), no broadening of the transmitted pulse temporal profile, apart from the effect of the frequency dependent

attenuation, has been evidenced" and that "for the time being, experiments do not suggest the presence of strong multiple scattering."

Given these practical constraints on $\underline{u}$ for acoustic use and the non-orientation of the cranial trabecular framework, the $c(\rho_{ct})$ formulation is sufficient.

4.5.2 GENERALIZATION OF APPARENT DENSITY

The model described here is based up on ρ_{ct}, since in patient studies only the apparent density obtained from the CT scan could be expected to be available. However, this leaves the question of whether this apparent density is suitably close to the true density of the material. The use of anatomic specimens of human calvaria in this study allows us to address this question by examining the effect of one further simple constraint on the conversion of Hounsfield units to CT data; we may require that the sum of the apparent densities of all CT points identified as being skull bone multiplied by the volume of a voxel of CT data be equal to the empirically measurable mass of the calvarium. A quadratic conversion between Hounsfield units and skull density can then be established based on this mass requirement, the density of air and the density of water. The correlation between this quadratic conversion and the original linear conversion provides an estimate of the reasonableness of using ρ_{ct} and also allows the $c(\rho_{ct})$ model to be generalized to other methods of acquiring bone density information.

Given a calvarium of mass M, therefore:

$$\begin{aligned} M &= \iiint_{skull} \rho.dV \cong \sum_{skull} \rho.\Delta V \\ \Rightarrow \quad \frac{M}{v} &= \sum_{skull} \rho \end{aligned} \tag{4.8}$$

where v is the volume of a voxel of CT data. Now, using a quadratic conversion of Hounsfield Units (HU):

$$\frac{M}{v} = \sum_{skull} (\kappa_2 HU^2 + \kappa_1 HU + \kappa_0)$$
$$\Rightarrow \quad \frac{M}{v n_{skull}} = \left[\frac{1}{n_{skull}} \sum_{skull} HU^2 \quad \frac{1}{n_{skull}} \sum_{skull} HU \quad 1 \right] \bullet \left[\kappa_2 \quad \kappa_1 \quad \kappa_0 \right] \tag{4.9}$$

where n_{skull} is the number of voxels identified as representing the calvarium, and κ values are parameters of the quadratic conversion. The parameters of the conversion can then be established by solving the following relationship:

$$\begin{bmatrix} \frac{1}{n_{air}} \sum_{air} HU^2 & \frac{1}{n_{air}} \sum_{air} HU & 1 \\ \frac{1}{n_{water}} \sum_{water} HU^2 & \frac{1}{n_{water}} \sum_{water} HU & 1 \\ \frac{1}{n_{skull}} \sum_{skull} HU^2 & \frac{1}{n_{skull}} \sum_{skull} HU & 1 \end{bmatrix} \begin{bmatrix} \kappa_2 \\ \kappa_1 \\ \kappa_0 \end{bmatrix} = \begin{bmatrix} \rho_{air} \\ \rho_{water} \\ \frac{M}{v n_{skull}} \end{bmatrix} \tag{4.10}$$

where the n_{air} and n_{water} are the number of voxels identified as representing air and water. This solution method may be extended over any number of CT scans of different calvaria by adding further lines to the relationship given above using the appropriate M, HU and n_{skull} values in each case. The resulting system of equations is over-determined but may be solved for the best values of the κ parameters using the standard least-squares metric; the rows of the linear system should be normalized to account for the different values of n_{air}, n_{water} and n_{skull} to prevent unacceptable weighting of the solution in the least-squares method. Based on the calvaria used in this study we find κ parameters of $[\kappa_2\ \kappa_1\ \kappa_0] = [-8.837\text{x}10^{-5}\ 1.092\ -36.8]$, which is very nearly linear over our region of interest.

4.5.3 LIMITS ON PHASE-BASED INFERENCE OF SPEED OF SOUND MODELS

Since this speed of sound model is based on the optimization of phase shift information, it is appropriate to enquire whether this model might be skewed by this method. Phase shift information may only take values between zero and 2π radians since phase is periodic by definition. Consequently it is not possible to distinguish a phase shift of ϕ

radians from a phase shift of $\phi + 2\pi$ radians since they are obviously identical modulo 2π. This is a problem of wrapping in phase.

If we consider the assigning of a particular speed of sound to a particular local density, this problem of phase wrapping might occur if the true speed of sound for that local density was sufficiently different from the speed of sound in the model. We can estimate the limits of this problem by calculating the size of error in the speed of sound required to produce this phase wrapping uncertainty. Given an ultrasound frequency of f, a material thickness of x and a speed of sound of c_o, the phase shift ϕ across the material would be expected to be:

$$\phi = \frac{2\pi fx}{c_0} \tag{4.11}$$

The limits we seek are the increase and decrease in c_0 necessary to produce a phase shift of π in the predicted phase.

$$\begin{aligned} \phi + \pi &= \frac{2\pi fx}{c_0 - \Delta c_{low}} \\ \phi - \pi &= \frac{2\pi fx}{c_0 + \Delta c_{high}} \end{aligned} \tag{4.12}$$

By re-arrangement, the following limits can therefore be established.

$$\begin{aligned} \Delta c_{low} &= c_0 - \frac{2c_0 fx}{2fx + c_0} \\ \Delta c_{high} &= \frac{2c_0 fx}{2fx - c_0} - c_0 \end{aligned} \tag{4.13}$$

Thus, if the model speed of sound is c_0, but the true speed of sound is $c_0 + \Delta c_{high}$, this will produce the same phase-shift information as if the true speed of sound were $c_0 - \Delta c_{low}$. Fortunately these limits are wide enough, as shown in Figure 4.10, that this problem does not occur in practice in this optimization.

4.5.4 COMPARISON TO OTHER PRIOR WORK

Bone, in its cortical and trabecular forms, may be considered to be a form of cellular solid. Cellular solids are those materials that are composed of a material lattice, that latticework forming either open or closed pores. [41]

The most straightforward model for speed of sound in a material is:

$$c = \sqrt{\frac{E}{\rho}} \tag{4.14}$$

where E is the Young's modulus of the material, and ρ is its density. However, for open-pore cellular materials it can be shown that $E \propto \rho^2$ and that for closed-pore materials $E \propto \rho^3$ [41].

This curve of speed of sound against bone density obtained in our study has a form in keeping with these theoretical material properties of skull. Initially it has a form similar to that of a square root function of speed against density for an open-celled porous solid, as would be the case for low volume fraction trabecular bone. The model transitions into a linear region in keeping with the material structure of dense cortical bone. The transitional region is between 1665 kg m^{-3}, the upper limit for the density of bulk trabecular bone based upon the 70% bone volume fraction requirement, and 1950 kg m^{-3}, the density for the bulk of cortical bone.

Using the values of ρ_{ct} for the calvaria in this study, the distribution of c in each calvarium can be calculated based on the model we have described. An appropriate average of the distribution of c provides a basis for comparison with pre-existing measurements of the speed of sound in bone.

The most appropriate average for the distribution of c is not the simple arithmetic mean of c. If we consider a very simple model for the propagation of a wave through some heterogeneous material, the time taken for the wave to travel through the material t is:

$$t = \int \frac{1}{c(x)} \delta x \tag{4.15}$$

Now if the material has thickness x then the average speed of sound c_{av} through the material becomes:

$$c_{av} = \frac{x}{t} = x.\left(\int \frac{1}{c(x)} \delta x \right)^{-1} \tag{4.16}$$

and so:

$$c_{av} = \left(\overline{c^{-1}} \right)^{-1} \tag{4.17}$$

where $\overline{c^{-1}}$ is the arithmetic mean of the distribution of the reciprocal of c. Calculating for the four calvaria here, bulk c_{av} speeds of sound of 2609.9ms^{-1}, 2884.8 ms^{-1}, 2829.9ms^{-1} and 2756.8ms^{-1} can be found. These values are entirely in keeping with the range of values seen in the current literature [22,39,40,45].

4.6 Conclusion

The Genetic Algorithm optimization method coupled to acoustic simulation and experimental phase measurements provides a useful method for determining the variation in speed of sound in cranial bone with density. The model produced in this paper using this method has been shown to predict the speed of sound in transcranial propagation better than pre-existing homogenized speed of sound methods. Finally, it must be cautioned that the model in this paper is based on measurements of the acoustic properties of a limited number of formalin-preserved human cadaveric skulls and therefore it cannot be safely assumed without further study that the model is representative of the entire population of human skulls or of *in vivo* transcranial propagation.

4.7 REFERENCES

[1] F. Fry and L. Johnson, Tumor irradiation with intense ultrasound *Ultrasound in Medicine and Biology*, vol. 4, pp. 337-411, 1978.

[2] P.P. Lele and A.D. Pierce, The thermal hypothesis of the mechanism of ultrasonic focal destruction in organised tissues. Interaction of ultrasound and biological tissues. FDA 73-8008 BRH/DBE, pp. 121-128, 1973. Bureau of Radiological Health. Washington, DC.

[3] F.L. Lizzi, D.J. Coleman, J. Driller, M. Ostromogilsky, S. Chang, and P. Greenall, Ultrasonic hyperthermia for Ophthalmic Therapy *IEEE Trans.Sonics Ultras.*, vol. SU-31, pp. 473-481, 1984.

[4] J.Y. Chapelon, P. Faure, M. Plantier, D. Cathignol, R. Souchon, F. Gorry, and A. Gelet, The feasibility of tissue ablation using high intensity electronically focused ultrasound *IEEE Ultrasonics Symp.*, vol. 93CH3301-9, pp. 1211-1214, 1993.

[5] N.T. Sanghvi, F. Fry, R. Bihrle, F.S. Foster, M.H. Phillips, J. Syrus, A.V. Zaitsev, and C.W. Hennige, Noninvasive surgery of prostate tissue by high-intensity focused ultrasound. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 43, pp. 1099-1110, 1996.

[6] G.R. ter Haar, D. Sinnett, and I. Rivens, High intensity focused ultrasound - a surgical technique for the treatment of discrete liver tumors *Phys Med Biol.*, vol. 34, pp. 1743-1750, 1989.

[7] W.J. Fry, W. Mosberg, J.W. Barnard, and F.J. Fry, Production of focal destructive lesions in the central nervous system with ultrasound. *Journal of Neurosurgery*, vol. 11, pp. 471-478, 1954.

[8] P.P. Lele, Production of deep focal lesions by focused ultrasound - current status *Ultrasonics*, vol. 5, pp. 105-122, 1967.

[9] J. Sun and K.H. Hynynen, Focusing of therapeutic ultrasound through a human skull: a numerical study *Journal of the Acoustical Society of America*, vol. 104, pp. 1705-1715, 1998.

[10] K. Hynynen and F.A. Jolesz, Demonstration of potential noninvasive ultrasound brain therapy through an intact skull *Ultrasound in Medicine and Biology*, vol. 24, pp. 275-283, 1998.

[11] M. Tanter, J.L. Thomas, and M. Fink, Focusing and steering through absorbing and aberrating layers: application to ultrasonic propagation through the skull *Journal of the Acoustical Society of America*, vol. 103, pp. 2403-2410, 1998.

[12] J.G. Lynn, R.L. Zwemer, A.J. Chick, and A.E. Miller, A new method for the generation and use of focused ultrasound in experimental biology *J.Gen.Physiol.*, vol. 26, pp. 179-193, 1942.

[13] J.G. Lynn and T.J. Putnam, Histology of cerebral lesions produced by focused ultrasound *Am.J.Path.*, vol. 20, pp. 637-652, 1944.

[14] J. Sun and K.H. Hynynen, The potential of transskull ultrasound therapy and surgery using the maximum available skull surface area *Journal of the Acoustical Society of America*, vol. 105, pp. 2519-2527, 1999.

[15] K. Hynynen, Trans-skull ultrasound therapy: the feasibility of using image-derived skull thickness information to correct the phase distortion *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, pp. 752-755, 1999.

[16] G.T. Clement and K. Hynynen, A noninvasive method for focusing ultrasound through the human skull *Phys Med Biol*, vol. 47, pp. 1219-1236, 2002.

[17] M. Tanter, J.F. Aubry, J. Gerber, J.L. Thomas, and M. Fink, Optimal focusing by spatio-temporal inverse filter: I. Basic Principles *Journal of the Acoustical Society of America*, vol. 110, pp. 37-47, 2001.

[18] J.F. Greenleaf, A graphical description of scattering *Ultrasound in Medicine and Biology*, vol. 12, pp. 603-609, 1986.

[19] M.A. Averkiou and R.O. Cleveland, Modeling of an electrohydraulic lithotripter with the KZK equation *Journal of the Acoustical Society of America*, vol. 106, pp. 102-112, 1999.

[20] I.M. Hallaj and R.O. Cleveland, FDTD simulation of finite-amplitude pressure and temperature fields for biomedical ultrasound *Acoust.Res.Lett.Online*, vol. 1, pp. 7-12, 1999.

[21] S. Lees, P.F. Cleary, J.D. Heeley, and E.L. Gariepy, Distribution of sonic plesio-velocity in a compact bone sample *Journal of the Acoustical Society of America*, vol. 66, pp. 641-646, 1979.

[22] F. Fry and J. Barger, Acoustical properties of the human skull *Journal of the Acoustical Society of America*, vol. 63, pp. 1576-1590, 1978.

[23] J.H. Holland. *Adaptation in natural and artificial systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[24] R. Lakes, H.S. Yoon, and J.L. Katz, Ultrasonic wave propagation and attenuation in wet bone *Journal of Biomedical Engineering*, vol. 8, pp. 143-148, 1985.

[25] C. Poloni and V. Pediroda. GA Coupled with Computationally Expensive Simulations: Tools to Improve Efficiency. In: *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, eds. D. Quagliarella, J. Periaux, C. Poloni, and G. Winter. Chichester, England: John Wiley & Sons, 1998.pp. 267-288.

[26] F. Fry, Transskull transmission of axisymmetric focused ultrasonic beams in the 0.5 to 1 MHz frequency range: implications for brain tissue visualization, interrogation, and therapy *Ultrasonic Tissue Characterization*, vol. 2, pp. 203-208, 1979.

[27] G.T. Clement and K. Hynynen, Correlation of ultrasound phase with physical skull properties *Ultrasound in Medicine and Biology*, vol. 28, pp. 617-624, 2002.

[28] J.L. Katz, Hard tissue as a composite material. I. Bounds on the elastic properties *Journal of Biomechanics*, vol. 4, pp. 455, 1971.

[29] S. Lees. Sonic properties of mineralized tissues. In: *Tissue characterization with ultrasound, volume 2*, ed. J.F. Greenleaf. CRC Press, 1986.pp. 207-226.

[30] S. Lees, J.M. Ahern, and M. Leonard, Parameters influencing the sonic velocity in compact calcified tissues of various species *Journal of the Acoustical Society of America*, vol. 74, pp. 28-33, 1983.

[31] R.E. Shaffer and G.W. Small, Genetic algorithms for the optimization of piecewise linear discriminants *Chemometrics and Intelligent Laboratory Systems*, vol. 35, pp. 87-104, 1996.

[32] C. De Boor. *A practical guide to splines*, New York: Springer-Verlag, 1978.

[33] J. Antonisse, A new interpretation of schema notation that overturns the binary encoding constraint, Proceedings of the Third International Conference on Genetic Algorithms, pp. 86-91, 1989.

[34] A.H. Wright. Genetic Algorithms for Real Parameter Optimization. In: *Foundations of Genetic Algorithms*, ed. G.J.E. Rawlins. San Mateo, CA: Morgan Kaufmann Publishers, 1991.pp. 205-218.

[35] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of Genetic Algorithms*, ed. G.J.E. Rawlins. San Mateo, CA: Morgan Kaufmann Publishers, 1991.pp. 69-93.

[36] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. Parallelization of Genetic Programming. In: *Genetic Programming III*, San Francisco, CA: Morgan Kaufmann Publishers, 1999.pp. 1025-1036.

[37] B.K.P. Horn, Closed-form solution of absolute orientation using unit quaternions *Journal of the Optical Society of America A*, vol. 4, pp. 629-642, 1986.

[38] P.J. Westervelt, Parametric Acoustic Array *Journal of the Acoustical Society of America*, vol. 35, pp. 535-537, 1963.

[39] H. Theismann and F. Pfander, Uber die Durchlassigkeit des Knochens fur Ultraschall *Strahlentherapie*, vol. 80, pp. 607-610, 1949.

[40] B. Martin and J.H. McElhaney, The acoustic properties of human skull bone *Journal of Biomedical Materials Research*, vol. 5, pp. 325-333, 1971.

[41] L.J. Gibson and M.F. Ashby. *Cellular solids: structure and properties*, Cambridge University Press, 1997.

[42] W. Abendschein and G.W. Hyatt, Ultrasonics and selected physical properties of bone *Clinical Orthopaedics and Related Research*, vol. 69, pp. 294-301, 1970.

[43] D.T. Blackstock, Thermoviscous attenuation of plane, periodic, finite-amplitude sound waves *Journal of the Acoustical Society of America*, vol. 36, pp. 534-542, 1964.

[44] A. Taflove. *Computational Electromagnetics: The finite-difference time-domain method*, Artech House, 1995.

[45] F. Duck. Acoustic properties of tissue at ultrasonic frequencies. In: *Physical properties of tissue: a comprehensive reference book*, Academic Press, London, 1990.

[46] P.H.F. Nicholson, M.J. Haddaway, and M.W.J. Davie, The dependence of ultrasonic properties on orientation in human vertebral bone *Phys Med Biol*, vol. 39, pp. 1013-1024, 1994.

[47] R.N. McCarthy, L.B. Jeffcott, and R.N. McCartney, Ultrasound speed in equine cortical bone: effects of orientation, density, porosity and temperature *Journal of Biomechanics*, vol. 23, pp. 1139-1143, 1990.

[48] D.N. White. The acoustic characteristics of skull. In: *Tissue characterization with ultrasound, volume 2* , ed. J.F. Greenleaf. CRC Press, 1986.pp. 1-39.

[49] M. Hayner and K. Hynynen, Numerical analysis of ultrasonic transmission and absorption of oblique plane waves through the human skull. *Journal of the Acoustical Society of America*, vol. 110, pp. 3319-3330, 2001.

[50] S. Chaffai, V. Roberjot, F. Peyrin, G. Berger, and P. Laugier, Frequency dependence of ultrasonic backscattering in cancellous bone: Autocorrelation model and experimental results *Journal of the Acoustical Society of America*, vol. 108, pp. 2403-2411, 2000.

4.8 TABLES

Table 4.1: Speed of sound in cranial bone as a function of apparent bone density by CT scan in intervals of 100 kg m^{-3}.

ρ_{ct} (kg m^{-3})	c (m s^{-1})	ρ_{ct} (kg m^{-3})	c (m s^{-1})
1000	1852.7	2300	2991.1
1100	2033.0	2400	3142.1
1200	2183.8	2500	3294.6
1300	2275.8	2600	3444.7
1400	2302.6	2700	3588.5
1500	2298.1	2800	3722.2
1600	2300.3	2900	3841.9
1700	2332.5	3000	3947.6
1800	2393.4	3100	4041.9
1900	2479.0	3200	4127.7
2000	2585.4	3300	4207.8
2100	2708.8	3400	4285.1
2200	2845.3		

4.9 FIGURES

Figure 4.1: A human calvarium mounted in a stereotactic positioning frame.

Figure 4.2: Examples from the population of potential speed-of-sound models.

Figure 4.3: A computer reconstruction of a human calvarium showing its positioning within a hemispherical transducer.

Figure 4.4: Block diagram of the genetic algorithm optimization of speed-of-sound model.

Figure 4.5: The final model for speed of sound in cranial bone based on apparent density by CT scan.

Figure 4.6: The probabilistic distribution of the phase correlation metric for randomly assigned phases.

Figure 4.7: Statistical significance curve for the phase correlation metric.

Figure 4.8: Comparison of the performance of the linear speed model and the Genetic Algorithm speed model for a skull not used in the optimization process.

Figure 4.9: CT cross-section showing an instance of complex lacunar structure.

Figure 4.10: Speed of sound limits in phase based optimization.

Figure 4.1

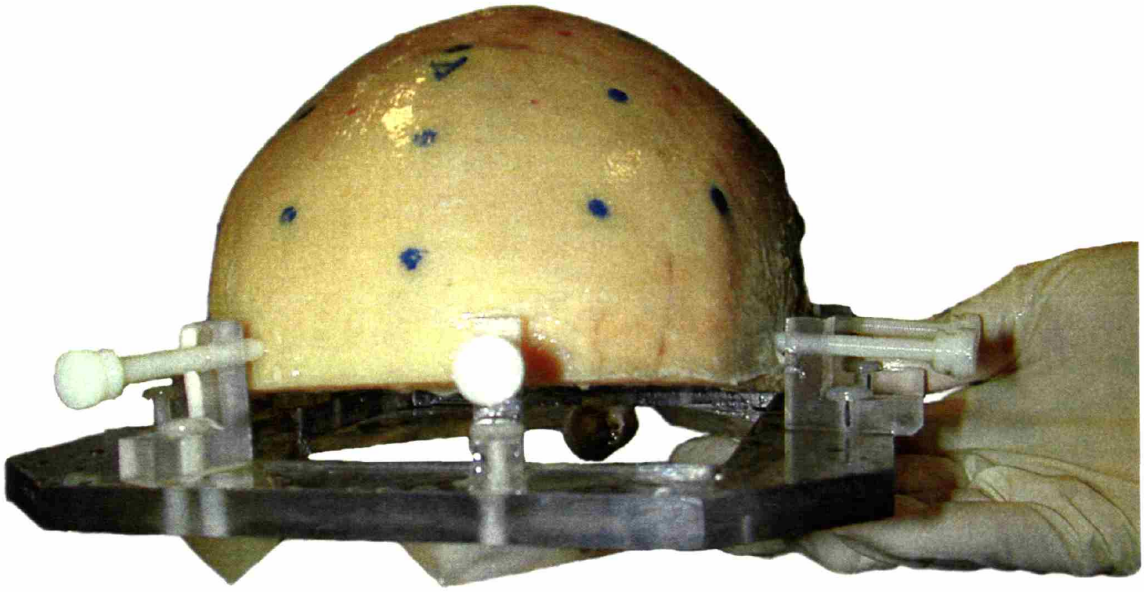
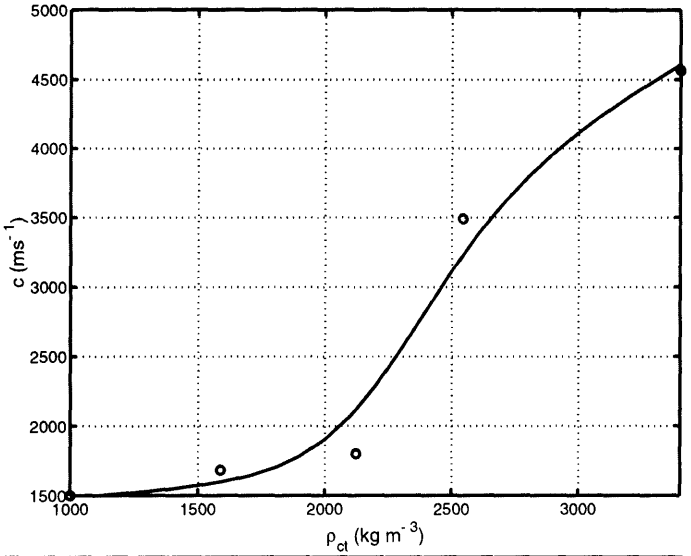


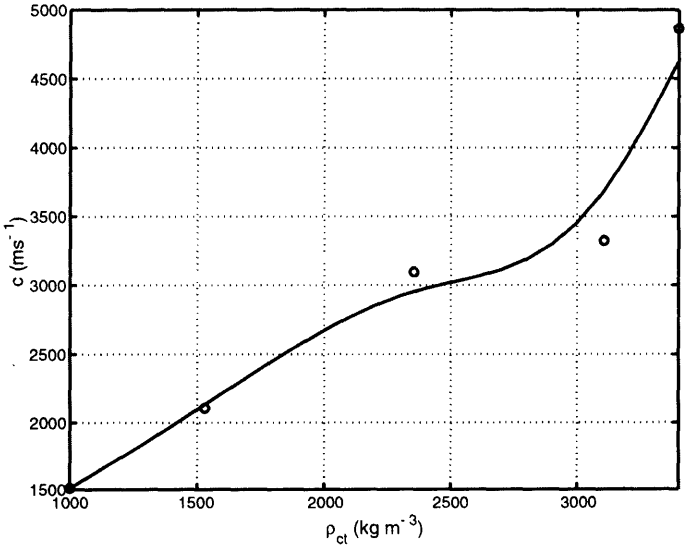
Figure 4.2

(a)



ρ_{ct}	1000.0	1588.5	2123.2	2543.4	3400.0
c	1500.0	1681.5	1799.9	3489.7	4566.4

(b)



ρ_{ct}	1000.0	1529.3	2354.4	3106.0	3400.0
c	1509.1	2105.4	3093.7	3322.3	4864.9

Figure 4.3

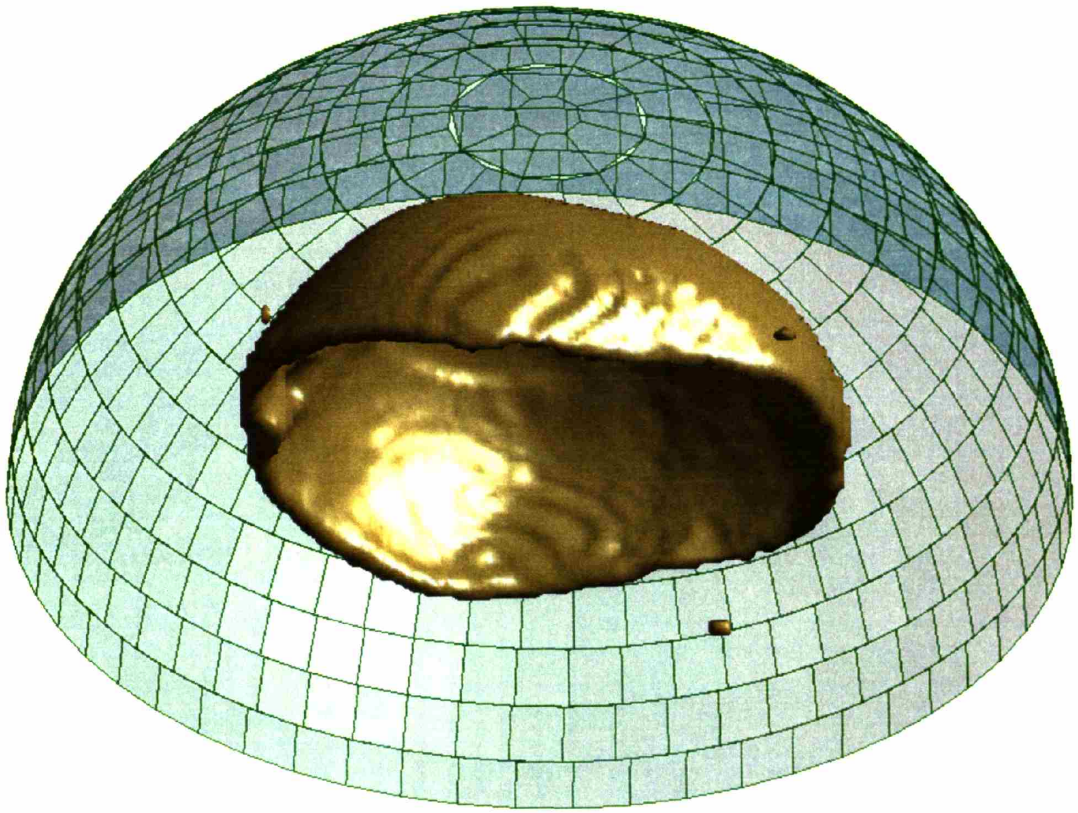


Figure 4.4

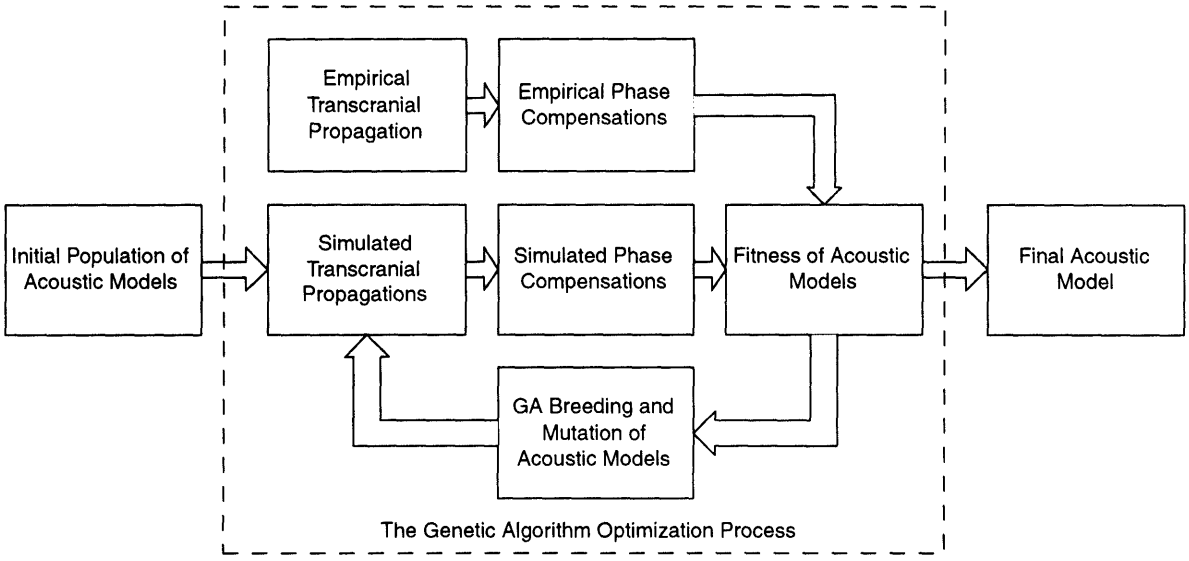
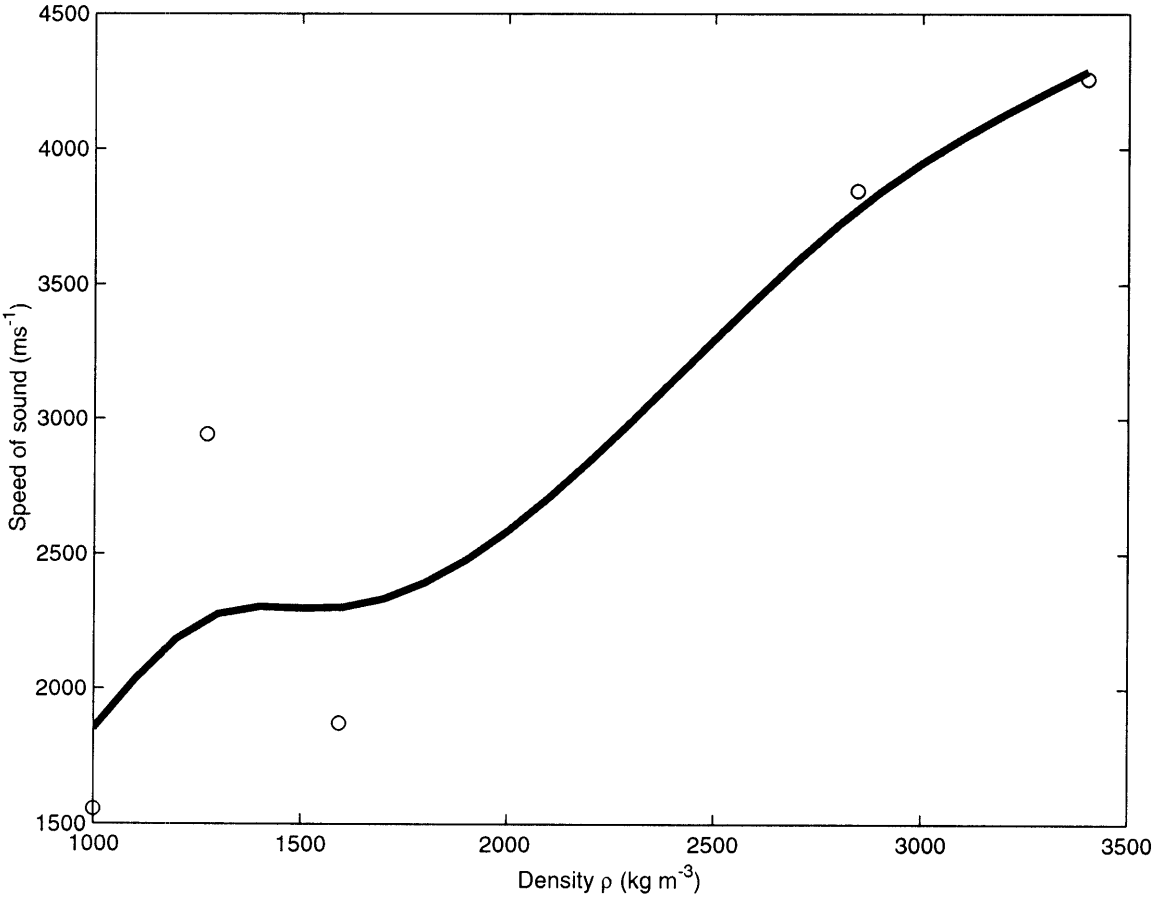

Empirical Transcranial Propagation
Empirical Phase Compensations
Initial Population of Acoustic Models
Simulated Transcranial Propagations
Simulated Phase Compensations
Fitness of Acoustic Models
Final Acoustic Model
GA Breeding and Mutation of Acoustic Models
The Genetic Algorithm Optimization Process

Figure 4.5



ρ_{ct}	1000.0	1272.0	1592.4	2844.5	3400.0
c	1554.9	2941.2	1871.7	3845.2	4256.4

Figure 4.6

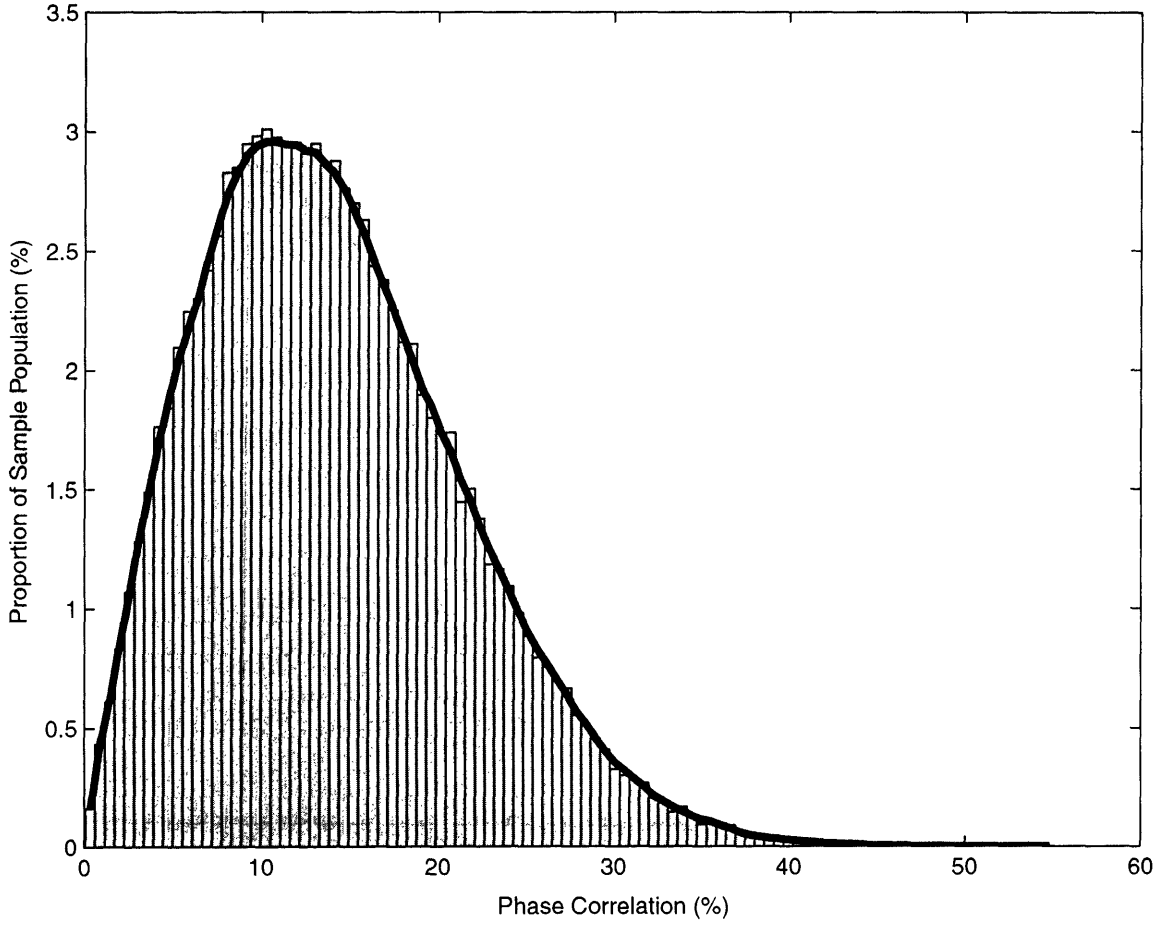

3.5
3
2.5
2
1.5
1
0.5
0
Proportion of Sample Population (%)
0
10
20
30
40
50
60
Phase Correlation (%)

Figure 4.7

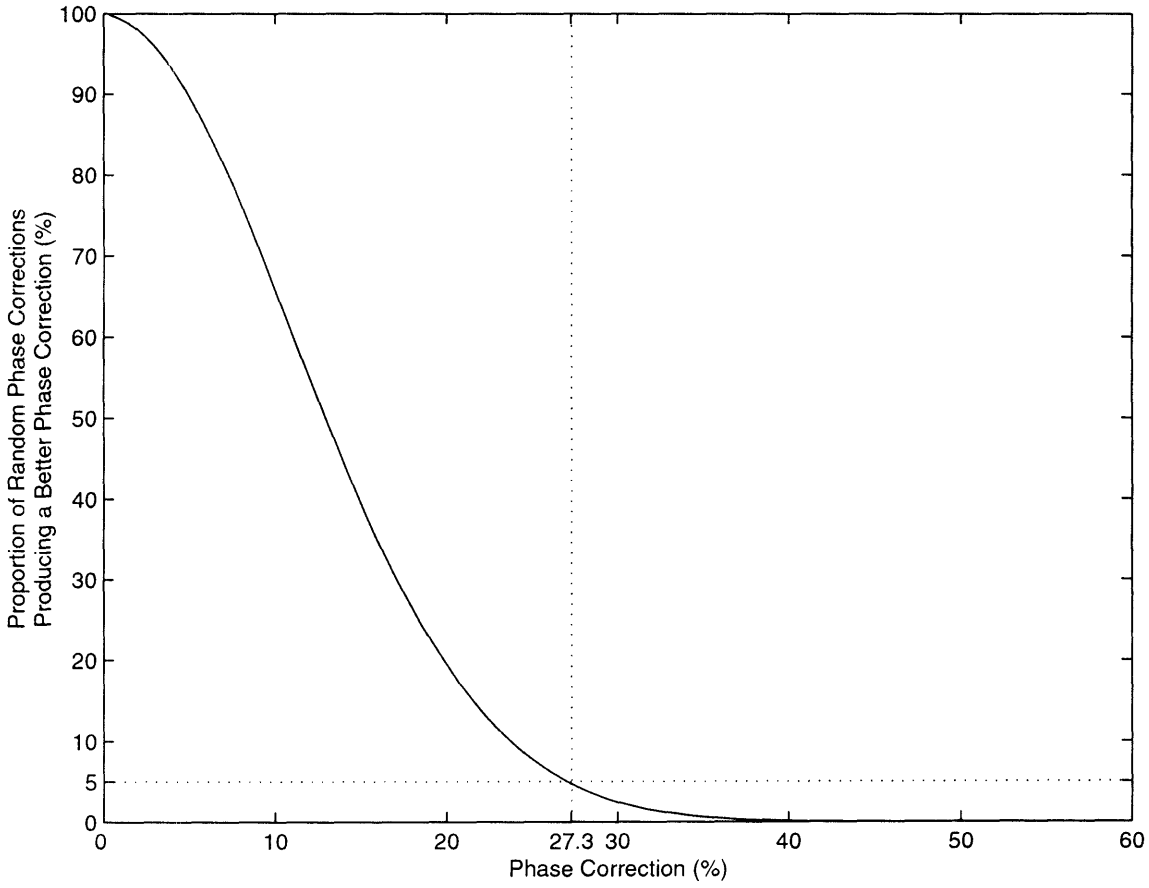

Proportion of Random Phase Corrections
Producing a Better Phase Correction (%)
100
90
80
70
60
50
40
30
20
10
5
0
0
10
20
27.3
30
40
50
60
Phase Correction (%)

Figure 4.8

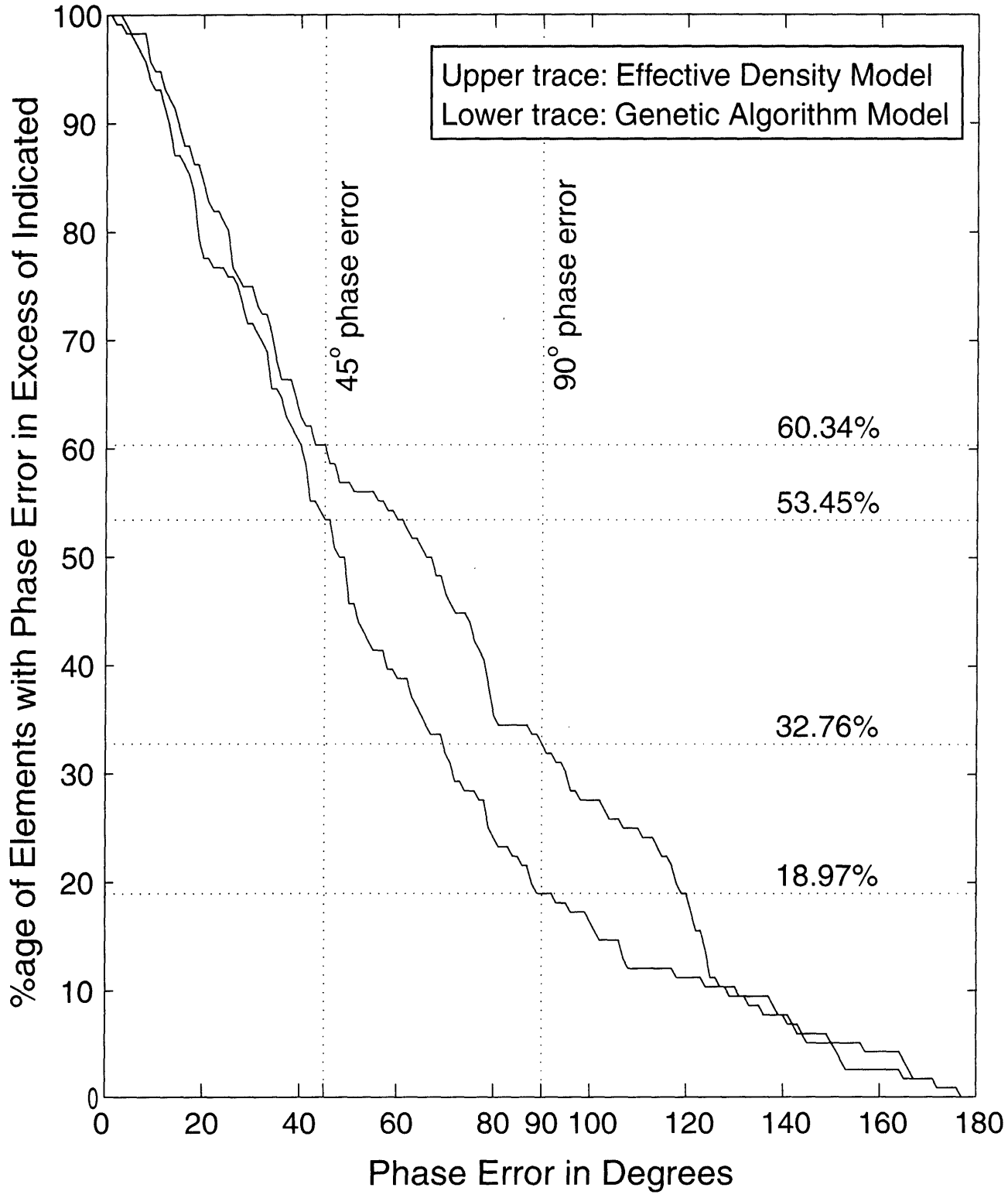

Upper trace: Effective Density Model
Lower trace: Genetic Algorithm Model
45° phase error
90° phase error
60.34%
53.45%
32.76%
18.97%
%age of Elements with Phase Error in Excess of Indicated
Phase Error in Degrees
0
10
20
30
40
50
60
70
80
90
100
0
20
40
60
80
90
100
120
140
160
180

Figure 4.9

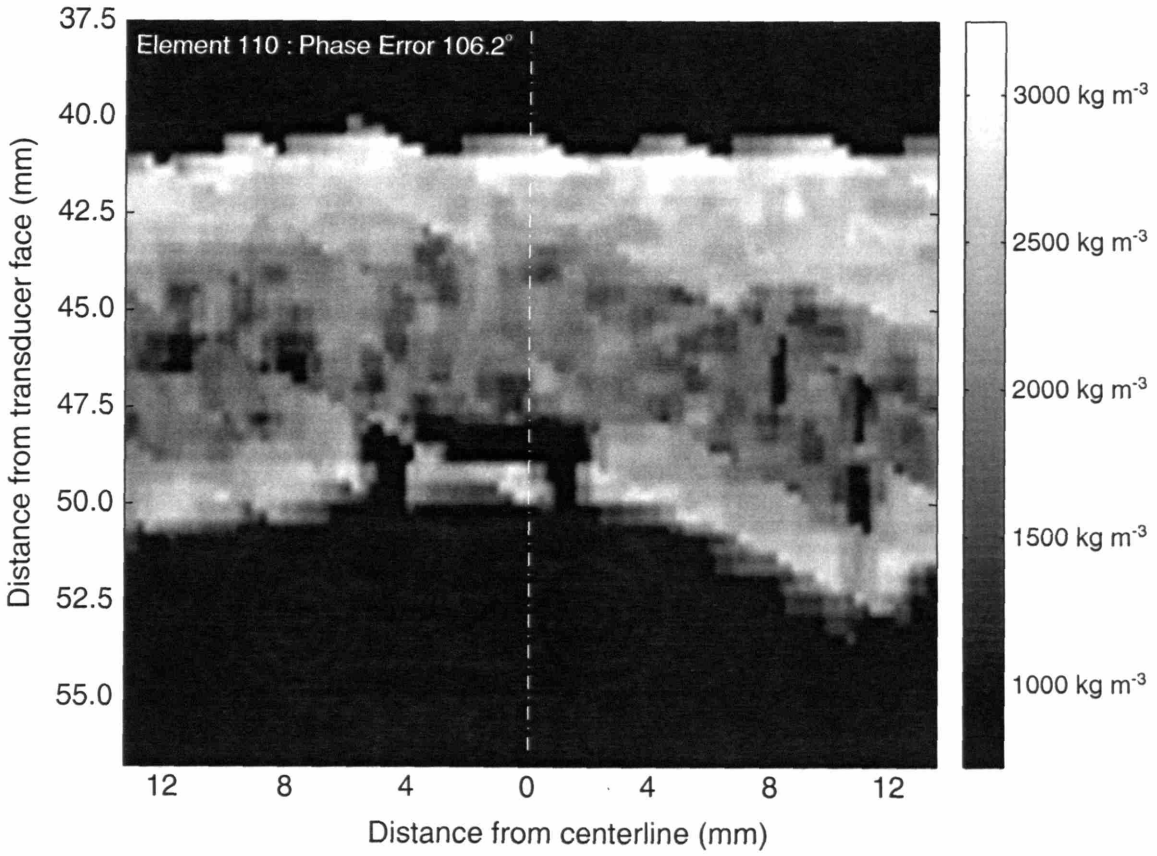
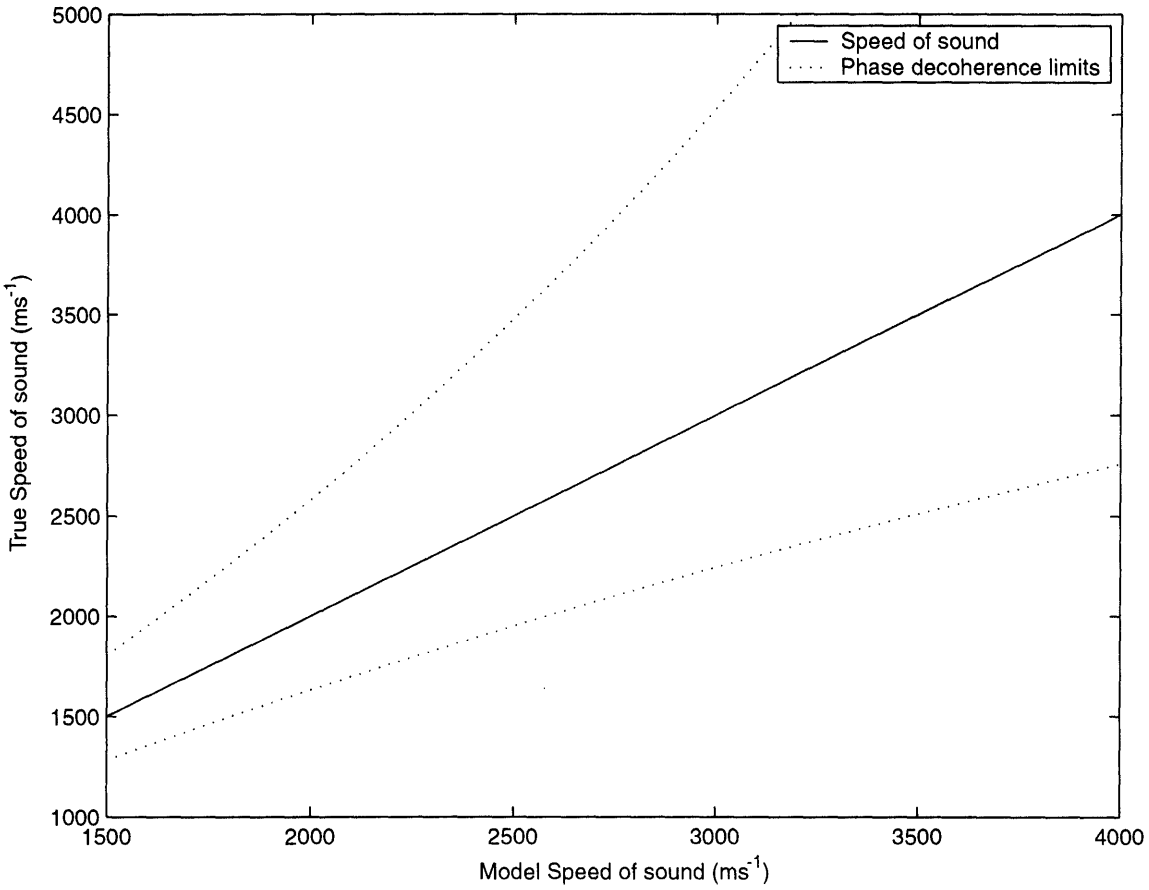

Element 110 : Phase Error 106.2°
Distance from transducer face (mm)
37.5
40.0
42.5
45.0
47.5
50.0
52.5
55.0
12
8
4
0
4
8
12
Distance from centerline (mm)
3000 kg m-3
2500 kg m-3
2000 kg m-3
1500 kg m-3
1000 kg m-3

Figure 4.10


Speed of sound
Phase decoherence limits
True Speed of sound (ms-1)
Model Speed of sound (ms-1)
5000
4500
4000
3500
3000
2500
2000
1500
1000
1500
2000
2500
3000
3500
4000

Chapter 5

A Unified Model For The Variation in Amplitude Attenuation Coefficient In Cranial Bone With Local Density and Ultrasound Frequency Based On Genetic Algorithm Optimization.

The work presented in this chapter has been submitted to *Physics in Medicine and Biology* as:

Connor, Clement and Hynynen, "A Unified Model For The Variation in Amplitude Attenuation Coefficient In Cranial Bone With Local Density and Ultrasound Frequency Based On Genetic Algorithm Optimization", (47), pp. 3925-3944.
The empirical measurements of transcranial acoustic amplitude change were performed by Greg Clement

5.1 ABSTRACT

The density and structure of bone is highly heterogeneous, causing wide variations in the reported acoustic attenuation coefficient for ultrasound propagation. However, this parameter is very important for establishing dosimetry in focused ultrasound treatment of the brain through an intact skull. Firstly, the acoustic energy transmitted through the skull into the brain is dependent on the acoustic attenuation in the skull. Secondly, acoustic absorption in the skull causes skull heating; this places a practical upper limit on the acoustic power that can be used for transcranial therapy *in vivo*. Current research on the propagation of high intensity focused ultrasound through an intact human skull for non-invasive therapeutic action on brain tissue therefore requires a detailed model for the acoustic attenuation coefficient in cranial bone. Such models have been difficult to derive empirically due to the aforementioned heterogeneity of bone itself.

We propose a model for the acoustic attenuation coefficient in cranial bone based upon the apparent density of bone determined by CT scan and upon ultrasound frequency. This model is based upon the coupling of empirical measurement, theoretical acoustic simulation and genetic algorithm optimization. The change in acoustic pressure amplitude caused by the presence of skull in an acoustic path is empirically measured. The ability of a theoretical acoustic simulation coupled with a particular acoustic attenuation model to predict this amplitude change is compared against the empirical data, thus providing the fitness function needed to perform genetic algorithm optimization. By performing genetic algorithm optimization over an initial population of amplitude attenuation coefficient models, an ultimate model for the amplitude attenuation coefficient at a particular ultrasound frequency in both the cortical and trabecular regions of cranial bone is produced. The range of acoustic frequencies studied is from 500kHz to 1MHz, which covers the window of useful frequencies for transcranial ultrasound thermal therapy, bounded at the lower end by wavelength and bounded at the upper end by absorption in the cranial bone.

The final model has a non-linear dependency of acoustic attenuation upon local bone density. Notably, the attenuation tends to be high in low density trabecular bone, decreases to a minimum attenuation in bone of intermediate porosity and then increases with density in higher density cortical bone. The final model was also used to produce an apparent bulk attenuation coefficient for skull across the frequency range studied here; a linear relationship between bulk attenuation coefficient and frequency is found.

5.2 INTRODUCTION

Focused ultrasound provides a non-invasive method for inducing temperature elevations in tumors [1-7], resulting in the coagulation of the tumor. It has long been recognized that the ability to produce a high intensity ultrasound focus within the brain, and hence non-invasive cautery at that focus due to acoustic absorption by brain tissue, would provide a minimally invasive means of treating deep-seated brain tumors without disturbing overlying structures [8,9]. It is clearly desirable to produce this focus through an intact skull rather than requiring craniotomy to establish an acoustic window, which has led to interest in the study of the propagation of ultrasound through the bones of the skull [10-12]. Currently, most effort in this field has been applied to studying the variation of speed of sound in the skull to allow for the correction of the distortion introduced into the acoustic field by the presence of the skull [10,13-15]. However, the acoustic attenuation coefficient also varies within the skull and this parameter is important for transcranial therapeutic procedures. If the acoustic attenuation coefficient is underestimated then less acoustic energy will be delivered to the targeted tissue than predicted and the heating in the skull will be greater than predicted – in effect, a double penalty is paid for this error. Overestimation of the amplitude pressure attenuation coefficient results in greater delivery of acoustic energy to the targeted tissue than predicted with a consequent risk of overtreatment. However, there is a poor consensus on the value of the amplitude attenuation coefficient for the bulk material of bone in the literature, and the treatment of attenuation with frequency in bone is sparse. [16-20]

Consequently, we require a model of local acoustic attenuation in bone based on the apparent bone density at a particular point and on the frequency of the ensonicating ultrasound field. However, such a model cannot be produced by direct empirical measurement due to the very heterogeneity of the material under study. We propose that this problem may be made tractable by using an empirical acoustic approach combined with the genetic algorithm optimization [21] of a population of candidate functions relating apparent bone density and frequency to acoustic attenuation coefficient.

5.3 METHOD

5.3.1 OVERVIEW

The acoustic properties of a material in bulk may be measured by examining the transmission of ultrasound through a sample of that material, and this technique may be used to examine the bulk properties of a heterogeneous material such as bone [22]. In this study, we concentrate on the amplitude difference between the received wave and the transmitted wave in continuous wave sonication. In particular we examine the change in this amplitude between the case where only degassed water is present between the transducer and receiver and the case where a human calvarium is placed between the transducer and the receiver.

The human calvaria used empirically above are first CT scanned at high clinical resolution. Once the overall geometric structure and the density distribution of the calvaria have been acquired, the experiment described above can be reproduced as a computational simulation of acoustic propagation. The change in amplitude measurement described above is calculated for the simulation. This measurement is very sensitive to the attenuation coefficient in the bone. The problem then becomes one of optimizing the model used for amplitude attenuation coefficient in bone in the simulation.

This optimization problem is suited to treatment with a genetic algorithm since there is a well-defined, quantifiable success function, which is the total squared error deviation of the simulated amplitudes from the empirical amplitudes. Furthermore the genetic algorithm optimization technique does not depend on gradient information that may be unacceptably noisy since it would only be available by numerical approximation in this case. This optimization method is also relatively insensitive to local minima and maxima. The genetic algorithm optimization technique can be easily tailored to known physical constraints such that only physically realistic models are permitted [23].

5.3.2 EMPIRICAL CONFIGURATION

Four *ex vivo* human calvaria were used to produce the initial data for genetic algorithm optimization. The skulls chosen represent the range of skull thicknesses found in humans. These skull sections were fixed and stored in a 10% buffered formaldehyde solution. Based on a previous study [18], it was assumed that the attenuation properties of the fixed skulls would be similar to that of fresh skulls. The calvaria were each mounted in a stereotactic frame, as shown in Figure 5.1, and CT scanned using a Siemens Somatom Plus 4 with deconvolution kernel AH-82. Fiducial markers on the frame allow the position and orientation of the skull during the experimental procedure to be accurately matched to a solid body geometric transformation of the reconstructed data.

Attenuation measurements through *ex vivo* human skull sections were acquired in a tank filled with degassed and deionized water and padded with anechoic rubber. A 0.2 mm-diameter Polyvinylidene difluoride (PVDF) hydrophone (Precision Acoustics, Dorchester, UK) was placed in the tank at the geometric center of a focused 8 cm-diameter ultrasonic array, with a 15 cm radius of curvature. This source transducer consisted of an air-backed PZT 1-3 composite material a center frequency of 0.9 MHz electroded into 9 individual elements (Imasonic, Besancon, France). Transducer signals to a specific element were generated by an arbitrary waveform generator (Wavetek, California, Model 305) fed to a power amplifier (ENI, New York, Model 2100L). The hydrophone's voltage response was sent though a Precision Acoustics pre-amp before it was recorded by a digital oscilloscope (Textronix, Oregon, Model 380). The voltage waveform was downloaded to a PC by GPIB control and the amplitude at the measurement location was determined from the FFT of the transducer driving frequency.

Waveforms resulting from the excitation of a single element were recorded both before and after an *ex vivo* human calvarium was placed in the ultrasound propagation path, allowing the signal attenuation to be determined. To avoid interference from reflections between the skull and the transducer surface, the signal was limited to a 30-cycle burst repeated at a rate of 50ms. The intersection point between the skull and the ultrasound

beam was controlled by a rotational stepping motor (Parker, Pennsylvania), which was mounted to the bone sample through its stereotactic frame. This single rotational degree of freedom limited the measurements to points along a fixed elevation line. Data were acquired along a sequence of spatial locations by rotating each calvarium over 84 degrees and measuring the transmitted waveform at 7 degree intervals. Three different elevation lines were considered on each skull by separately exciting an element at the top, center and bottom of the transducer array, measured relative to their positions along the rotational axis. At each measurement location, the frequency was also varied to provide information on the attenuation level as a function on frequency. The frequencies sampled were 500 kHz, 650 kHz, 750 kHz, 850 kHz and 1 MHz; this frequency range covers the useful acoustic window in cranial bone for transcranial focused ultrasound surgery [10,24]. A separate waveform was recorded and evaluated for each frequency step. Above this frequency window, sufficient acoustic scattering occurs such that this empirical technique would only be suitable for measuring acoustic insertion loss rather than attenuation. However, Chaffaï *et al.* [20] found that for:

> "ultrasound transmission through bone at frequencies less than 1 MHz (the clinically useful frequencies) , no broadening of the transmitted pulse temporal profile, apart from the effects of the frequency-dependent attenuation, has been evidenced. Furthermore, it has been argued that, would multiple scattering be present, its impact on measurements could be decreased dramatically by the strong attenuation encountered in trabecular bone. For the time being, experiments do not suggest the presence of strong multiple scattering."

The overall experimental configuration is shown in schematically in Figure 5.2. Given the number of skulls, rotational angles and active transducer elements, an initial optimization data set of 156 acoustic paths per frequency band was produced.

5.3.3 GENETIC ALGORITHM REPRESENATION

A model for the amplitude attenuation coefficient in bone can be determined by admitting an initial large pool of candidate attenuation coefficient models, and ascertaining how well these simulations predict the amplitude changes as established by empirical study. These attenuation coefficient models are then used to produce a new generation of attenuation coefficient models that are similarly studied. However, it is required that these models be stated in a form amenable to manipulation by genetic algorithms.

Bone may be considered to be a composite of inorganic mineral within an organic framework. The mineral component is predominantly hydroxyapatite, which may be present in a crystalline form with a density of up to 3170 kg m^{-3} [25]. Type I collagen contributes 90% of the organic material and, like most organic materials, has a density similar to that of water [26].

The genetic algorithm representation of the attenuation function is as five "knots" arranged on a graph of bone density against attenuation coefficient. The density of water sets the lower bound on density. The upper bound is set at 3400 kg m^{-3}, which is above the maximum possible density of 3170 kg m^{-3}; optimization techniques may perform poorly if the maximum of the optimization is located in a corner of the search space, and this small margin potentially alleviates that risk. The attenuation coefficient is constrained to not be negative at any density. No formal upper bound is set on the attenuation coefficient because there is little practical computational reason to do so. Furthermore, previously reported values for the attenuation coefficient in cranial bone have been given as bulk properties and we must allow for the possibility that the attenuation coefficient is some small portion of the tissue may be much greater.

The continuous model of the speed of sound can now be represented in the discrete allelic form required for this method of optimization [23,27]. The function $\alpha(\rho,f)$, the model speed of sound dependent on the local apparent bone density for a particular ultrasound frequency, is generated from these allelic knots by minimizing [28]:

$$\upsilon.\sum_{i=1}^{5}\left[\alpha_{i,f}-\alpha(\rho_i,f)\right]^2+(1-\upsilon).\int_{\rho_{lo}}^{\rho_{hi}}\left(\frac{d^2\alpha(\rho,f)}{d\rho^2}\right)^2 d\rho \qquad (5.1)$$

where $\alpha_{i,f}$ and $\rho_{i,f}$ are the positions of the allelic knots at frequency f. ρ_{lo} and ρ_{hi} are the lower and upper bounds on density. υ is a numeric value between 0 and 1 that balances the relative importance of fitting the knots closely with the requirement that the function $\alpha(\rho,f)$ be reasonably smooth. This operation is part of the class of cubic smoothing splines.

This avoids the well-known problem with polynomial function fitting where the knots are perfectly fit but the function displays unacceptable hilliness in between the knots. Selecting a value of $\upsilon = 0.99$ places strong emphasis on fitting the knots (the first part of the equation) while providing acceptably smooth and responsive fitting in the regions between.

A quantitative success function is required to measure the fitness of each particular acoustic attenuation model. This success function is calculated by determining the root mean square error between the empirically measured amplitude of the transmitted wave and the simulated amplitude of the transmitted wave:

$$\text{GA Success Function} = 100\% \times \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{\widehat{p_s^{(i)}}}{\widehat{p_w^{(i)}}}-\frac{p_s^{(i)}}{p_w^{(i)}}\right)^2} \qquad (5.2)$$

where n is the number of acoustic paths in a particular frequency band, $p_s^{(i)}$ is the empirical amplitude of ultrasound transmitted through skull on acoustic path i, $p_w^{(i)}$ is the empirical amplitude of ultrasound transmitted through water on acoustic path i, and $\widehat{p_s^{(i)}}$ and $\widehat{p_w^{(i)}}$ are the simulated counterparts of $p_s^{(i)}$ and $p_w^{(i)}$. The genetic algorithm is therefore configured to minimize this function, which is the two-norm of the error.

The alleles are stored as floating point numbers hence the genetic algorithm is of the real-coded type. Most original work on the robustness of genetic algorithms was performed on binary coded algorithms. However, by extension of the demonstration of the robustness

of genetic algorithms over non-binary alphabets [29], real-coded algorithms have also been shown to perform in accordance with Holland's schemata [21,30].

Each generation of models is evolved from the previous one by the three steps of competitive tournament selection, breeding based on allelic crossover, and the application of spontaneous mutation [31].

Two-thirds of the time a random mutation is added to the alleles of a particular model, with a permissible mutation range of ± 100 kg m^{-3} and ± 15 Np m^{-1}. While this mutation rate would be high for binary-coded algorithms where a single bit mutation can produce a huge variation in the numerical value it represents, it is appropriate for real-coded genetic algorithms that are more robust in this sense. This higher level of small mutations improves the efficiency of the genetic algorithm in searching for local minima when the algorithm is close to convergence. Otherwise, the genetic algorithm optimization for attenuation coefficient proceeds in a similar manner to the method described in Connor *et al.* [32] for optimizing the speed of sound function in cranial bone.

Any models in this new generation that are not physically valid by the criteria given above are deleted, and the second and third steps above are repeated until a complete new generation is produced. Once this is achieved, parallel processing is used to increase the speed of the genetic algorithm optimization; many simulations are performed concurrently over a network of computers [33].

Genetic algorithm optimization is a very robust and global optimization technique, but it does not guarantee that the optimization result is locally minimum. To rectify this potential weakness, the final model produced by genetic algorithm optimization for a particular ultrasound frequency is then optimized by Sequential Quadratic Programming (SQP) [34,35]. The overall optimization process is illustrated in Figure 5.3. An example of the convergence of this genetic algorithm is shown in Figure 5.4.

5.3.4 SIMULATION OF ACOUSTIC PROPAGATION

It is vitally important that the acoustic path that is simulated match the acoustic path studied empirically. Each experimental calvaria is mounted in a stereotactic frame to ensure that it can be rigidly and reproducibly positioned in the hemispherical array described in section 5.3.2. The positions of the fiducial markers on the frames can be measured relative to the hemispherical ultrasound transducer using a three-dimensional Cartesian calibrated positioning system. By identifying these same fiducial markers in the CT data, it is then possible to determine the geometric solid body transformation from the orientation of the skull in the CT data set to the orientation of the skull in the experimental apparatus. The optimal solid body transformation can be determined in closed form using Horn's algorithm [36].

An extra subtlety is introduced by the use of the rotator to provide increased acoustic paths for simulation. The axis of rotation is fixed parallel to the z-axis of the calibrated positioning system, and hence the complete transformation from orientation in the CT scan dataset to orientation in the experimental apparatus is given by:

$$p_{\text{cps}} = \left[\begin{array}{ccc|c} \cos\theta & -\sin\theta & 0 & x_r(1-\cos\theta)+y_r\sin\theta \\ \sin\theta & \cos\theta & 0 & -x_r\sin\theta+y_r(1-\cos\theta) \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \mathbf{M}_{\text{cps}\leftarrow\text{CT}} p_{\text{CT}} \qquad (5.3)$$

where $\mathbf{M}_{\text{cps}\leftarrow\text{CT}}$ is the solid body transformation from orientation in the CT dataset and orientation in the calibrated position system in classical augmented matrix form, θ is the angle of rotation through which the rotator is turned, and (x_r,y_r) defines the position of the rotator axis in the calibrated positioning system.

As the bones of the skull are heterogeneous, an algorithm is required that is capable of simulating acoustic propagation in a material with rapidly changing local acoustic properties. Acoustic propagation is simulated using a finite difference time domain (FDTD) algorithm based on the following formulation of the Westervelt [37] equation:

$$\frac{\partial^2 p}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3} + \frac{2\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \frac{\partial p}{\partial z}\cdot\frac{\partial(\ln \rho)}{\partial z} = 0 \qquad (5.4)$$

where p is the pressure of the acoustic wave, ρ is the local density, δ is the local acoustic diffusivity, β is the co-efficient of non-linearity, c is the local speed of sound.

The value of the non-linear parameter β is set to zero, effectively simulating linear acoustic propagation. The pressure amplitude of the ultrasound used empirically is sufficiently low that non-linear acoustic effects are negligible; the Goldberg number Γ is also expected to be small due to the high attenuation α[38]. Furthermore, the value of β in bone has not yet been satisfactorily determined.

For harmonic excitation, $\delta = \frac{2ac^3}{(2\pi f)^2}$ where α is the acoustic amplitude attenuation co-efficient and f is the frequency of the ultrasound wave. As stated in section 5.3.2, here f may take values of 500kHz, 650kHz, 750kHz, 850kHz or 1MHz to harness the skull acoustic window. However, we can remove this explicit dependence on frequency from the finite difference algorithm by observing that $\omega = 2\pi f$ where ω is angular frequency, and that for a purely harmonic excitation $\frac{\partial^3 p}{\partial t^3} \equiv -\omega^2 \frac{\partial p}{\partial t}$. Substituting into Equation (5.4) gives:

$$\frac{\partial^2 p}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \frac{2\alpha}{c}\frac{\partial p}{\partial t} + \frac{2\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \frac{\partial p}{\partial z}\cdot\frac{\partial(\ln \rho)}{\partial z} = 0 \qquad (5.5)$$

Discretization parameters Δt and Δz are chosen to ensure numerical stability in the simulation. This requirement is enforced by requiring the Courant-Friedrichs-Levy (CFL) number to be less than or equal to unity at all points, where $\mathrm{CFL} = c\Delta t/\Delta z$ [39].

The local density at a particular point is generated from the CT scan of the calvarium. CT scans return data as Hounsfield Unit (*HU*) rather than as density. This problem can be overcome by ensuring that a sample of both water and air appear in the CT scan. Since

the densities of water and air can be reasonably taken to be 1000 kg m^{-3} and 0.0 kg m^{-3} respectively, a linear normalization relation can be produced where:

$$\kappa_1 = \frac{1}{\overline{H_{water}} - \overline{H_{air}}} \quad ; \quad \kappa_0 = \frac{-\overline{H_{air}}}{\overline{H_{water}} - \overline{H_{air}}} \tag{5.6}$$

and thus for any particular point we state that:

$$\rho_{ct} = \kappa_1 HU + \kappa_0 \tag{5.7}$$

where ρ_{ct} is the apparent local density based on the CT scan in kg m^{-3}, and κ_1 and κ_0 are constants with units of kg m^{-3} HU^{-1} and kg m^{-3} respectively.

5.3.5 HOMOGENIZING CT DATA AND ATTENUATION COEFFICIENTS

The genetic algorithm presented here is based on a one-dimensional formulation of the Westervelt equation as detailed in Section 5.3.4. Ideally the genetic algorithm optimization would be conducted with a full three-dimensional simulation of acoustic propagation. However, even using suitable distributed computing techniques, to perform the optimization in this way with an equivalent population of speed models and number of generations might be expected to have taken several years with current computing power. Consequently, a full three-dimensional treatment is not possible at this time. However, it is possible to analyze the effects of averaging the density of the cranial bone under sonication or of homogenizing the attenuation model presented here.

The distribution of ultrasound intensity in the focal plane of one of the transducer elements used in this study can be calculated numerically using the Huygens-Fresnel integral, as shown in Figure 5.5. This allows the half-power beam width to be calculated. In this case, a 3dB reduction in beam intensity occurs 8.33mm off-axis. A representative average of skull density can then be determined by taking the average density of points within a radius of 8.33mm perpendicular to the axis of propagation for each point along the ultrasound path as it passes through the skull.

The attenuation models presented here may also be homogenized to a single value representative of a bulk attenuation property. A simple acoustic model for transmission of a wave through an absorbent material in a water bath is:

$$\frac{p}{p_0} = \frac{4Z_w Z_s}{\left(Z_w + Z_s\right)^2} . e^{-\alpha_{bulk} d} \tag{5.8}$$

where Z_w is the acoustic impedance of water, d is the thickness of the material, α_{bulk} is the acoustic attenuation parameter of the material, and Z_s is the acoustic impedance of the material. Z_w is 1.5 MRayls and Z_s is approximately 5 MRayls [19]. p_0 is the initial pressure amplitude of the wave and p is the amplitude of the transmitted wave.

However, if the attenuation coefficient is now allowed to vary along the beam path, the expression becomes:

$$\frac{p}{p_0} = \frac{4Z_w Z_s}{\left(Z_w + Z_s\right)^2} . e^{\int_0^d -\alpha(\rho(x))\,dx} \tag{5.9}$$

By examining these expressions, it can be inferred that taking the mean of the attenuation coefficient function as applied to the densities of cranial bone is an appropriate operator for producing a bulk attenuation coefficient. Hence, simply:

$$\alpha_{bulk} \approx \overline{\alpha\left(\rho_{skull}\right)} \tag{5.10}$$

5.3.6 ASSESSING THE PERFORMANCE OF ACOUSTIC ATTENUATION MODELS

The root mean square error function described in section 5.3.3 is useful for controlling genetic optimization because it penalizes all deviations from the empirically measured values. For instance, the algebraic mean error would be a poor choice for a GA success function. If a particular model were to overestimate some attenuation values but underestimate others, these errors might cancel and nevertheless produce a respectably small algebraic mean error. The algebraic mean error (or one-norm) is therefore a poor guide for optimization. However, for the purposes of transcranial ultrasound surgery, the most important property of an acoustic attenuation model is its ability to predict the total

intracranial pressure field from the contributions of many active transducer elements. Therefore the algebraic mean error, defined as:

$$100\% \times \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\widehat{p_s^{(i)}}}{\widehat{p_w^{(i)}}} - \frac{p_s^{(i)}}{p_w^{(i)}} \right) \tag{5.11}$$

where n, $\widehat{p_s^{(i)}}$, $\widehat{p_w^{(i)}}$, $p_s^{(i)}$ and $p_w^{(i)}$ are as defined in Section 5.3.3, is the most appropriate test of the performance of the attenuation model for clinical use.

5.4 RESULTS

Ultimately, after 45 generations, the models shown in Figure 5.6 and Figure 5.7 were produced for the variation in amplitude attenuation coefficient against ultrasound frequency and apparent bone density by CT scan. These models are stated in numerical form in Table 5.1 and Table 5.2.

5.4.1 ROBUSTNESS OF THE ACOUSTIC ATTENUATION MODELS.

The final acoustic attenuation models for 500 kHz and 650 kHz had root mean square errors of less than 1.5%. The final models for 750 kHz, 850 kHz and 1 MHz had root mean square error values less than 1%.

The analysis of a finite number of experimental calvaria is used in this paper to produce a general model for acoustic attenuation in skull. This problem is therefore of the class of problems where an optimization process is used to produce a generally applicable result from a pool of empirical data. Therefore, it must be established that the genetic algorithm has general applicability by using it in the analysis of ultrasound propagation through a calvarium that was not used in the optimization procedure.

A new calvarium was placed in the same experimental apparatus as described in section 5.3.2 above. The amplitude changes due to transcranial propagation of the ultrasound were measured over the same range of frequencies and orientations, such that a total of 195 acoustic paths were measured through this new calvarium. Table 5.3 shows the comparison of algebraic mean error by frequency band for the four skulls used in the original genetic algorithm training data and for this new calvarium. These values are sufficiently comparable in range that the risk of substantial overtraining in the genetic optimization process can be discounted.

5.4.2 COMPARISON OF THIS ALGORITHM TO ALGORITHMS BASED ON HOMOGENIZATION OF DENSITY AND ATTENUATION

Using the methods described in Section 5.3.5, a bulk attenuation parameter can therefore be generated for each skull at each frequency; these bulk parameters are shown in Table 5.4.

Table 5.5 shows the results for the genetic algorithm model for attenuation on CT data as described above as well as the results produced by applying the genetic algorithm to homogenized CT data and the results produced by applying the homogenized bulk alpha properties to homogenized CT data. Applying the genetic algorithm models directly to homogenized CT data produced very poor results as the act of homogenizing the CT data tends to average away much information about the structure of the cranial bone. This homogenization does not match well to the genetic algorithm model, which is specifically based on actual densities, not average densities. The situation is uniformly improved by using the homogenized attenuation properties with the homogenized CT data, but the results in this case are still substantially poorer than are produced by the genetic optimization models for attenuation based directly on density. The performance of these three modeling techniques independent of frequency is shown graphically in Figure 5.8. This figure also includes the performance of Fry and Barger's [18] classical attenuation model for skull. The x-axis specifies an error in predicted acoustic amplitude as detailed in Equation (5.11), and the y-axis shows the proportion of propagations that have an error greater or equal than this. Consequently, a better performing model is represented by a curve that trends towards the bottom-left corner of the graph [13,32]. Therefore, based on the calvaria used in this study, the genetic algorithm optimization models for attenuation based on local density substantially outperform the homogenized attenuation models.

5.5 Discussion

This paper describes a novel method for establishing a model of the acoustic amplitude attenuation coefficient in cranial bone over a range of frequencies. An important difference from previous work is that this method produces a model for the attenuation of sound at any particular point in the skull; the model is not dependent on dividing the cranial bone into regions or layers or on ascribing homogenized properties to any part of the skull. This model is valid for cranial bone of both cortical and trabecular structure, and is based on a coupled strategy of genetic algorithm optimization of CT-based acoustic simulation and empirical measurement of the change in amplitude of an ultrasound wave on transmission through human calvaria. The results of this method are given in Table 5.1 and Table 5.2, and shown graphically in Figure 5.6 and Figure 5.7.

By definition, skull bone is present in the prefocal region of any non-invasive transcranial focused ultrasound surgery procedure. Heating of skull bone is potentially a limiting factor in this treatment modality due to the high value of α in skull bone. As indicated in the introduction, this limit is especially susceptible to a high value of α in skull as the thermal deposition in the skull is proportional to α, and because a high value of α in skull requires that an ultrasound field of greater intensity be used to produce a therapeutically useful intracranial acoustic field.

The modeling process described in this paper makes certain assumptions about the nature of trabecular orientation in cranial bone and the use of CT data to provide apparent density measurements. It is assumed that there is no overall axis of organization in cranial trabecular bone, given that cranial bone is not weight-bearing, and that accurate density measurements can be extracted from CT data. In practice, these assumptions are reasonable; their justifications are treated in substantial detail in Connor *et al.* [32].

Prior research on acoustic attenuation in bone has concentrated on producing a bulk attenuation coefficient [16-20]. Therefore, to compare the genetic algorithm models to previously published data, an operation is required to convert the apparent density

measurements of cranial bone and the acoustic attenuation models presented here into a single bulk attenuation property. Such an operation has already been described in Section 5.4.2, and these equivalent bulk attenuation properties are given in Table 5.4. These results strongly support a linear dependence of attenuation on frequency over the range used in this study, as shown in Figure 5.9. Prior work on the bulk ultrasound attenuation property of skull in the low megahertz range agrees with this outcome [17,18,40]. There is substantial variation in previously published attenuation coefficients for cranial bone, but the attenuation coefficients produced here are broadly comparable. Fry *et al.* [18] gives values for 167 Np m^{-1} MHz^{-1} and 215 Np m^{-1} MHz^{-1} for the inner and outer layers of cortical bone, and Hueter [17] gives an attenuation of 52 Np m^{-1} for an ultrasound frequency of 600 kHz. It should be noted that Martin *et al.* [19] presented values for attenuation in cranial bone that were around an order of magnitude higher, but it has been since noted that the cranial bone samples used in that study were ~¼λ in thickness causing resonant effects to mask the true attenuation [18]. It must be cautioned that the models in this paper are based on measurements of the acoustic properties of a limited number of formalin-preserved human cadaveric skulls and therefore it cannot be directly assumed without further study that these models are appropriately representative of the entire population of human skulls or of *in vivo* transcranial propagation.

White [40] noted that acoustic attenuation in trabecular bone increases more dramatically with ultrasound frequency than in attenuation in cortical bone. This accounts for the marked increase in attenuation in cranial bone above 1.5MHz [17]. Bone, in its cortical and trabecular forms, may be considered to be a form of cellular solid. Cellular solids are those materials that are composed of a material lattice, that latticework forming either open or closed pores [41]. The acoustic properties of cellular materials depend on the nature of these pores. For materials of intermediate porosity, the pores can only be considered to be open or closed in relation to the wavelength of the ultrasound propagating through the material. Connor *et al.* [32] determined that at an ultrasound frequency of 750kHz, a transition from trabecular to cortical behavior (i.e. from open-pore to closed-pore behavior) in the speed of sound in cranial bone occurred at 1950 kg m^{-3}. This density value approximates the trough in the model for attenuation

coefficient at this frequency. We hypothesize that the value of ρ that gives the minimum of the $\alpha(\rho)$ curve for a particular ultrasound frequency f marks the point at which bone can be said to have transitioned from trabecular to cortical form in regard of its acoustic properties.

This relationship between the minimum of the $\alpha(\rho)$ curve and the ultrasound frequency appears to be linear over the range of frequencies studied here. Let $\rho_{\min}$ be this transitional density at the point at which $\alpha(\rho)$ is minimum for some frequency f. By linear regression, it can be established that:

$$\rho_{\min} = 1468 + 7.745 \times 10^{-4} f \tag{5.12}$$

where f is in hertz, as shown in Figure 5.10. The correlation coefficient r for this relationship is 0.947, which is statistically significant with $p<0.02$. This transition density increases with frequency since increased density corresponds to decreased pore size in the trabecular lattice and increased frequency corresponds to decreased ultrasound wavelength in the cranial bone. This relationship provides a means to differentiate trabecular and cortical bone for acoustic applications.

5.6 Conclusion

A Genetic Algorithm optimization technique based on empirical measurement and acoustic simulation can be used to determine the acoustic attenuation characteristics of cranial bone based on ultrasound frequency and local bone density. The models produced in this paper using this method are necessary for more accurate simulation of transcranial acoustic propagation, and also serve to elucidate the changes in the acoustic behavior of bone as it transitions from trabecular to cortical form. Furthermore, these models are necessary for accurate prediction of the level of thermal deposition of acoustic energy in the skull during non-invasive transcranial focused ultrasound surgery.

5.7 REFERENCES

[1] F. Fry and L. Johnson, Tumor irradiation with intense ultrasound *Ultrasound in Medicine and Biology*, vol. 4, pp. 337-411, 1978.

[2] P.P. Lele and A.D. Pierce, The thermal hypothesis of the mechanism of ultrasonic focal destruction in organised tissues. Interaction of ultrasound and biological tissues. FDA 73-8008 BRH/DBE, pp. 121-128, 1973. Bureau of Radiological Health. Washington, DC.

[3] F.L. Lizzi, D.J. Coleman, J. Driller, M. Ostromogilsky, S. Chang, and P. Greenall, Ultrasonic hyperthermia for Ophthalmic Therapy *IEEE Trans.Sonics Ultras.*, vol. SU-31, pp. 473-481, 1984.

[4] J.Y. Chapelon, P. Faure, M. Plantier, D. Cathignol, R. Souchon, F. Gorry, and A. Gelet, The feasibility of tissue ablation using high intensity electronically focused ultrasound *IEEE Ultrasonics Symp.*, vol. 93CH3301-9, pp. 1211-1214, 1993.

[5] N.T. Sanghvi, F. Fry, R. Bihrle, F.S. Foster, M.H. Phillips, J. Syrus, A.V. Zaitsev, and C.W. Hennige, Noninvasive surgery of prostate tissue by high-intensity focused ultrasound. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 43, pp. 1099-1110, 1996.

[6] G.R. ter Haar, D. Sinnett, and I. Rivens, High intensity focused ultrasound - a surgical technique for the treatment of discrete liver tumors *Phys Med Biol.*, vol. 34, pp. 1743-1750, 1989.

[7] K. Hynynen, O. Pomeroy, D.N. Smith, P.E. Huber, N. McDannold, J. Kettenbach, J. Baum, S. Singer, and F.A. Jolesz, MR imaging-guided focused ultrasound surgery of fibroadenomas in the breast: a feasibility study. *Radiology*, vol. 219, pp. 176-185, 2001.

[8] W.J. Fry, W. Mosberg, J.W. Barnard, and F.J. Fry, Production of focal destructive lesions in the central nervous system with ultrasound. *Journal of Neurosurgery*, vol. 11, pp. 471-478, 1954.

[9] P.P. Lele, Production of deep focal lesions by focused ultrasound - current status *Ultrasonics*, vol. 5, pp. 105-122, 1967.

[10] J. Sun and K.H. Hynynen, Focusing of therapeutic ultrasound through a human skull: a numerical study *Journal of the Acoustical Society of America*, vol. 104, pp. 1705-1715, 1998.

[11] K. Hynynen and F.A. Jolesz, Demonstration of potential noninvasive ultrasound brain therapy through an intact skull *Ultrasound in Medicine and Biology*, vol. 24, pp. 275-283, 1998.

[12] M. Tanter, J.L. Thomas, and M. Fink, Focusing and steering through absorbing and aberrating layers: application to ultrasonic propagation through the skull *Journal of the Acoustical Society of America*, vol. 103, pp. 2403-2410, 1998.

[13] G.T. Clement and K. Hynynen, Correlation of ultrasound phase with physical skull properties *Ultrasound in Medicine and Biology*, vol. 28, pp. 617-624, 2002.

[14] G.T. Clement and K. Hynynen, Field characterization of therapeutic ultrasound phased arrays through forward and backward planar projection *Journal of the Acoustical Society of America*, vol. 108, pp. 441-446, 2000.

[15] K. Hynynen and J. Sun, Trans-skull ultrasound therapy: the feasibility of using image-derived skull thickness information to correct the phase distortion *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, pp. 752-755, 1999.

[16] H. Theismann and F. Pfander, Uber die Durchlassigkeit des Knochens fur Ultraschall *Strahlentherapie*, vol. 80, pp. 607-610, 1949.

[17] T.F. Hueter, Messung der Ultraschallabsorption in Tierischen Geweben and ihre Abhangigkeit von der Frequenz *Naturwissenschaften*, vol. 39, pp. 21-22, 1952.

[18] F. Fry and J. Barger, Acoustical properties of the human skull *Journal of the Acoustical Society of America*, vol. 63, pp. 1576-1590, 1978.

[19] B. Martin and J.H. McElhaney, The acoustic properties of human skull bone *Journal of Biomedical Materials Research*, vol. 5, pp. 325-333, 1971.

[20] S. Chaffai, F. Padilla, G. Berger, and P. Laugier, In vitro measurement of the frequency-dependent attenuation in cancellous bone between 0.2 and 2MHz *Journal of the Acoustical Society of America*, vol. 108, pp. 1281-1289, 2000.

[21] J.H. Holland. *Adaptation in natural and artificial systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[22] R. Lakes, H.S. Yoon, and J.L. Katz, Ultrasonic wave propagation and attenuation in wet bone *Journal of Biomedical Engineering*, vol. 8, pp. 143-148, 1985.

[23] C. Poloni and V. Pediroda. GA Coupled with Computationally Expensive Simulations: Tools to Improve Efficiency. In: *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, eds. D. Quagliarella, J. Periaux, C. Poloni, and G. Winter. Chichester, England: John Wiley & Sons, 1998.pp. 267-288.

[24] F. Fry, Transskull transmission of axisymmetric focused ultrasonic beams in the 0.5 to 1 MHz frequency range: implications for brain tissue visualization, interrogation, and therapy *Ultrasonic Tissue Characterization*, vol. 2, pp. 203-208, 1979.

[25] J.L. Katz, Hard tissue as a composite material. I. Bounds on the elastic properties *Journal of Biomechanics*, vol. 4, pp. 4551971.

[26] S. Lees. Sonic properties of mineralized tissues. In: *Tissue characterization with ultrasound, volume 2*, ed. J.F. Greenleaf. CRC Press, 1986.pp. 207-226.

[27] R.E. Shaffer and G.W. Small, Genetic algorithms for the optimization of piecewise linear discriminants *Chemometrics and Intelligent Laboratory Systems*, vol. 35, pp. 87-104, 1996.

[28] C. De Boor. *A practical guide to splines*, New York: Springer-Verlag, 1978.

[29] J. Antonisse, A new interpretation of schema notation that overturns the binary encoding constraint, Proceedings of the Third International Conference on Genetic Algorithms, pp. 86-91, 1989.

[30] A.H. Wright. Genetic Algorithms for Real Parameter Optimization. In: *Foundations of Genetic Algorithms*, ed. G.J.E. Rawlins. San Mateo, CA: Morgan Kaufmann Publishers, 1991.pp. 205-218.

[31] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of Genetic Algorithms*, ed. G.J.E. Rawlins. San Mateo, CA: Morgan Kaufmann Publishers, 1991.pp. 69-93.

[32] C.W. Connor, G.T. Clement, and K. Hynynen, A unified model for the speed of sound in cranial bone based on genetic algorithm optimization *Phys Med Biol*, vol. 47, pp. 3925-3944, 2002.

[33] J.R. Koza, F.H. Bennett, D. Andre, and M.A. Keane. Parallelization of Genetic Programming. In: *Genetic Programming III*, San Francisco, CA: Morgan Kaufmann Publishers, 1999.pp. 1025-1036.

[34] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*, London: Academic Press, 1981.

[35] M.J.D. Powell. *Mathematical Programming: The State Of The Art*, Springer-Verlag, 1983.

[36] B.K.P. Horn, Closed-form solution of absolute orientation using unit quaternions *Journal of the Optical Society of America A*, vol. 4, pp. 629-642, 1986.

[37] P.J. Westervelt, Parametric Acoustic Array *Journal of the Acoustical Society of America*, vol. 35, pp. 535-537, 1963.

[38] D.T. Blackstock, Thermoviscous attenuation of plane, periodic, finite-amplitude sound waves *Journal of the Acoustical Society of America*, vol. 36, pp. 534-542, 1964.

[39] A. Taflove. *Computational Electromagnetics: The finite-difference time-domain method*, Artech House, 1995.

[40] D.N. White. The acoustic characteristics of skull. In: *Tissue characterization with ultrasound, volume 2* , ed. J.F. Greenleaf. CRC Press, 1986.pp. 1-39.

[41] L.J. Gibson and M.F. Ashby. *Cellular solids: structure and properties*, Cambridge University Press, 1997.

5.8 TABLES

Table 5.1: The genetic algorithm form of the relationships between acoustic attenuation coefficient, apparent bone density and ultrasound frequency.

Table 5.2: The acoustic attenuation coefficient in cranial bone as a function of ultrasound frequency and apparent bone density by CT scan in intervals of 100 kg m^{-3}.

Table 5.3: Comparison of Genetic Algorithm training data to Genetic Algorithm test data.

Table 5.4: Equivalent bulk attenuation coefficient parameters for cranial bone based on the functions produced by genetic algorithm optimization.

Table 5.5: Comparison of the performance of the alpha model based on density to homogenized models.

Table 5.1

Frequency	Model Solution In Genetic Algorithm Form					
500 kHz	ρ_{ct} (kg m^{-3})	1000.0	1759.9	2325.4	3349.3	3400.0
	α (Np m^{-1})	218.2	0.0	71.8	87.6	142.8
650 kHz	ρ_{ct} (kg m^{-3})	1000.0	1711.6	2461.6	3027.1	3400.0
	α (Np m^{-1})	347.1	55.6	139.3	148.1	162.1
750 kHz	ρ_{ct} (kg m^{-3})	1000.0	2126.5	2416.8	2717.6	3400.0
	α (Np m^{-1})	521.2	50.8	123.7	209.3	114.8
850 kHz	ρ_{ct} (kg m^{-3})	1000.0	1443.3	2054.8	2440.0	3400.0
	α (Np m^{-1})	529.7	360.3	41.1	182.7	222.1
1000 kHz	ρ_{ct} (kg m^{-3})	1000.0	1542.2	1760.2	1991.2	3400.0
	α (Np m^{-1})	578.6	472.8	168.2	166.2	567.7

Table 5.2

ρ_{ct} (kg m^{-3})	$\alpha_{500\ kHz}$ (Np m^{-1})	$\alpha_{650\ kHz}$ (Np m^{-1})	$\alpha_{750\ kHz}$ (Np m^{-1})	$\alpha_{850\ kHz}$ (Np m^{-1})	$\alpha_{1000\ kHz}$ (Np m^{-1})
1000	208.9	334.5	512.1	533.1	592.0
1100	175.9	288.9	455.7	488.6	558.1
1200	143.9	244.5	400.1	443.7	522.8
1300	113.6	202.7	346.4	398.2	484.9
1400	86.2	164.5	295.4	351.7	443.0
1500	62.4	131.4	248.0	303.9	395.8
1600	43.3	104.4	205.1	256.1	342.3
1700	29.6	85.0	167.7	210.9	286.5
1800	22.4	73.9	136.5	170.7	236.9
1900	21.5	70.4	112.6	138.1	198.0
2000	25.4	73.0	96.8	115.7	170.2
2100	32.8	80.0	90.0	105.9	153.1
2200	42.0	90.2	92.9	108.2	145.9
2300	51.6	101.9	103.8	118.7	148.0
2400	60.2	113.6	119.7	133.1	158.5
2500	67.5	124.0	137.6	147.5	176.8
2600	73.8	132.2	154.8	160.2	201.9
2700	79.3	138.6	168.2	171.5	233.2
2800	84.3	143.6	175.5	181.4	269.9
2900	89.1	147.5	176.6	190.3	311.2
3000	93.8	150.8	172.5	198.1	356.3

Table 5.3

Frequency	GA Training Data	GA Test Data
500 kHz	5.86%	4.58%
650 kHz	3.52%	-0.09%
750 kHz	1.97%	-1.18%
850 kHz	1.31%	-2.19%
1000 kHz	0.34%	-2.43%

Table 5.4

α_{bulk} (Np m^{-1})	500 kHz	650 kHz	750 kHz	850 kHz	1000 kHz
Skull A	52.9	111.5	170.4	201.0	267.2
Skull B	50.9	105.4	140.4	162.0	219.9
Skull C	55.4	110.7	144.1	164.0	221.5
Skull D	51.1	106.6	146.9	170.2	227.2
Skull E	54.8	110.7	149.7	171.9	231.7
Mean	53.0	109.0	150.3	173.8	233.5

Table 5.5

Frequency	Original Model Original CT Data	Original Model Homogenized CT Data	Homogenized Model Homogenized CT Data
500 kHz	5.86%	-26.97%	-19.17%
650 kHz	3.52%	-17.95%	-12.13%
750 kHz	1.97%	-12.34%	-8.53%
850 kHz	1.31%	-10.07%	-6.83%
1000 kHz	0.34%	-7.05%	-4.08%

5.9 FIGURES

Figure 5.1: A human calvarium mounted in a stereotactic positioning frame.

Figure 5.2: Schematic diagram of the experimental apparatus.

Figure 5.3: Block diagram of the genetic algorithm optimization of the amplitude attenuation model.

Figure 5.4: Stages of evolution during genetic algorithm optimization of the attenuation coefficient in cranial bone at an ultrasound frequency of 500 kHz.

Figure 5.5: Focal plane of the ultrasound beam with half-power beam width

Figure 5.6: The final models for amplitude attenuation coefficient in cranial bone based on apparent density by CT scan and ultrasound frequency.

Figure 5.7: The relationship between apparent density of cranial bone by CT scan, ultrasound frequency and attenuation coefficient as a three-dimensional surface.

Figure 5.8: Comparison of the performance of the genetic algorithm attenuation model to derived homogenized attenuation models.

Figure 5.9: Graph of bulk attenuation coefficient against ultrasound frequency.

Figure 5.10: Graph of density of minimum attenuation in cranial bone against ultrasound frequency.

Figure 5.1

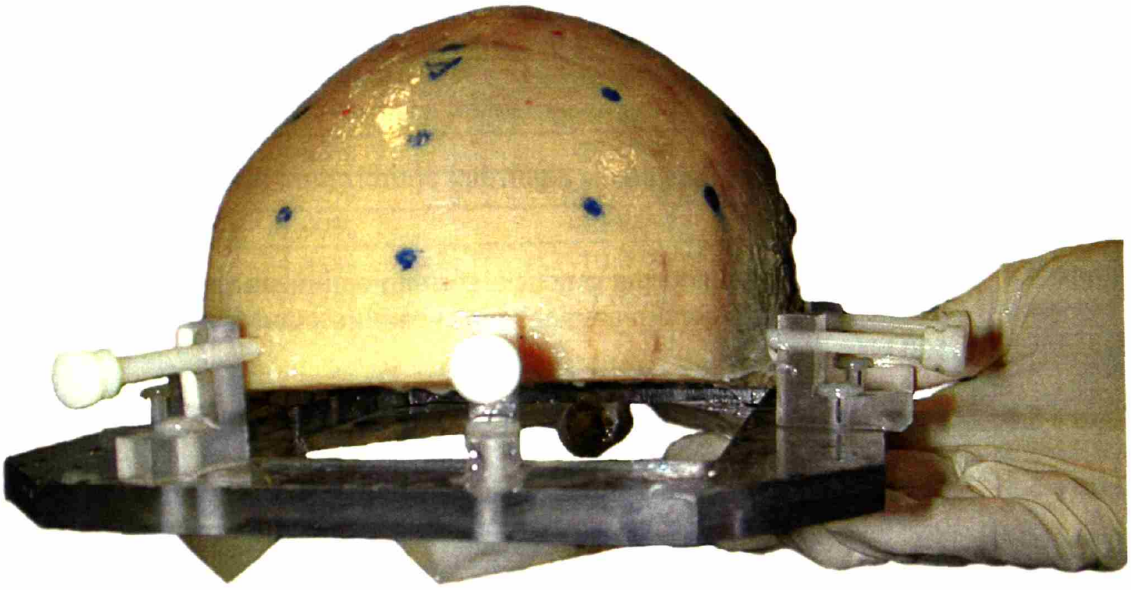


Figure 5.2

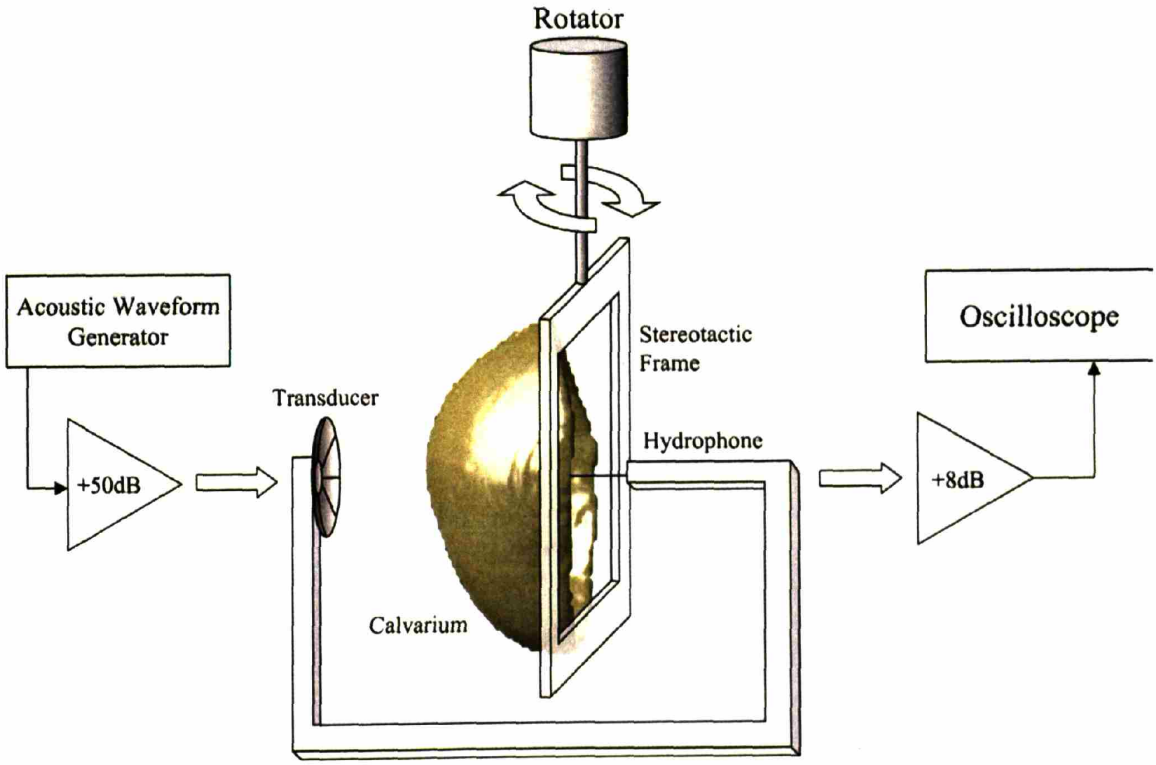

Rotator
Acoustic Waveform Generator
Oscilloscope
Stereotactic Frame
Transducer
Hydrophone
+50dB
+8dB
Calvarium

Figure 5.3

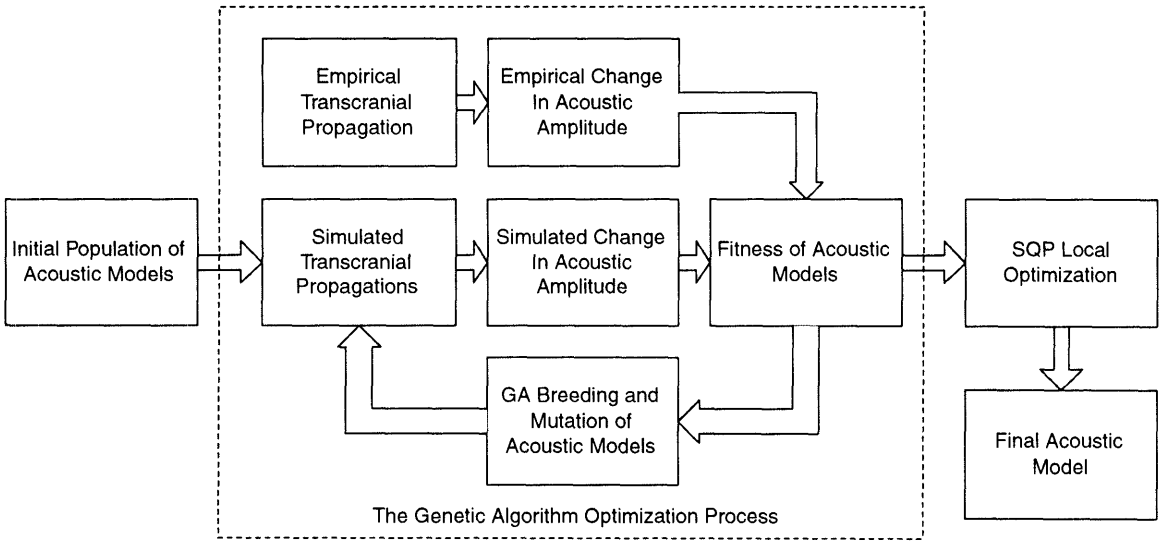

Empirical Transcranial Propagation
Empirical Change In Acoustic Amplitude
Initial Population of Acoustic Models
Simulated Transcranial Propagations
Simulated Change In Acoustic Amplitude
Fitness of Acoustic Models
SQP Local Optimization
GA Breeding and Mutation of Acoustic Models
Final Acoustic Model
The Genetic Algorithm Optimization Process

Figure 5.4

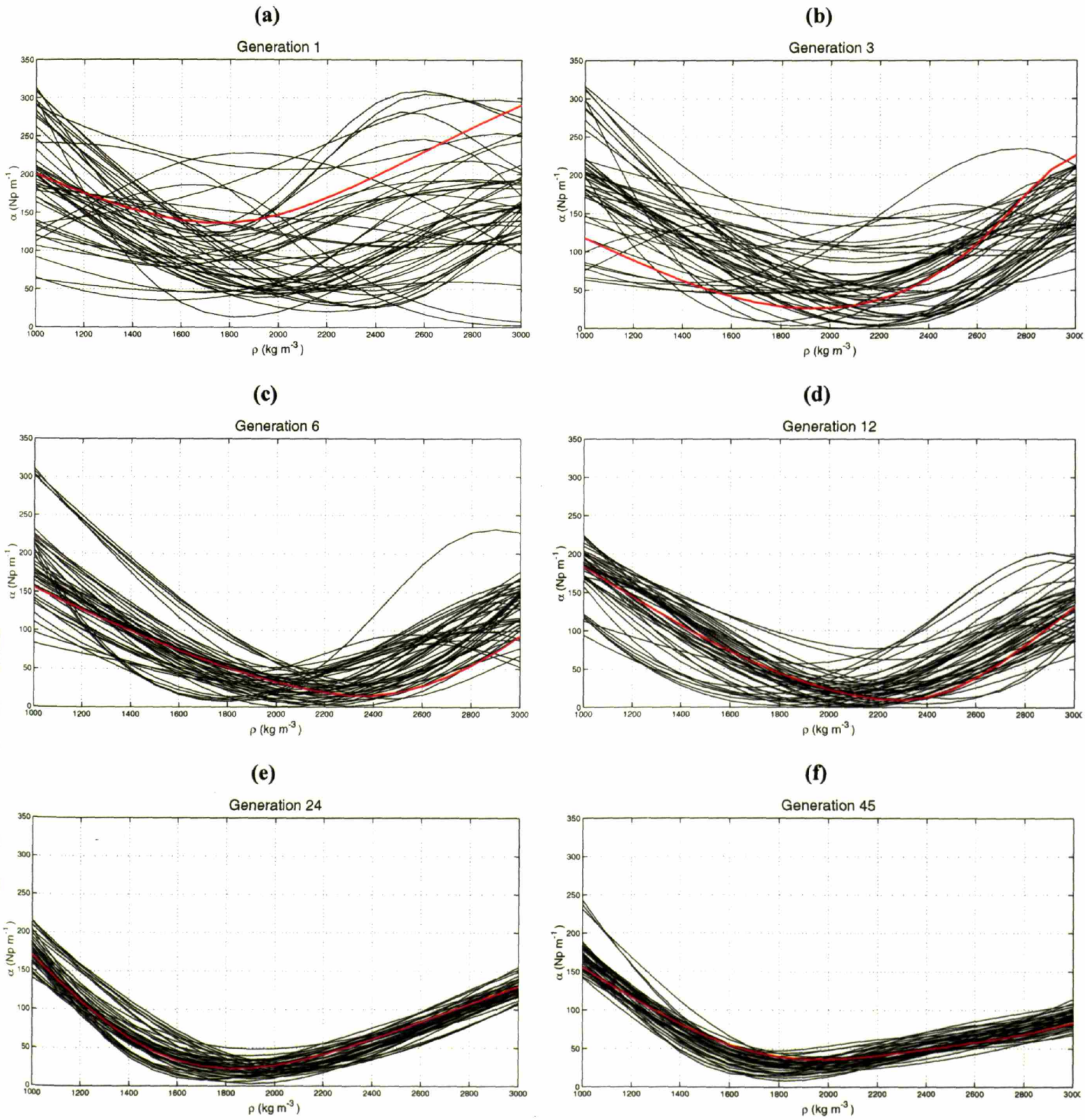

(a)
Generation 1
(b)
Generation 3
(c)
Generation 6
(d)
Generation 12
(e)
Generation 24
(f)
Generation 45
α (Np m-1)
ρ (kg m-3)

Figure 5.5

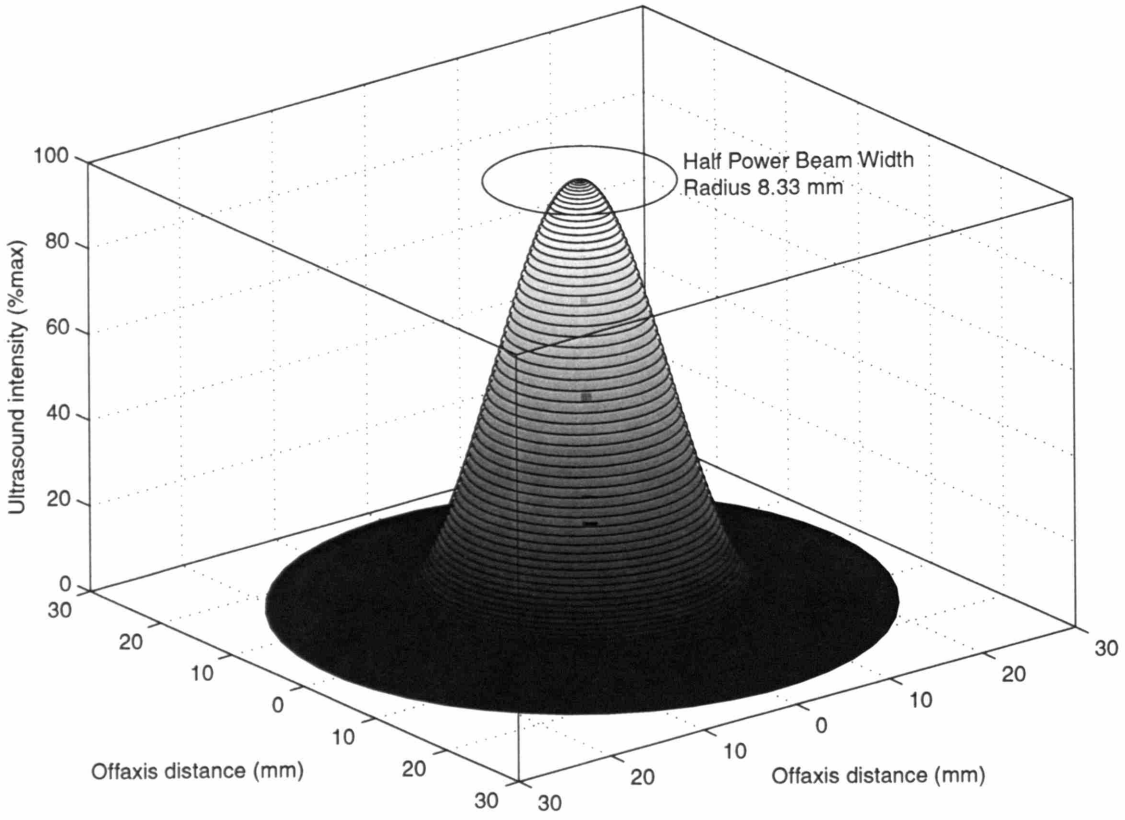

Half Power Beam Width
Radius 8.33 mm
Ultrasound intensity (%max)
100
80
60
40
20
0
30
20
10
0
10
20
30
Offaxis distance (mm)
30
20
10
0
10
20
30
Offaxis distance (mm)

Figure 5.6

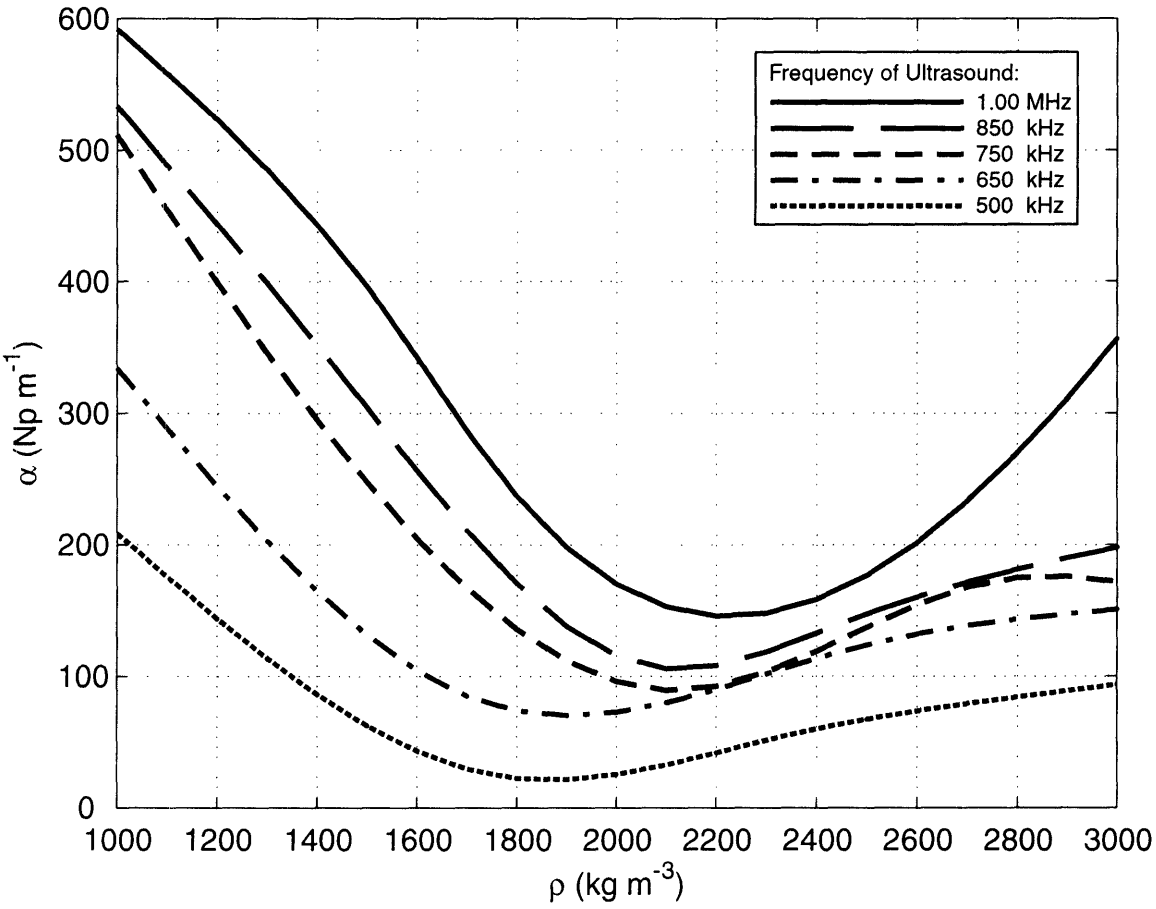

Frequency of Ultrasound:
1.00 MHz
850 kHz
750 kHz
650 kHz
500 kHz
600
500
400
300
200
100
0
α (Np m-1)
1000
1200
1400
1600
1800
2000
2200
2400
2600
2800
3000
ρ (kg m-3)

Figure 5.7

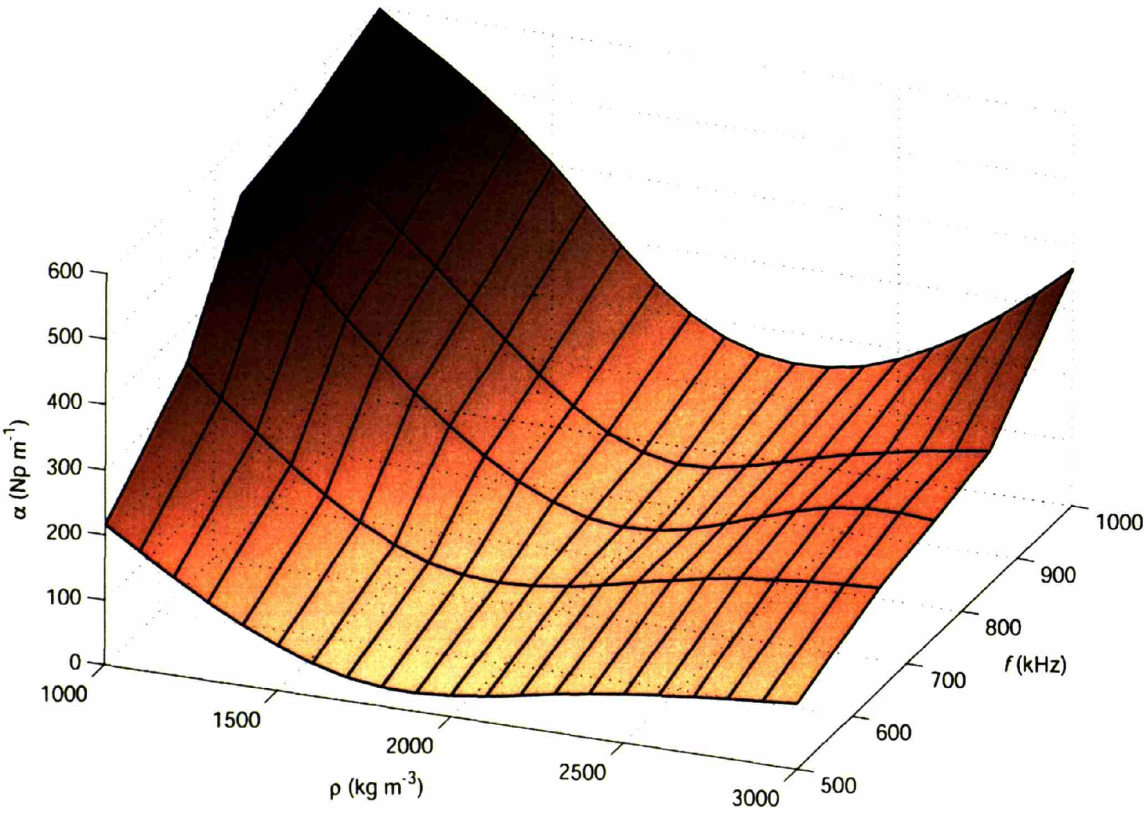

600
500
400
300
200
100
0
α (Np m-1)
1000
1500
2000
2500
3000
ρ (kg m-3)
500
600
700
800
900
1000
f (kHz)

Figure 5.8

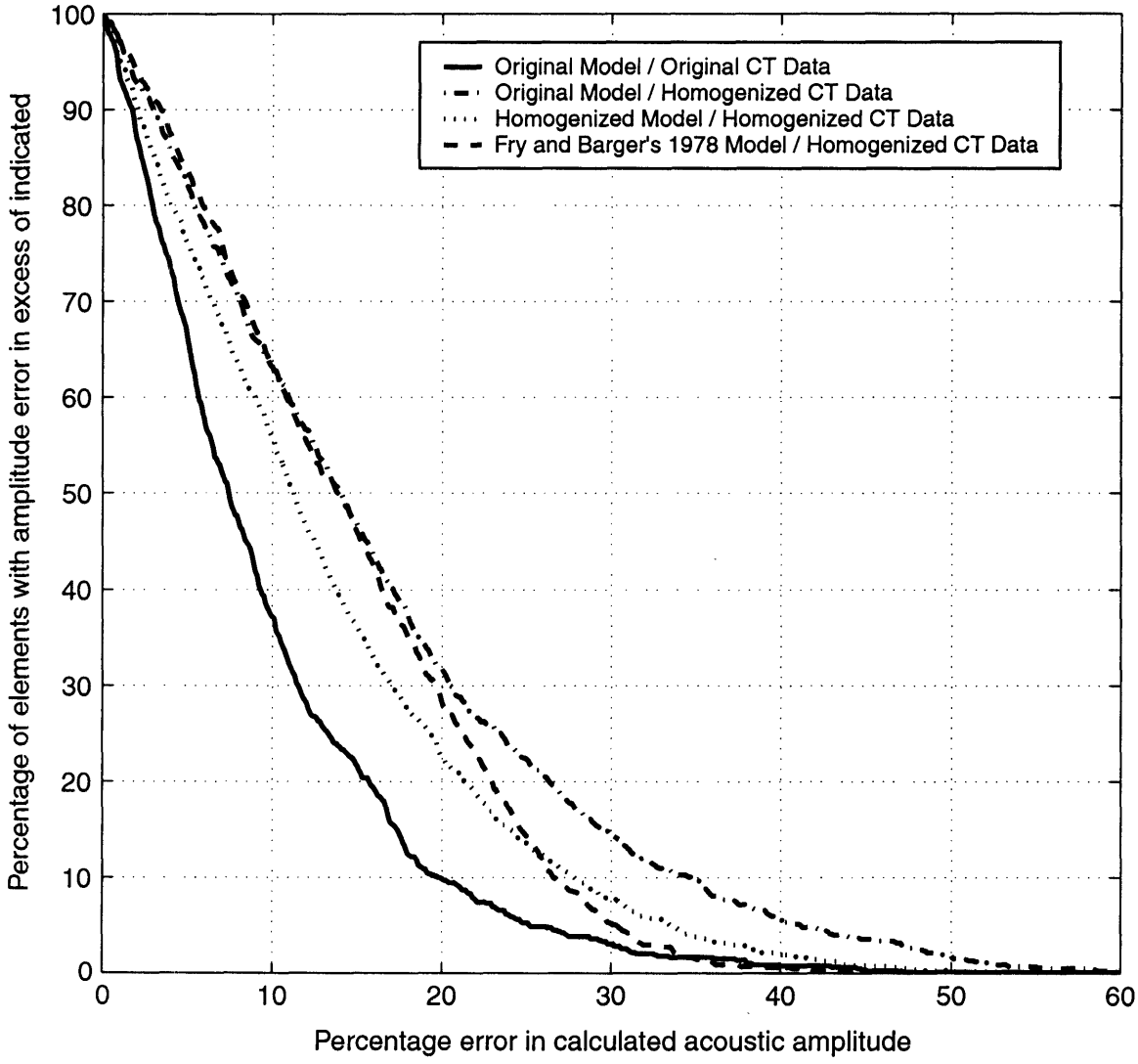

Original Model / Original CT Data
Original Model / Homogenized CT Data
Homogenized Model / Homogenized CT Data
Fry and Barger's 1978 Model / Homogenized CT Data
Percentage of elements with amplitude error in excess of indicated
Percentage error in calculated acoustic amplitude
0
10
20
30
40
50
60
70
80
90
100

Figure 5.9

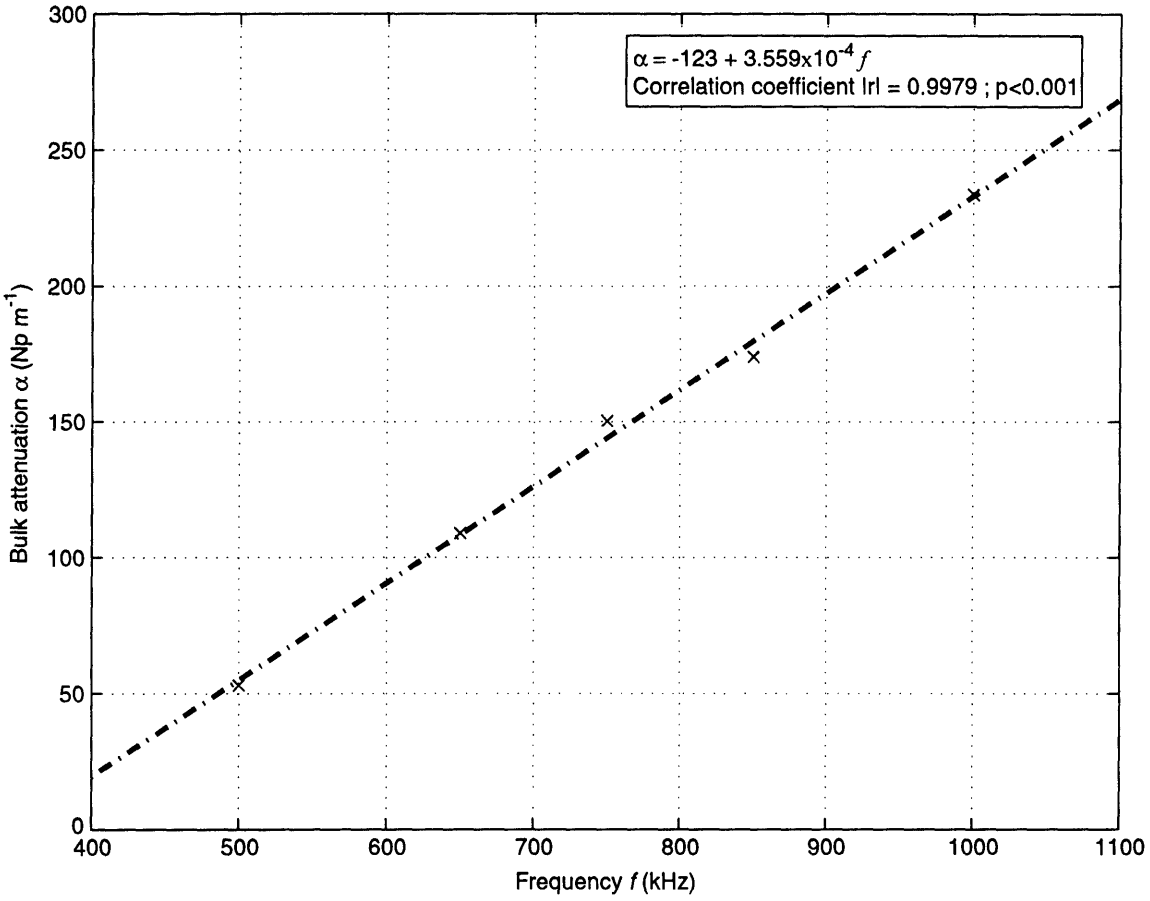
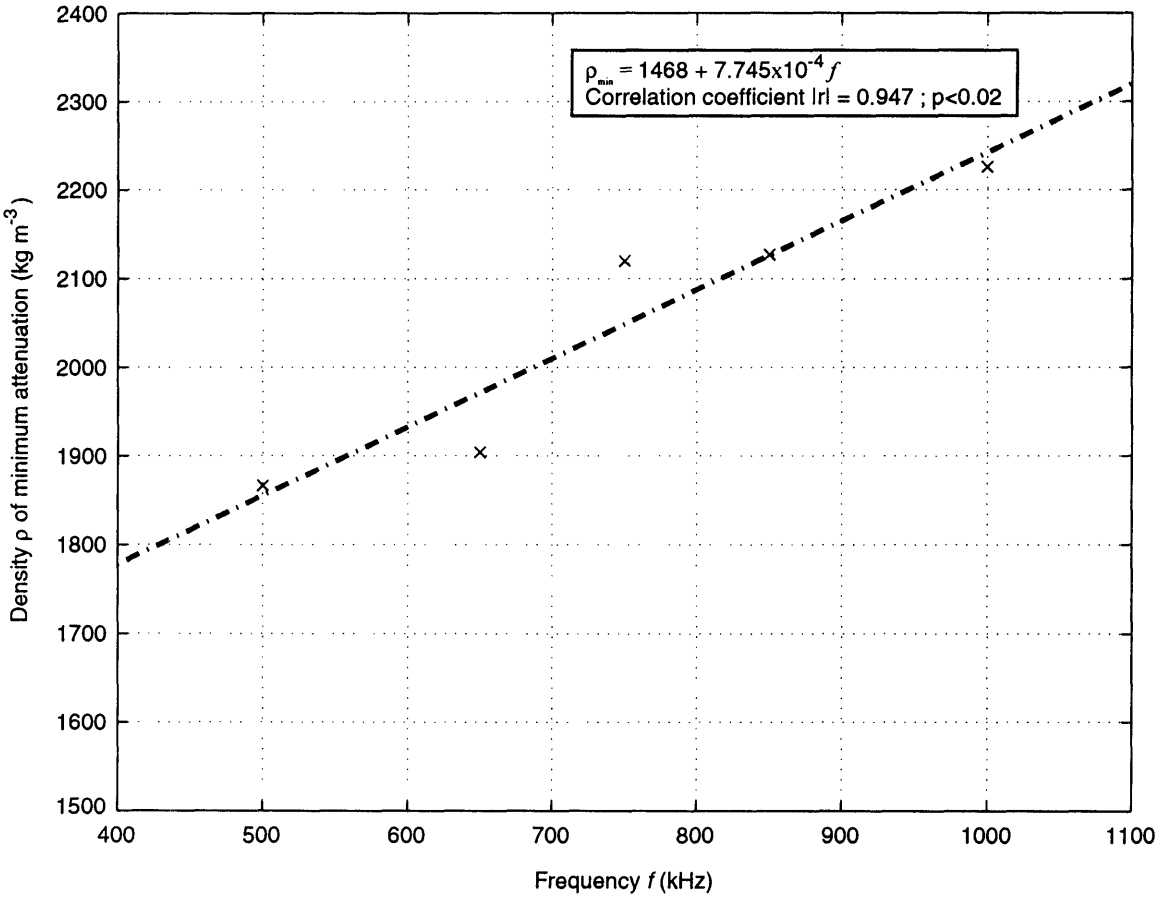

α = -123 + 3.559x10^-4 f
Correlation coefficient |r| = 0.9979 ; p<0.001
Bulk attenuation α (Np m^-1)
Frequency f (kHz)
0
50
100
150
200
250
300
400
500
600
700
800
900
1000
1100

Figure 5.10


$\rho_{min} = 1468 + 7.745 \times 10^{-4} f$
Correlation coefficient |r| = 0.947 ; p<0.02
Density ρ of minimum attenuation (kg m^{-3})
Frequency f (kHz)
2400
2300
2200
2100
2000
1900
1800
1700
1600
1500
400
500
600
700
800
900
1000
1100

Chapter 6

Patterns Of Thermal Deposition In The Skull During Transcranial Focused Ultrasound Surgery.

6.1 ABSTRACT

The induction of temperature elevation by focused ultrasound is a non-invasive surgical technique for destroying tissue. This technique has been used clinically in soft tissues such as liver, prostate and breast. It has long been desired to extend this technique to non-invasive treatment of brain tumors. Although the skull was once thought to be an unsurpassable barrier to focused ultrasound treatment, it has been shown that the distortion caused by the skull can be corrected to produce a useful intracranial focus. However, the attenuation experienced by the ultrasound in passing through cranial bone is large, and consequently the skull is subject to the deposition of acoustic energy as heat. The nature and extent of this heating process has been difficult to characterize empirically. It is practically difficult to implant a sufficient number of thermocouples to obtain detailed temperature data directly, and bone is an unsuitable medium in which to perform non-invasive thermometry using proton chemical shift MRI. Furthermore, skull specimens used experimentally lack active blood perfusion of the skull and the overlying scalp.

This paper describes the use of large-scale acoustic and thermal simulations to calculate the distribution of temperature within the skull and brain that can be expected to occur during therapeutically useful focused ultrasound sonications of the brain. The results demonstrate that standing waves may be formed within the skull during transcranial sonication leading to non-uniform skull heating. However, the results also show that these effects can be sufficiently controlled to allow therapeutic ultrasound to be focused in the cranial base region of the brain without causing thermal damage to the scalp, skull or outer surface of the brain.

6.2 INTRODUCTION

It has long been known that high-powered ultrasound, if focused appropriately, can be used to attack deep-seated tumors non-invasively [1-3]. Many brain tumors, such as those localized in the deep gray matter, brain stem, thalamus or motor areas, are currently surgically inaccessible without causing severe damage to the overlying structures of the brain [4]. Consequently, the use of focused ultrasound to permit minimally invasive treatment of brain tumors is very desirable [5-9]. Ideally, this therapeutic ultrasound sonication should occur through an intact skull. Such a technique would employ an ultrasound transducer placed outside the skull, and would require that the ultrasound field be compensated for the intervening skull bone present in the acoustic path. This paper analyzes the thermal effects that would be produced within an intact skull when highly focused ultrasound emitted from a large multi-element hemispherical array is used to thermally coagulate intracranial lesions within brain tissue.

The simulations performed in this paper are based on a full three-dimensional finite-difference time-domain simulation of acoustic propagation and thermal behavior. The simulations are populated with material properties derived from 3D reconstructions of high-resolution CT scans of human cadaveric skulls. The CT data provides the shape and structure of the skull, and can be used to model the speed of sound within the skull and its attenuation properties. The simulation volume required to model the intracranial ultrasound pressure field is very large; a method is elaborated here for separating and reincorporating the acoustic field to allow parallel computation to be performed. The three-dimensional temperature profile within cranial bone during transcranial focused ultrasound sonication is calculated for a range of human calvaria, for a range of target positions within the brain, and for a range of cooling techniques. Finally, the variation in thermal behavior in the skull is analyzed with regard to the desired temperature at the therapeutic focus, the desired therapeutic treatment volume, and the acoustic pressure of ultrasound used. These simulations also allow for the effects of tissue blood perfusion to be considered; this cannot be tested empirically in *ex vivo* skulls. This theoretical study therefore provides a useful counterpoint to earlier empirical work [10-12].

6.3 METHOD

The first step in simulating the intracranial acoustic field is to capture the structural and material properties of the skull using a high resolution CT scan. In preparation for CT scanning [13], each experimental *ex vivo* calvarium is fixed into a rigid stereotactic frame. This frame has several points that serve as fiducial markers, both to the naked eye and in the CT data. These markers allow the calculation of the position and orientation of the calvarium in the data set acquired from the CT scan [14]. This registration process ensures that the geometry of the computer based acoustic simulations can be matched to the empirical configuration of the transducer and calvarium. CT imaging studies were performed with a Somatom Plus 4 (Siemens) scanner using the AH82 deconvolution kernel with beam properties of 120kV at 200mA and an axial slice separation of 1mm.

The ultrasound acoustic field for transcranial therapy is generated using a large hemispherical array [15]. The sonications presented here are produced using a 500-element array with an internal radius of 0.15m. A 3D reconstruction of a calvarium within this 500-element transducer is shown in Figure 6.1.

The thickness, structure and density of human skull vary between individuals. In order to characterize this variation in this study, three human calvaria were selected for scanning based upon their thickness. These calvaria are representative of the range in variation seen in humans. Following extraction of the geometric structure of each of the calvaria from the CT data, the inside of the modeled calvarium is filled with brain tissue and the outside is covered with scalp tissue of thickness 2.5mm [16].

6.3.1 STRUCTURE OF THE ACOUSTIC SIMULATIONS

A CT scan of a human calvarium will typically produce data covering a volume of 0.2m x 0.2m x 0.2m. The experimental hemispherical transducer used for transcranial focused ultrasound research has an internal radius of 0.15m. Consequently, a full 3D acoustic simulation in Cartesian coordinates would have to encompass a volume of 0.3m x 0.3m x 0.2m at least. Ultrasound at a frequency of 0.75MHz has a wavelength of 2mm in water. A Cartesian mesh with a resolution of 8 points per wavelength would contain a minimum of 1.152 billion points. A simulation of this magnitude is currently impossible with commodity hardware due to the online storage and processing requirements it imposes.

An alternative approach is required. It is reasonably assumed that non-linear effects are negligible in these sonications based on experiments with sharply focused transducers [17], and thus the acoustic fields due to each active transducer element may be linearly superposed to determine the overall acoustic field in a particular region of interest. If the region of interest is sufficiently off-axis as viewed from a particular transducer element, the contribution from that element to the overall field may be neglected as being small; experimentally that element may be left unexcited.

Therefore, instead of determining the acoustic field throughout the calvarium in one simulation, this problem may be decomposed into determining the field in the calvarium due to each active transducer element. These elemental simulations can be constrained spatially to the regions of the acoustic field of that element that are of interest. Most importantly, these elemental simulations are readily parallelizable – each elemental simulation may be performed independently with the final results recombined, allowing distributed computing to be performed. This provides an enormous computational advantage.

A simpler example of this process is shown in Figure 6.2, in which the arrangement of simulations for 5 active elements in a 64-element hemispherical array is shown. While

the studies performed in this paper use a 500-element hemispherical array, the manner in which the 5 simulations interact in Figure 6.2 is exactly the same and is easier to understand. The orientations of these simulations are maintained relative to the original frame of reference of the CT scan. Each simulation has a solid body transformation **R** associated with it that collectively allow the final acoustic field from each to simulation to be aligned and superposed. With a distributed processing network of 10 unexceptional contemporary desktop PCs, the acoustic field in the calvarium due to all 500 active elements of a 500-element hemispherical array can be calculated in approximately 2 days. The simulation techniques used to achieve this are outlined below.

The required size of the simulation mesh for each element can be determined by considering the spatial distribution of energy radiated by that element in its focal plane, since this energy is spatially constrained to a sharp peak of intensity with little activity outside that peak. The elements of the 500-element hemispherical array have a typical Bessel intensity distribution in the focal plane with a half-power radius of 8.33mm. In fact, if the simulation is configured such that it circumscribes the intensity peak at the 10% level in the focal plane, it is straightforward to show by numerical integration that the simulation accounts for over 97% of the energy radiated to the focal plane by that transducer element.

6.3.2 FINITE DIFFERENCE-TIME DOMAIN (FDTD) SIMULATION OF TRANSCRANIAL PROPAGATION

The propagation of ultrasound through cranial bone is modeled using the Westervelt equation [18].

$$\nabla^2 p - \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3} + \frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2} + \left(\frac{\partial p}{\partial t}\right)^2 \right] - \nabla p \cdot \nabla(\ln \rho) = 0 \qquad (6.1)$$

where p is pressure, c is the speed of sound, δ is acoustic diffusivity, β is the coefficient of non-linearity and ρ is the density. The simulation is performed on a cylindrical mesh. This geometry provides a greater density of data points around the axis of the mesh,

which is typically the region of greatest interest; this provides an advantage over a uniform Cartesian mesh structure [19].

Recognizing that $d(\ln \rho)/dz \equiv \rho^{-1}.d\rho/dz$, in cylindrical coordinates, the Westervelt equation becomes:

$$\begin{aligned} &\frac{\partial^2 p}{\partial r^2}+\frac{1}{r}\frac{\partial p}{\partial r}+\frac{1}{r^2}\frac{\partial^2 p}{\partial \theta^2}+\frac{\partial^2 p}{\partial z^2}-\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}+\frac{\delta}{c^4}\frac{\partial^3 p}{\partial t^3}+\ldots \\ &\frac{\beta}{\rho c^4}\left[p\frac{\partial^2 p}{\partial t^2}+\left(\frac{\partial p}{\partial t}\right)^2\right]-\frac{1}{\rho}\left[\frac{d\rho}{dr}\frac{dp}{dr}+\frac{1}{r^2}\frac{d\rho}{d\theta}\frac{dp}{d\theta}+\frac{d\rho}{dz}\frac{dp}{dz}\right]=0 \end{aligned} \tag{6.2}$$

Since $\delta = 2\alpha c^3/\omega^2$, for harmonic excitation $\frac{2\alpha}{c\omega^2}\frac{\partial^3 p}{\partial t^3} \equiv -\frac{2\alpha}{c}\frac{\partial p}{\partial t}$, where α is the acoustic amplitude absorption coefficient and ω is the angular frequency.

While the cylindrical mesh has its geometrical benefits, as stated previously, it brings with it some special problems regarding discretization of the partial differential equation above and the stability of the FDTD method. The numerical stability becomes very restrictive close to the cylindrical axis of the mesh due to the spatial density of the mesh points. A very small finite-difference time step Δt must be used if divergent oscillation of the solution is to be avoided; however, a small time-step leads to a computational model that is grossly inefficient and slow. This problem can be solved by running different parts of the FDTD mesh at time steps that are integer multiples of the basic time step Δt; this method is termed *gearing* and is treated in substantial detail in Connor *et al.* [20].

6.3.3 THE ACOUSTIC PROPERTIES OF CRANIAL BONE

FDTD simulation is based on the local properties of the material. As cranial bone is a heterogeneous material, simulation of acoustic propagation through it should have values of α and c that vary within the tissue. These values can be assigned to each mesh point in the tissue based on the local apparent CT density. The variation of material properties in heterogeneous structures can be determined using genetic algorithm optimization, and has already been performed for speed of sound in cranial bone [21] and for attenuation

coefficient in cranial bone [22]. The speed of sound and attenuation coefficient models used in these simulations can be found in Table 6.1. The value of the non-linear parameter β is set to zero, effectively simulating linear acoustic propagation. The value of β has not yet been satisfactorily determined in skull as the Goldberg number Γ is expected to be small due to the high attenuation α [23]. The pressure away from the geometric center is sufficiently low that non-linear effects are negligible in this region, and consequently pre-focal non-linear effects do not accumulate appreciably. The acoustic and thermal properties of other materials used in this study are given in Table 6.2.

6.3.4 Completing and Initiating Simulations with the Huygens-Fresnel integral

The FDTD process is an appropriate choice for determining the propagation of the ultrasound field through cranial bone, since this is a medium in which the material density, speed of sound and absorption coefficient each have substantial spatial variance. However, the FDTD process is a less appropriate choice for propagating through materials that are acoustically homogeneous or may be considered to be nearly so, such as the degassed water surrounding the skull or the brain tissue encased by it. Under these circumstances, a more analytical approach can give greater efficiency and we have combined the FDTD process with the Huygens-Fresnel integral for our analyses.

The basis for the following analytical methods is the evaluation of the Huygens-Fresnel integral. In its most general form, we have:

$$U(P_1) = \iint_{\Sigma} h(P_0, P_1).U(P_0)\, ds \tag{6.3}$$

where U is particle velocity, h is an acoustic transfer function between spatial points P_0 and P_1, and Σ is some two-dimensional surface which may or may not be tangible. Now, given that $U = p/\rho c$ where p is acoustic pressure amplitude, and ρc is acoustic impedance, we have:

$$p(P_1) = \iint_{\Sigma} h(P_0, P_1).p(P_0)\, ds \tag{6.4}$$

which is more useful here, and where the transfer function h is:

$$h(P_0, P_1) = \frac{1}{j\lambda}.\frac{\exp(jk|\underline{r}_{01}|)}{|\underline{r}_{01}|}.\cos(\underline{n}, \underline{r}_{01}) \tag{6.5}$$

where λ is the acoustic wavelength, k is the acoustic wave number, $\underline{r}_{01}$ is the vector from P_0 to P_1, $\underline{n}$ is the unit vector normal to the surface Σ, and $\cos(\underline{n}, \underline{r}_{01})$ is the obliquity function between the two vectors $\underline{n}$ and $\underline{r}_{01}$. The obliquity factor represents the limited contribution of some surface element ds to the acoustic field at a point that is substantially off the normal axis for that surface element. Often in analytical approximations, it is assumed that $\cos(\underline{n}, \underline{r}_{01}) \approx 1$. However, we can completely analyze the two cases for the obliquity function that are relevant to these simulations. These are the obliquity factor from a point in one plane of the cylindrical mesh to a target point and the obliquity factor from a point on a hemispherical surface to a target point, as follows.

6.3.4.1 Plane-to-plane obliquity factor

The FDTD process is used to establish the ultrasound field inside the skull, and then the Huygens-Fresnel integral is used to complete the acoustic propagation through the brain tissue. Since the simulation is performed on cylindrical mesh, we require the obliquity factor between points on two parallel circular planes. Hence, for two points P_0 and P_1 at cylindrical polar coordinates $[r_0,\theta_0,z_0]$ and $[r_1,\theta_1,z_1]$, we have:

$$|\underline{r}_{01}|.\cos(\underline{r}_{01}, \underline{n}) = \left(\begin{bmatrix} r_1\cos(\theta_1) \\ r_1\sin(\theta_1) \\ z_1 \end{bmatrix} - \begin{bmatrix} r_0\cos(\theta_0) \\ r_0\sin(\theta_0) \\ z_0 \end{bmatrix} \right) \bullet \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{6.6}$$

since in this case the unit normal vector $\underline{n}$ is clearly parallel to the z-axis. The remaining task is to determine an expression for $|\underline{r}_{01}|$, the scalar distance between P_0 and P_1. This distance can be considered to have two components: the distance between $[r_0,\theta_0]$ and $[r_1,\theta_1]$in a circular plane and the distance between z_0 and z_1. From the cosine law, the first part is:

$$\underline{r}_{01,plane}^2 = r_0^2 + r_1^2 - 2r_0 r_1 \cos(\theta_1 - \theta_0) \tag{6.7}$$

And consequently:

$$\cos(\underline{r}_{01}, \underline{n}) = \frac{z_1 - z_0}{\sqrt{r_0^2 + r_1^2 - 2r_0 r_1 \cos(\theta_1 - \theta_0) + (z_1 - z_0)^2}} \tag{6.8}$$

6.3.4.2 Hemispherical transducer obliquity function

The Huygens-Fresnel integral is also used to determine the ultrasound field in the degassed water between the transducer and skull before the FDTD process is used to calculate the transcranial portion of the simulation. This step requires the obliquity function from the surface of the hemispherical transducer to some other point. However, this analysis is more complicated than the foregoing plane-to-plane case as the orientation of the unit surface normal vector $\underline{n}$ is no longer constant. Consider a point on the surface of the transducer with position vector $\underline{r}_t$, some arbitrary other point $\underline{r}_1$, and the position of the geometric focus of the hemispherical array $\underline{z}_f$ where, as in Figure 6.2, the orientation of the simulation is configured so that the geometric center of the array is on the cylindrical axis of the simulation mesh:

$$\underline{r}_t = \begin{bmatrix} r_t \cos(\theta_t) \\ r_t \sin(\theta_t) \\ z_t \end{bmatrix} ; \underline{r}_1 = \begin{bmatrix} r_1 \cos(\theta_1) \\ r_1 \sin(\theta_1) \\ z_1 \end{bmatrix} ; \underline{r}_f = \begin{bmatrix} 0 \\ 0 \\ z_f \end{bmatrix} \tag{6.9}$$

And therefore:

$$\left|\underline{r}_1 - \underline{r}_t\right| . \left|\underline{r}_f - \underline{r}_t\right| . \cos(\underline{r}_1 - \underline{r}_t, \underline{r}_f - \underline{r}_t) = \begin{bmatrix} r_1 \cos(\theta_1) - r_t \cos(\theta_t) \\ r_1 \sin(\theta_1) - r_t \sin(\theta_t) \\ z_1 - z_t \end{bmatrix} \bullet \begin{bmatrix} -r_t \cos(\theta_t) \\ -r_t \sin(\theta_t) \\ z_f - z_t \end{bmatrix} \tag{6.10}$$

And by similar arguments to those used in the plane-to-plane case, the obliquity factor can be determined to be:

$$\cos(\underline{r}_1 - \underline{r}_t, \underline{r}_f - \underline{r}_t) = \frac{r_t^2 - r_t r_1 \cos(\theta_1 - \theta_t) + (z_1 - z_t)(z_f - z_t)}{\left(\sqrt{r_t^2 + (z_f - z_t)^2}\right)\left(\sqrt{r_t^2 + r_1^2 - 2r_t r_1 \cos(\theta_1 - \theta_t) + (z_1 - z_t)^2}\right)} \tag{6.11}$$

6.3.5 HIERARCHICAL MULTIRESOLUTION MODELING OF SOURCES FOR ACOUSTIC SIMULATION

Evaluating obliquity functions and their associated transfer functions can become sufficiently numerically intensive that the gain over efficient FDTD analysis can be small. However, we can obtain a great increase in computational speed by using a hierarchical multiresolution representation of the transducer surface. The technique of multiresolution modeling was first applied to the rendering of 3D graphics [24-26]; instead of using the maximum resolution of the 3D models for all the objects in a 3D scene, the models for objects far from the camera are coarsened. Because these objects are distant in the scene, the coarsening of the 3D model is imperceptible and rendering time is improved with no apparent loss of image quality. Originally, a number of 3D models of varying resolution would be created for each object. A hierarchical multiresolution model improves on this by containing both the highest 3D resolution model and a description of how the parts of the model relate – hierarchically – to the overall structure. This description allows a model of appropriate coarseness to be produced dynamically by retaining necessary detail and replacing superfluous detail with a simpler form.

This process has a counterpart in analytical acoustics. While the Huygens-Fresnel integral can be used to analyze the entire acoustic field of a transducer of arbitrary shape, in practice it is typically seen in the formulation of far-field approximations of simple transducer geometries. The Fresnel and Fraunhoffer approximations are based, respectively, on the assumption that the obliquity factor is unity and that the distance $\underline{r}_{01}$ can be reasonably approximated using the second order binomial expansion of the square root. Indeed, the Fraunhoffer distance is defined as the distance at which this approximation becomes reasonable. In essence this is a transition to a more tractable, coarser analytical model from a more detailed, perhaps analytically intractable one at the

point where the geometry of the problem makes this admissible; a distinct conceptual parallel to the method outlined above for 3D modeling.

By adding hierarchical multiresolutioning to the definition of the transducer surface, an order of magnitude increase in the speed of evaluating the FDTD-initiating Huygens-Fresnel integral is readily achievable. Furthermore, this technique is not dependent on the transducer having any special geometric form, nor does it require that further analytical approximations for the acoustic field be provided [27,28]. With appropriate definition of the hierarchical form of the transducer surface, multiresolutioning can be used efficiently for arbitrary transducer geometries.

6.3.6 FOURIER CAPTURE IN FDTD ANALYSIS

Once the ultrasound wave has been propagated through the skull, it is necessary to convert the ultrasound field back into phasor form so that the full intracranial field can be propagated through the brain tissue by the Huygens-Fresnel integral. This requires the temporal Fourier transformation of the FDTD data. Ordinarily, this process would impose impossibly large storage and processing requirements. However, let us stipulate (without loss of generality in the linear harmonic case) that:

$$\tau = n_{cycle}.\Delta t, \qquad n_{cycle} \in \mathbb{Z}^{+} \tag{6.12}$$

i.e. the temporal period τ of the ultrasound wave is an integer multiple n_{cycle} of the FDTD step size Δt. Under this condition the analysis is substantially more tractable, since the Fourier transform can be calculated in one FDTD cycle. First we calculate $\mathfrak{F}_P$ and $\mathfrak{F}_Q$, the phase and quadrature amplitudes of the time signal, for each point in the FDTD mesh.

$$\begin{aligned} \mathfrak{F}_P &= \sum_{n=N-n_{cycle}}^{N} \left(p(n).\hat{p}_t(n) \right) \\ \mathfrak{F}_Q &= \sum_{n=N-n_{cycle}}^{N} \left(p(n).\hat{p}_t(n+0.25n_{cycle}) \right) \end{aligned} \tag{6.13}$$

where the datum phase is established by using $\hat{p}_t$, the unit amplitude time signal of the ultrasound wave at the transducer. The important point here is that the final values of $\mathfrak{F}_P$

and $\mathfrak{F}_Q$ at time N can be accumulated over the preceding n_{cycle} time steps, so very little extra storage is required. The amplitude and phase of the ultrasound field can be calculated directly from $\mathfrak{F}_P$ and $\mathfrak{F}_Q$ at every point in the mesh:

$$(p_0, \phi) = \left(\frac{2}{n_{cycle}} \sqrt{\mathfrak{F}_P^2 + \mathfrak{F}_Q^2}, \tan^{-1}\left(\mathfrak{F}_Q / \mathfrak{F}_P\right) \right) \tag{6.14}$$

where p_0 and ϕ are the amplitude and phase of the ultrasound field at a particular point in the mesh, and hence consequently $\tilde{p} = p_0 e^{i\phi}$ where $\tilde{p}$ is the pressure phasor.

6.3.7 FROM ACOUSTIC FIELDS TO THERMAL DEPOSITION

Temperature distribution throughout the tissue under sonication is modeled using the Bioheat Equation [29]:

$$\rho_t C_t \frac{\partial T}{\partial t} = k_t \nabla^2 T - \rho_b C_b W (T - T_a) + Q \tag{6.15}$$

where C is specific heat capacity, k is thermal conductivity and W is the perfusion rate. The variable Q varies spatially and represents the acoustic energy absorbed by the tissue at a given point, and is given by [30]:

$$Q = \frac{1}{n\tau} \int_{t'}^{t'+n\tau} \frac{\delta}{\rho c^4} \cdot \left(\frac{\partial p}{\partial t} \right)^2 dt \tag{6.16}$$

However, given that $\delta = 2\alpha c^3 / \omega^2$, the expression for Q can be simplified as follows:

$$Q = \overline{\frac{2\alpha}{\rho c \omega^2} \left(\frac{\partial p}{\partial t} \right)^2} = \frac{2\alpha}{\rho c \omega^2} \overline{\left(\frac{\partial p}{\partial t} \right)^2} = \frac{2\alpha}{\rho c \omega^2} \cdot \omega^2 \cdot \overline{p^2} = \frac{\alpha}{\rho c} \cdot p_{\max}^2 = \frac{\alpha}{\rho c} \cdot \left(\sum_n \tilde{p}_n \right)^2 \tag{6.17}$$

where $\tilde{p}_n$ is the complex pressure amplitude phasor due to element n.

Thus, to produce the complete spatial variability of Q in the co-ordinate system of the thermal simulation, we have:

$$Q_{thermal}(x,y,z) = \frac{\alpha}{\rho c} \cdot \left(\sum_{\substack{\text{n active} \\ \text{elements}}} \tilde{p}\left(\mathbf{R}_n \mathbf{R}_{thermal}^{-1} \begin{bmatrix} x & y & z & 1 \end{bmatrix}' \right) \right)^2 \tag{6.18}$$

where α is the local acoustic amplitude absorption coefficient, c is the local speed of sound, and ρ is the local density. $\mathbf{R}_{thermal}$ and $\mathbf{R}_n$ are the solid body transformation as described in section 6.3.1 in classical augmented matrix form for the thermal simulation co-ordinate system and for the simulation of a particular active transducer element respectively.

The Bioheat Equation is discretized and solved numerically on a cylindrical mesh with equivalent node spacing to that used in the acoustic simulations. As with the acoustic case, this achieves a high density of mesh points around the focus. The small spatial separation of points in the vicinity of the focus can cause difficulties with ensuring the numerical stability of the algorithm. However, this problem can be overcome by allowing variable time-steps to exist in the numerical discretization [20]. The boundary conditions of the simulation are constructed by requiring no energy flow across a boundary. This constitutes a classical Dirichlet boundary condition for temperature, and is reasonable for large simulation volumes where the boundaries are distant from strong thermal sources and sinks. The axial and radial resolutions are both 2.53×10^{-4} m, and the angular resolution is 7.5°. The shortest distance between points is 16.5μm, in the axial region of the cylindrical mesh.

6.3.8 THERMAL CONDUCTIVITY IN CRANIAL BONE

The thermal conductivity parameter k in cranial bone is significant because it is a measure of the ability of cranial bone to dissipate absorbed acoustic energy to thermal sinks, and because of the relative importance of the potential appearance of hot-spots in cranial bone under sonication. Hot-spots might potentially be produced by the formation of standing waves between the inner and outer surface of the skull leading to areas of

locally greater acoustic heating. A high value of k reduces the importance of standing waves by allowing the rapid dissipation of local hot-spots into cooler surrounding cranial bone.

Clattenburg *et al.* [31] noted that the value of k in bone appeared to be isotropic and not dependent on trabecular orientation in trabecular bone. Directional differences in k were observed in cortical bone [32], but these were considered to be sufficiently small that isotropic behavior could be considered a reasonable approximation. Consequently, it appears that a single value of k is sufficient for the purposes of thermal simulation in cranial bone, provided that a single representative value of k can be determined.

A wide variation exists in the literature for the value of k in bone [31-35], and much of this work has been accomplished in femoral bone rather than cranial bone. The bones of the cranial skull are somewhat different to femoral bone in structure, however. The trabecular bone in skull tends to be of higher apparent density, and the transition from cortical to trabecular bone is less distinct. Consequently, an investigation into the thermal conductivity of cranial bone was performed using a fresh sample of bovine calvarium using implanted heating elements and thermistors [36]. Measurements of the thermal conductivity of bovine cranial bone produced a value of $k = 0.43$ W m^{-1} K^{-1}. This value is in accordance with those described previously [31,32] for bovine samples, and also with theoretical expectations.

6.3.9 MODELING THERMAL DAMAGE AND THERAPEUTIC VOLUME IN TISSUE

The therapeutic effect of a sonication can be determined once the temperature profile throughout the sonication has been calculated as shown in Section 6.3.7. The volume of tissue exposed to a therapeutic level of focused ultrasound is calculated using the Sapareto-Dewey [37] integral, which converts a given temperature profile at a particular location to an equivalent period of constant thermal exposure at 43°C, as follows:

$$T_{43} = \int_0^t R^{T_{ref}-T(t)} dt \qquad R = \left\{ \begin{matrix} 0.5, T \geq T_{ref} \\ 0.25, T \leq T_{ref} \end{matrix} ; T_{ref} = 43^{\circ}C \right\} \tag{6.19}$$

Tissue exposed to an equivalent exposure of 240 minutes at 43°C undergoes coagulation; no tissue exposed to less than an equivalent exposure of 30 minutes at 43°C is damaged [37]. These two limits provide a means for estimating the margin of the therapy.

6.3.10 CONFIGURATION OF SIMULATIONS AS A PARAMETRIC STUDY

At the most abstract level, a simulation of transcranial focused ultrasound surgery is dependent on only three factors:

(1) the skull through which ultrasound is being propagated,

(2) the targeted position of the ultrasound focus in the brain, which determines the position of that skull relative to the ultrasound transducer,

(3) the coupling medium between the skull and the transducer.

A complete simulation, using the methods described previously, requires firstly the simulation of acoustic propagation from each transducer element, secondly the reconstruction of these elemental propagated fields into a single total acoustic field from which the thermal deposition of acoustic energy may be determined, and thirdly a simulation of the resulting thermal behavior.

As described earlier, three calvaria were CT scanned to span the range of skull thicknesses seen in humans. Of these calvaria, one is thin, one is of intermediate thickness, and one is thick. Simulations of transcranial focused ultrasound surgery were performed for each of these calvaria in turn, with the calvaria positioned relative to the hemispherical transducer such that the natural geometric focus of the transducer would fall upon the central (or petroclival) region of the cranial base.

The intermediate thickness calvarium was then simulated in three further different positions relative to the transducer such that the geometric focus of the transducer would fall, respectively, 2.5cm laterally, 5cm laterally, and 5cm rostrally of the position

described above. With respect to equivalent anatomical landmarks, the lateral shifts encompass the cerebellopontine angles and the jugular foramina, and the rostral shift is representative of the position of the corpus callosum [38].

The coupling medium between the transducer and the skull is degassed water at 15°C; this acts as both an acoustic coupling medium and also as a thermal sink to palliate any increase in intracranial temperature caused by the transcranial propagation of ultrasound. However, this water may either be actively exchanged to maintain its temperature at 15°C, or it may be used to cool the skull passively. From a historical perspective, empirical studies on transcranial focused ultrasound have also tended to make use of skull specimens without a layer of scalp due to the difficulty in maintaining fleshy experimental cadaveric specimens. Therefore, the effect of the presence or absence of scalp on the thermal behavior of calvaria also warrants examination.

Taken as a whole, these simulations form a parametric variation about the main case of a calvarium of intermediate thickness with scalp through which ultrasound is focused onto the central region of the cranial base, using active exchange of the coupling medium to assist cooling. The parameters are the thickness of the calvarium, its position, the presence or absence of scalp, and the use of active cooling.

6.4 RESULTS

Figure 6.3a and Figure 6.3b show the distribution of the acoustic pressure field for the main parametric case of the calvarium of intermediate thickness with a central acoustic focus. Figure 6.3a is windowed in pressure to ±1MPa to reveal the overall distribution of the field throughout the simulation volume. Since this windowing creates saturation in the pressure scale, Figure 6.3b is windowed to ±8.5MPa to demonstrate the actual sharpness of the acoustic focus. The focus produced by this hemispherical array is ellipsoidal with a major diameter of 3mm and a minor diameter of 2mm.

In each of the parametric cases, the acoustic pressure at the surface of the transducer was adjusted to attempt to produce a peak tissue temperature of 70°C in brain at the ultrasound focus after 20 seconds of sonication. The acoustic pressures used for each calvarium and transducer configuration are given in Table 6.3.

The temperature distributions at the end of sonication are shown in Figure 6.4a-f for the whole simulation volume. Figure 6.5a-f shows the same results magnified to display the thermal distribution with the skull more clearly. The temperature changes within these calvaria are non-uniform, indicative of non-uniform absorption of acoustic energy. The first five cases as shown in Figure 6.4a-e and Figure 6.5a-e, and as described in Table 6.3, show that it is possible to achieve a therapeutic peak focal temperature within brain tissue using a transcranial ultrasound field without causing thermal damage to cranial bone or to the scalp. The ultrasound focus in these cases is positioned in a manner that would be appropriate for treating brain tumors near the cranial base.

The parametric case shown in Figure 6.4f and Figure 6.5f places the focus in a much more rostral location as would be required for a tumor higher in the cerebrum, and here it was not possible to create a suitable therapeutic focus at the targeted point as the heating in the skull was too great.

If the scalp tissue is removed to leave denuded calvaria, the thermal distributions shown in Figure 6.6a-f are produced. The calvaria are actively cooled here, and it is apparent upon examination that the peak temperatures within the denuded calvaria are lower than in the comparable previous case.

Figure 6.7a-f shows the thermal distribution produced in these same denuded calvaria, but under the assumption that no active cooling period was allowed and that sonication began immediately. The peak temperatures within these calvaria are noticeably higher than those in Figure 6.5a-f and Figure 6.6a-f.

The thermal effect of the absence or presence of the scalp is shown in Figure 6.8a and Figure 6.8b respectively. The scalp is not greatly heated during sonication since the attenuation in the scalp is small in comparison to the attenuation in skull. While the skull can still be cooled effectively through the scalp [16], there is a difference between the peak temperature within the calvaria in these cases of 3.6°C.

6.5 Discussion

It is clear that some intracranial heating must occur due to the transcranial ultrasound sonication. Ideally this heating should be minimized to avoid causing intracranial tissue damage. However, the pattern of heating within the skull appears not to be uniform. Figure 6.5f in particular indicates that heating occurs in several distinct locations. These spots appear to be periodically spaced, a pattern of heating that implies the existence of standing acoustic waves or constructive interference within the skull. A similar pattern of heating can also be seen more faintly in the main parametric case of Figure 6.5b. The distribution of acoustic wavefronts within the skull can be seen in Figure 6.3a. This distribution would appear to allow for the formation of standing waves where an acoustic path through the skull could be found that comprises an integer number of half-wavelengths. Such paths would tend to produce standing waves, although identifying these paths *a priori* is complicated by the varying speed of sound within the bone. The high acoustic attenuation within the skull means that the formation of standing waves is somewhat suppressed since a single wavefront cannot undergo multiple internal reflections without being dissipated. However, since the local heating effect of the ultrasound is proportional to the square of the acoustic pressure amplitude, a small degree of constructive interference may be responsible for the larger perturbations in heating pattern seen here. Figure 6.9a shows the distribution of the absorbed acoustic power Q, as defined in Section 6.3.7, for the main parametric case. The presence of parallel stripes of power deposition in this figure are indicative of standing waves, giving rise to the thermal distribution shown in Figure 6.9b after 20 seconds of sonication.

A homogenized representation of brain has been used throughout the simulations presented here. This approach would not be acceptable for the purposes of simulating ultrasound imaging of the brain, since then the fine structure of the brain under sonication would be of great interest. However, for the purposes of modeling therapeutic focused ultrasound, the brain can be considered to be approximately homogeneous. The variations in acoustic impedance within the brain and associated structures are small in comparison to the variation of acoustic impedance within the skull. Consequently, in comparison to

the distortions in the acoustic field induced by the skull, the effects of inhomogeneity within the brain are likewise small. Once the ensonicating ultrasound field has passed through the skull, it can be considered to be within an almost homogeneous medium for the purposes of a therapeutic acoustic field. This simplifying assumption is also necessary to perform a general analysis of thermal behavior in transcranial ultrasound. The thermal simulations are also performed with this homogenized representation of brain, and the assumption of homogeneity remains valid in this case provided that the tissue surrounding the therapeutic ultrasound focus is locally homogeneous. Given the size of the therapeutic focus, this assumption will be upheld except when the tissue being targeted is directly adjacent to a ventricle, venous sinus or major cerebral artery. In such cases, the thermal properties of the appropriate fluid should also be modeled [39].

While all the sonications presented here have been configured to produce a peak temperature within brain of 70°C after 20 seconds, the desired therapeutic outcome can be varied easily for a particular combination of skull and transducer orientation since the initial acoustic analysis is unchanged and only the thermal analysis requires recalculation. An example family of treatment curves is given in Figure 6.10a-d for the main parametric case, allowing the treatment profile to be modified for a specific peak focal temperature, peak intracranial temperature or desired treatment volume.

While the analysis presented in this paper has concentrated on a single transcranial sonication of the brain, the size of the focal spot produced means that most practical treatments would require multiple sonications to treat a lesion within the brain. Some pause is therefore necessary between sonications so that the temperature within the skull can recover. Figure 6.11a shows the response of intracranial temperature to cooling before and after an individual sonication. Cooling may be performed passively into a static body of cold degassed water between the skull and transducer, or it may be performed actively by continuously exchanging and chilling this body of water. The cooling processes detailed in Figure 6.11a use degassed water at a temperature of 15°C. Active cooling is clearly superior in reducing temperature within the skull after the period of sonication. Furthermore, the temperature lapse rate with active cooling indicates that a

short recovery time may be possible between focused ultrasound sonications without accumulating excessive skull temperatures. For comparison, Figure 6.11b shows the cooling response in a denuded skull as might be achieved with an empirical specimen.

It is interesting to note that if the lowest published value [33] for thermal conductivity in bone ($k = 0.16$ W m^{-1} K^{-1}) had been germane in cranial bone in this study, it would have apparently precluded the possibility of transcranial focused ultrasound surgery given the unavoidability of thermal damage to bone at that conductivity. However, as described in Sections 6.3.8, more contemporary studies demonstrate values for thermal conductivity that around three times greater.

The thermal simulations presented here have assumed a constant and uniform blood perfusion in each tissue. Extending the thermal modeling to permit non-uniform perfusion is implementationally trivial. However, this would also require brain perfusion data which might be obtained from MRI studies, scalp perfusion data which might be obtained from Doppler flowimetry, and skull perfusion data which is presently uncharacterized. Skull temperature is most affected by changes in scalp perfusion; however removing scalp perfusion entirely only produces a difference of 6.5% in the change in peak temperature in the skull in the main parametric case. The effect of variation in perfusion is therefore small.

The technique of transmitting ultrasound at therapeutic intensity through an intact skull appears to be perfectly adequate for forming lesions towards the cranial base where the focus is well removed from the intervening skull in the acoustic path. As shown in Figure 6.4, it was not possible to produce a focus in the rostral location with a uniform amplitude excitation of the hemispherical array. However, we hypothesize that the development of a suitable amplitude shading algorithm will increase the intracranial volume within which transcranial sonication can safely produce small thermally coagulated lesions. The development of such an optimal amplitude shading algorithm is the subject of further study.

6.6 Conclusion

Transcranial focused ultrasound surgery is dependent on being able to transmit sufficient acoustic energy to a targeted focus in the brain to cause thermal coagulation without also inducing tissue damage within the surrounding brain or skull tissue. The nature of thermal deposition in the skull during transcranial focused ultrasound surgery has long been unclear. Skull temperature cannot be easily measured in this circumstance with thermocouples due to variation in heating with the skull and the difficulty in implanting thermocouples in the acoustic path without adversely distorting acoustic propagation. Furthermore, non-invasive thermometry by proton resonance shift MRI is ineffective in skull due to the low water content of bone.

The simulation methods described in this paper allow straightforward prediction of both the acoustic field within cranial bone, as well as the thermal behavior of the skull and brain while under therapeutic focused ultrasound sonication. This study also demonstrates that therapeutic temperatures may be reached in the region of brain corresponding to skull-base tumors with transcranial focused ultrasound using uniform pressure amplitude excitation of a large hemispherical ultrasound array without causing damaging temperature elevations in the skull, scalp or outer surface of the brain.

6.7 REFERENCES

[1] J.G. Lynn and T.J. Putnam, Histology of cerebral lesions produced by focused ultrasound *Am.J.Path.*, vol. 20, pp. 637-652, 1944.

[2] W.J. Fry, W. Mosberg, J.W. Barnard, and F.J. Fry, Production of focal destructive lesions in the central nervous system with ultrasound. *Journal of Neurosurgery*, vol. 11, pp. 471-478, 1954.

[3] P.P. Lele, Production of deep focal lesions by focused ultrasound - current status *Ultrasonics*, vol. 5, pp. 105-122, 1967.

[4] A. Morita, L.N. Sekhar, and D.C. Wright, Current concepts in the management of tumors of the skull base *Cancer Control*, vol. 5, pp. 138-149, 1998.

[5] F. Fry, Transskull transmission of axisymmetric focused ultrasonic beams in the 0.5 to 1 MHz frequency range: implications for brain tissue visualization, interrogation, and therapy *Ultrasonic Tissue Characterization*, vol. 2, pp. 203-208, 1979.

[6] J. Sun and K.H. Hynynen, Focusing of therapeutic ultrasound through a human skull: a numerical study *Journal of the Acoustical Society of America*, vol. 104, pp. 1705-1715, 1998.

[7] K. Hynynen and F.A. Jolesz, Demonstration of potential noninvasive ultrasound brain therapy through an intact skull *Ultrasound in Medicine and Biology*, vol. 24, pp. 275-283, 1998.

[8] M. Tanter, J.L. Thomas, and M. Fink, Focusing and steering through absorbing and aberrating layers: application to ultrasonic propagation through the skull *Journal of the Acoustical Society of America*, vol. 103, pp. 2403-2410, 1998.

[9] G.T. Clement and K. Hynynen, A noninvasive method for focusing ultrasound through the human skull *Phys Med Biol*, vol. 47, pp. 1219-1236, 2002.

[10] N.I. Vykhodtseva, V. Sorrentino, F.A. Jolesz, R.T. Bronson, and K. Hynynen, MRI detection of the thermal effects of focused ultrasound on the brain *Ultrasound in Medicine and Biology*, vol. 26, pp. 871-880, 2000.

[11] N.B. Smith, J.M. Temkin, F. Shapiro, and K. Hynynen, Thermal effects of focused ultrasound energy on bone tissue *Ultrasound in Medicine and Biology*, vol. 27, pp. 1427-1433, 2001.

[12] N. McDannold, R.L. King, F.A. Jolesz, and K. Hynynen, The use of quantitative temperature images to predict the optimal power for focused ultrasound surgery: in vivo verification in rabbit muscle and brain *Med Phys*, vol. 29, pp. 356-365, 2002.

[13] G.T. Clement and K. Hynynen, Correlation of ultrasound phase with physical skull properties *Ultrasound in Medicine and Biology*, vol. 28, pp. 617-624, 2002.

[14] B.K.P. Horn, Closed-form solution of absolute orientation using unit quaternions *Journal of the Optical Society of America A*, vol. 4, pp. 629-642, 1986.

[15] J. Sun and K.H. Hynynen, The potential of transskull ultrasound therapy and surgery using the maximum available skull surface area *Journal of the Acoustical Society of America*, vol. 105, pp. 2519-2527, 1999.

[16] G.M.J. Van Leeuwen, J.W. Hand, J.J.W. Lagendijk, D.V. Azzopardi, and A.D. Edwards, Numerical modeling of temperature distributions within the neonatal head *Pediatric Research*, vol. 48, pp. 351-356, 2000.

[17] K. Hynynen, The role of nonlinear ultrasound propagation during hyperthermia treatments *Med Phys*, vol. 18, pp. 1156-1163, 1991.

[18] P.J. Westervelt, Parametric Acoustic Array *Journal of the Acoustical Society of America*, vol. 35, pp. 535-537, 1963.

[19] P.P. Delsanto, R.S. Schechter, and R.B. Mignogna, Connection machine simulation of ultrasonic wave propagation in materials III: The three-dimensional case *Wave Motion*, vol. 26, pp. 329-339, 1997.

[20] C.W. Connor and K. Hynynen, Bio-acoustic thermal lensing and nonlinear propagation in focused ultrasound surgery using large focal spots: a parametric study *Phys Med Biol*, vol. 47, pp. 1911-1928, 2002.

[21] C.W. Connor, G.T. Clement, and K. Hynynen, A unified model for the speed of sound in cranial bone based on genetic algorithm optimization *Phys Med Biol*, vol. 47, pp. 3925-3944, 2002.

[22] C.W. Connor, G.T. Clement, and K. Hynynen, A unified model for the variation in amplitude attenuation coefficient in cranial bone with local density and ultrasound frequency based on genetic algorithm optimization. *In Submission*, vol. 2002.

[23] D.T. Blackstock, Thermoviscous attenuation of plane, periodic, finite-amplitude sound waves *Journal of the Acoustical Society of America*, vol. 36, pp. 534-542, 1964.

[24] A. Gueziec and G. Taubin, Multi-resolution modeling and 3D geometry compression *Computational Geometry*, vol. 14, pp. 1-3, 1999.

[25] L. Kobbelt, J. Vorsatz, and H.-P. Seidel, Multiresolution hierarchies on unstructured triangle meshes *Computational Geometry*, vol. 14, pp. 5-24, 1999.

[26] D. Doo and M. Sabin, Behaviour of recursive division surfaces near extraordinary points *Computer Aided Design*, vol. 10, pp. 356-360, 1978.

[27] K.B. Ocheltree and L.A. Frizzell, Determination of power deposition patterns for localized hyperthermia: a steady-state analysis. *International Journal of Hyperthermia*, vol. 3, pp. 269-279, 1987.

[28] K.B. Ocheltree and L.A. Frizzell, Determination of power deposition patterns for localized hyperthermia: a transient analysis. *International Journal of Hyperthermia*, vol. 4, pp. 281-296, 1988.

[29] H.H. Pennes, Analysis of tissue and arterial blood temperatures in the resting human forearm *Journal of Applied Physiology*, vol. 1, pp. 93-122, 1948.

[30] A.D. Pierce. *Acoustics: an introduction to its physical principles and applications*, 1989.

[31] R. Clattenburg, J. Cohen, S. Conner, and N. Cook, Thermal properties of cancellous bone *Journal of Biomedical Materials Research*, vol. 9, pp. 169-182, 1975.

[32] S.R.H. Davidson and D.F. James, Measurement of thermal conductivity of bovine cortical bone *Medical Engineering & Physics*, vol. 22, pp. 741-747, 2000.

[33] S. Biyikli, M.F. Modest, and R. Tarr, *Journal of Biomedical Materials Research*, vol. 20, pp. 1335-1345, 1986.

[34] J.C. Chato. Heat transfer in bioengineering. In: *Advanced Heat Transfer*, ed. B.T. Chao. Urbana: University of Illinois Press, 1969.pp. 404-412.

[35] E.S. Zelenov, Experimental investigation of the thermophysical properties of compact bone *Mechanics of Composite Materials*, vol. 21, pp. 759-762, 1985.

[36] H.F. Bowman, Thermal conductivity in bovine skull ed. Personal Communication. May 3, 2002.

[37] S. Sapareto and W. Dewey, Thermal dose determination in cancer therapy *Int J Radiation Oncology Biol.Phys.*, vol. 10, pp. 787-800, 1984.

[38] J. Nolte. *The Human Brain* , St.Louis, Missouri: Mosby Inc., 1998.

[39] M.C. Kolios, M.D. Sherar, and J.W. Hunt, Large blood vessel cooling in heated tissues: a numerical study *Phys Med Biol*, vol. 40, pp. 477-494, 1995.

[40] F. Duck. *Physical properties of tissue: a comprehensive reference book*, London: Academic Press, 1990.

[41] International Commission on Radiation Units and Measurements, ICRU Report 61: Tissue Substitutes, Phantoms and Computational Modelling in Medical Ultrasound, ICRU Publications, Bethesda, Maryland, 1998.

[42] F.W. Kremkau, R.W. Barnes, and C.P. McGraw, Ultrasonic attenuation and propagation speed in normal human brain *Journal of the Acoustical Society of America*, vol. 70, pp. 29-38, 1981.

[43] S.A. Goss, L.A. Frizzell, and F. Dunn, Ultrasonic absorption and attenuation in mammalian tissues *Ultrasound in Medicine and Biology*, vol. 5, pp. 181-186, 1979.

[44] T.E. Cooper and G.J. Trezek, A probe technique for determining the thermal conductivity of tissue. *Journal of Heat Transfer*, vol. 94, pp. 133-140, 1972.

[45] S. Vietla, R.C. Eberhart, and D.M. Meyer, The influences of hypothermia and tissue perfusion on temperature distributions in simulated cranial surgery ed. R.B. Roemer. Advances in Bioheat and Mass Transfer, pp. 9-17, 1993. American Society of Mechanical Engineers. New York.

[46] F.C. Henriques and A.R. Moritz, Studies of thermal injury I. the conduction of heat to and through skin and the temperatures attained therein. A theoretical and experimental investigation. *Am.J.Path.*, vol. 23, pp. 531-549, 1947.

[47] C.J. Henschel, Heat impact of revolving instruments on vital dentin tubules *J Dent Res*, vol. 22, pp. 323-333, 1943.

[48] L.R. Williams and R.W. Leggett, Reference values for resting blood flow to organs of man. *Clinical Physics and Physiological Measurement*, vol. 10, pp. 187-217, 1989.

6.8 TABLES

Table 6.1: Acoustic properties of cranial bone based on apparent CT density.

Table 6.2: Material and acoustic properties for water and brain.

Table 6.3: Uniform excitation pressure over the 500-element array to produce a peak temperature of 70°C after a 20 second sonication.

Table 6.1

ρ_{ct} (kg m^{-3})	c (m s^{-1})	α (Np m^{-1})
1000	1852.7	512.1
1100	2033.0	455.7
1200	2183.8	400.1
1300	2275.8	346.4
1400	2302.6	295.4
1500	2298.1	248.0
1600	2300.3	205.1
1700	2332.5	167.7
1800	2393.4	136.5
1900	2479.0	112.6
2000	2585.4	96.8
2100	2708.8	90.0
2200	2845.3	92.9
2300	2991.1	103.8
2400	3142.1	119.7
2500	3294.6	137.6
2600	3444.7	154.8
2700	3588.5	168.2
2800	3722.2	175.5
2900	3841.9	176.6
3000	3947.6	172.5

Table 6.2

	Degassed Water	Brain	Scalp	Skull
ρ (kg m^{-3})	1.000 [40]	1.3	1.09 [41]	From CT Data
c (m s^{-1})	1500 [40]	1562 [42]	1615 [41]	From Table 6.1
α (Np m^{-1})	2.88×10^{-4} [40]	2.9 [43]	3.22 [41]	From Table 6.1
k (W m^{-1} K^{-1})	0.615 [40]	0.528 [44]	0.34 [45]	0.43 [36]
c_v (J kg^{-1} K^{-1})	4180 [40]	3640 [44]	3530 [46]	1440 (trabecular) [31] 1300 (cortical) [47]
W (ml ml^{-1} s^{-1})	0 (by definition)	8.33×10^{-3} [48]	1.5×10^{-3} [48]	3.33×10^{-4} [48]

Table 6.3

Calvarium and Orientation		Dose Excitation Pressure
(a)	Thin calvarium, central	54.3 kPa
(b)	Intermediate calvarium, central	68.8 kPa
(c)	Thick calvarium, central	82.6 kPa
(d)	Intermediate calvarium, near lateral	64.3 kPa
(e)	Intermediate calvarium, far lateral	59.8 kPa
(f)	Intermediate calvarium, rostral	N/A

6.9 FIGURES

Figure 6.1: 3D reconstruction of calvarium mounted in the 500-element hemispherical array.

Figure 6.2: The intersection of acoustic simulation volumes, shown for a 64-element hemispherical array.

Figure 6.3: Distribution of acoustic pressure within the scalp, calvarium and brain.
(a) Pressure amplitude windowed to 1MPa to show the overall field.
(b) Pressure amplitude windowed to 8.5MPa to emphasize the focal field.

Figure 6.4: Thermal deposition during transcranial focused ultrasound with active pre-cooling.
(a) Thin calvarium.
(b) Intermediate thickness calvarium.
(c) Thick calvarium.
(d) Intermediate thickness calvarium with near lateral displacement of focus.
(e) Intermediate thickness calvarium with far lateral displacement of focus.
(f) Intermediate thickness calvarium with rostral displacement of focus.

Figure 6.5: Patterns of thermal deposition within the skull with active pre-cooling.
(a) Thin calvarium.
(b) Intermediate thickness calvarium.
(c) Thick calvarium.
(d) Intermediate thickness calvarium with near lateral displacement of focus.
(e) Intermediate thickness calvarium with far lateral displacement of focus.
(f) Intermediate thickness calvarium with rostral displacement of focus.

Figure 6.6: Patterns of thermal deposition within the denuded skull with active pre-cooling.
(a) Thin calvarium.
(b) Intermediate thickness calvarium.
(c) Thick calvarium.
(d) Intermediate thickness calvarium with near lateral displacement of focus.
(e) Intermediate thickness calvarium with far lateral displacement of focus.
(f) Intermediate thickness calvarium with rostral displacement of focus.

Figure 6.7: Patterns of thermal deposition within the denuded skull without pre-cooling.
(a) Thin calvarium.
(b) Intermediate thickness calvarium.
(c) Thick calvarium.
(d) Intermediate thickness calvarium with near lateral displacement of focus.
(e) Intermediate thickness calvarium with far lateral displacement of focus.
(f) Intermediate thickness calvarium with rostral displacement of focus.

Figure 6.8: Thermal distribution and temperature profile immediately after sonication in a skull of intermediate thickness.
(a) With scalp.
(b) Without scalp.

Figure 6.9: Location and heating effect of standing waves within the skull.
(a) The distribution of absorbed acoustic power, Q, within the skull.
(b) The thermal deposition caused within the skull by this pattern of acoustic energy absorption.

Figure 6.10: Dosage relationships for skull of intermediate thickness oriented centrally.
(a) Ultrasound pressure amplitude and focal temperature.
(b) Ultrasound pressure amplitude and volume of coagulation.
(c) Focal temperature and volume of coagulation.
(d) Focal temperature and peak intracranial temperature.

Figure 6.11: Effect of active and passive cooling on temperature within the skull against time.
(a) With scalp.
(b) Without scalp.

Figure 6.1



Figure 6.2

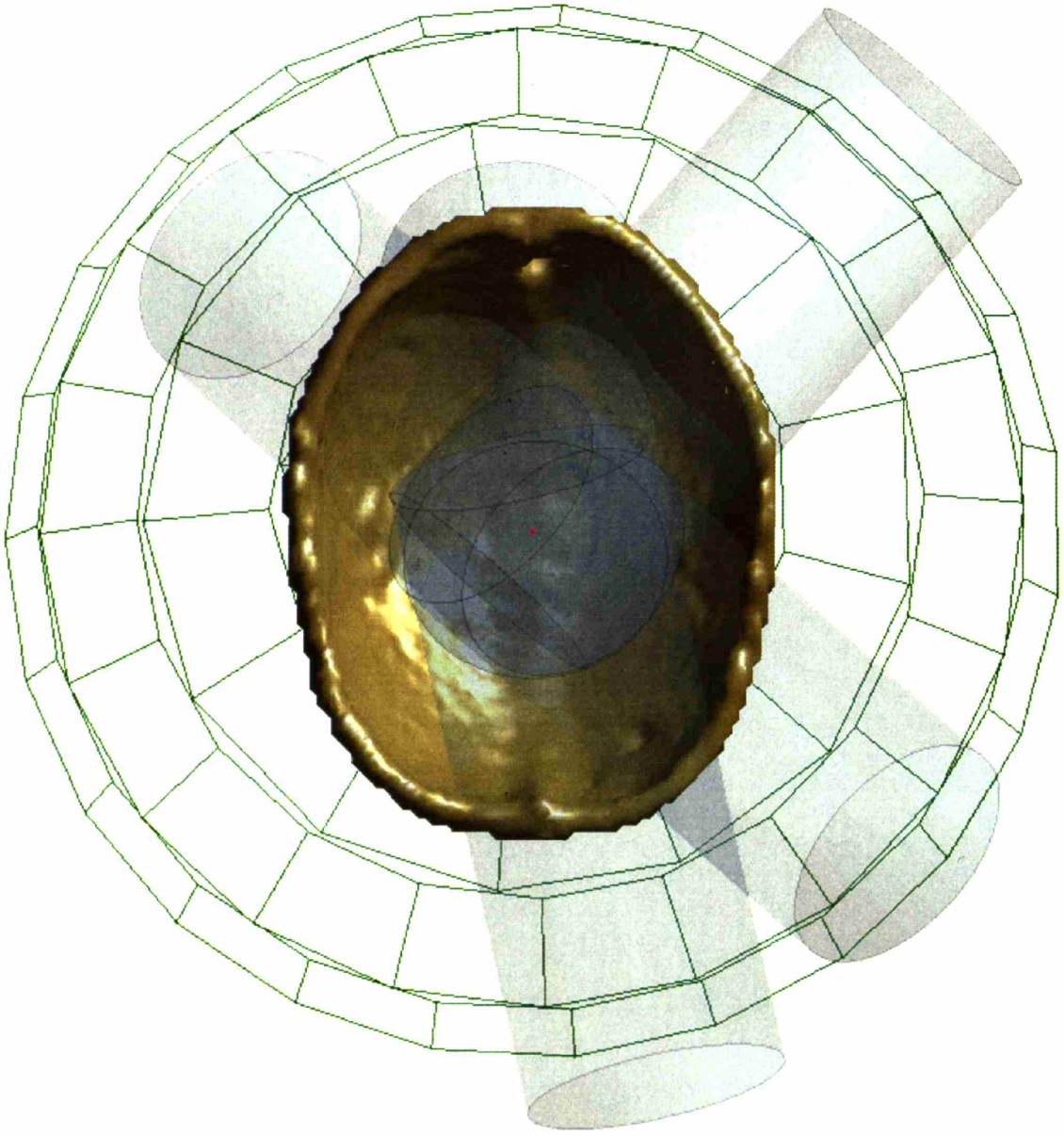


Figure 6.3

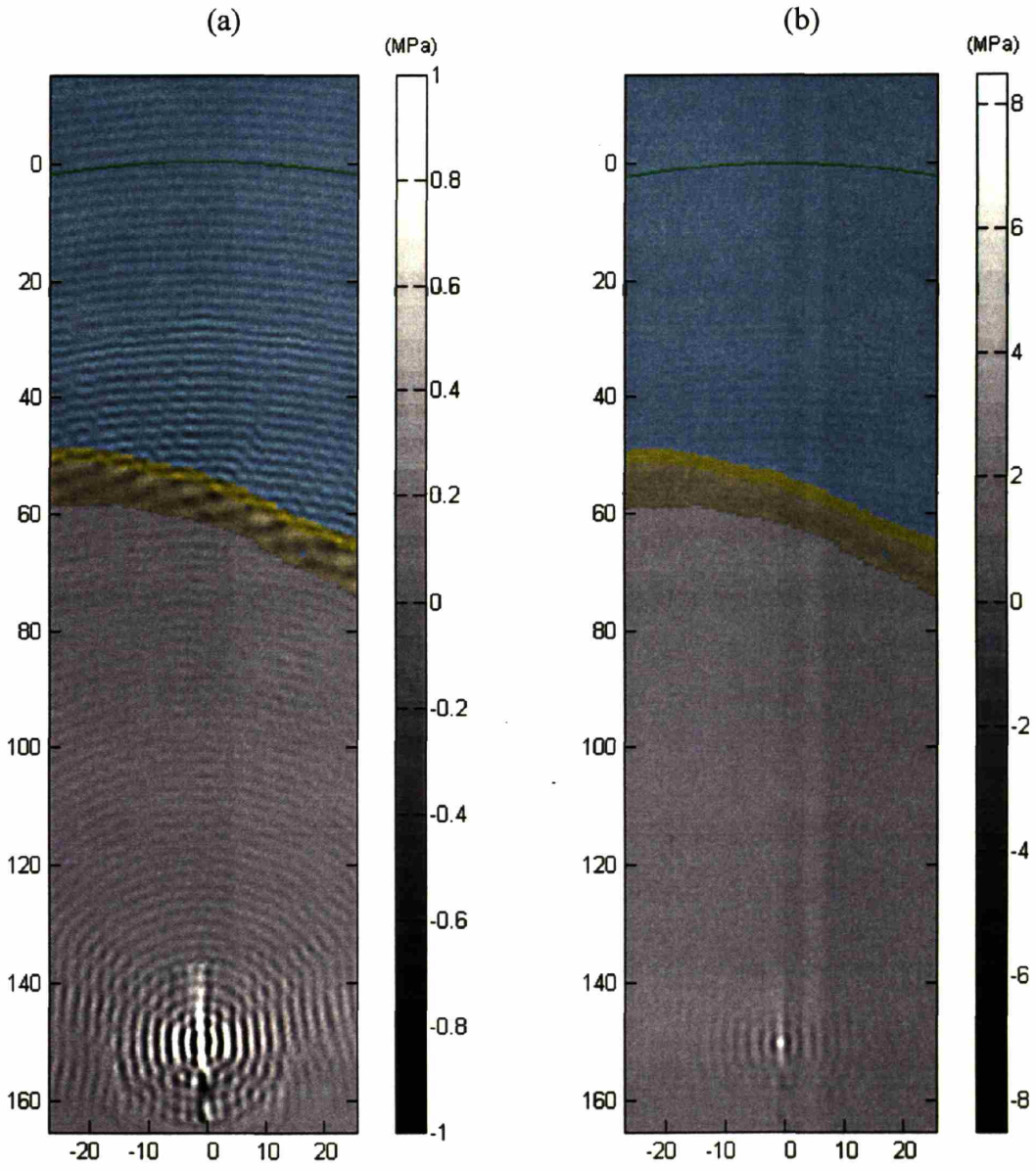

(a)
(MPa)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140
160
-20
-10
0
10
20
(b)
(MPa)
8
6
4
2
0
-2
-4
-6
-8
0
20
40
60
80
100
120
140
160
-20
-10
0
10
20

Figure 6.4

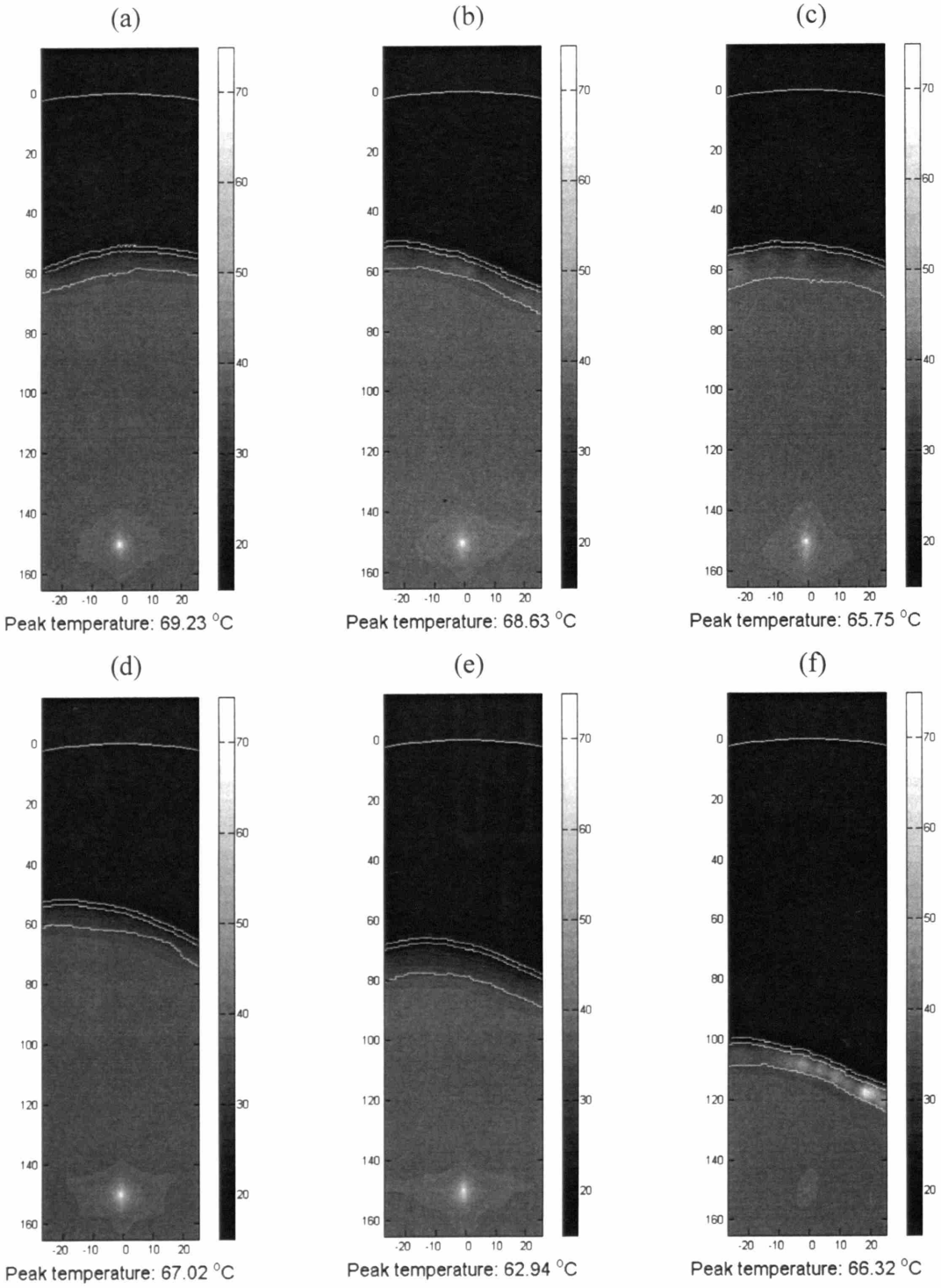

(a)
(b)
(c)
Peak temperature: 69.23 °C
Peak temperature: 68.63 °C
Peak temperature: 65.75 °C
(d)
(e)
(f)
Peak temperature: 67.02 °C
Peak temperature: 62.94 °C
Peak temperature: 66.32 °C

Figure 6.5

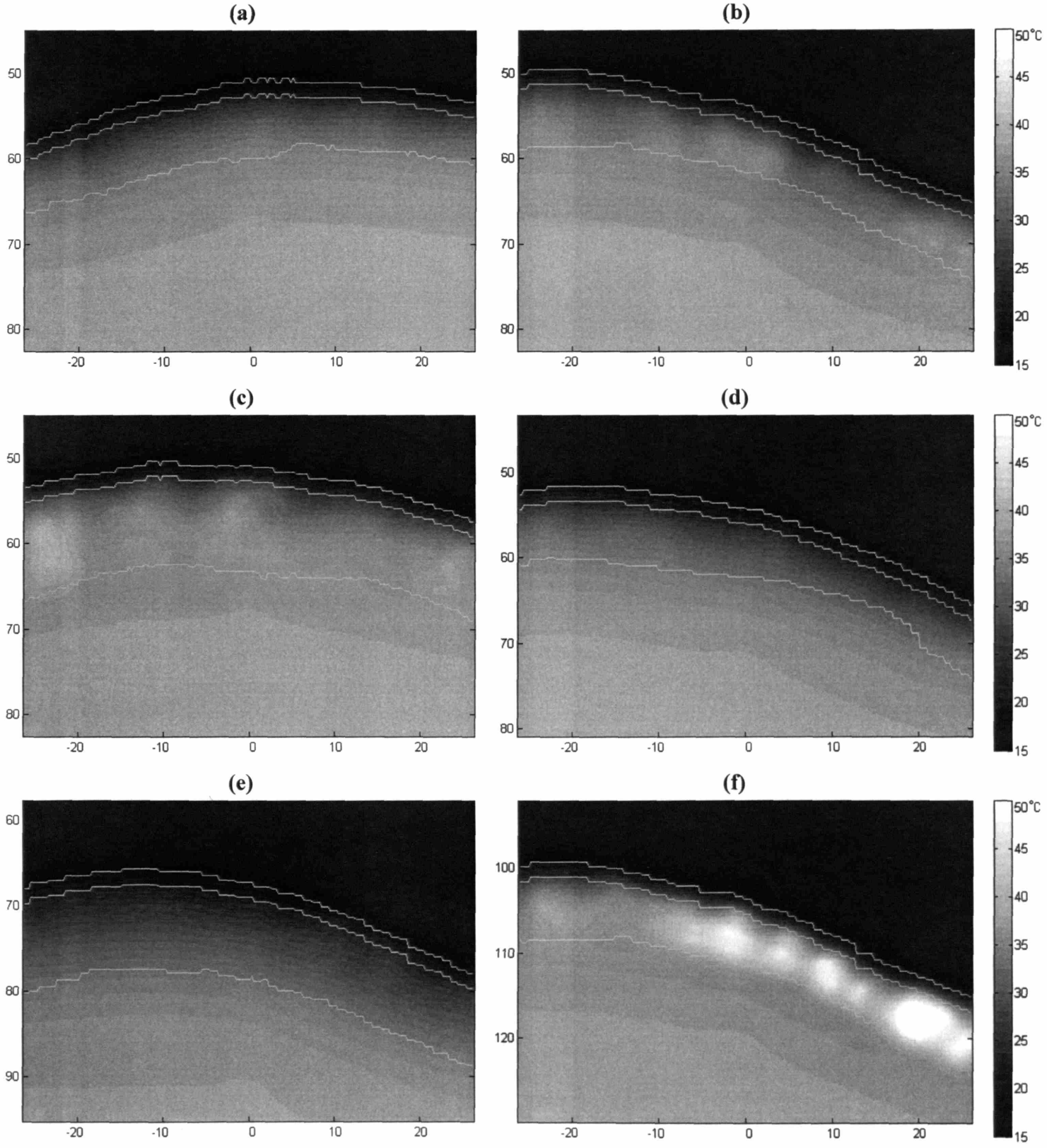

(a)
(b)
(c)
(d)
(e)
(f)
50
60
70
80
90
100
110
120
-20
-10
0
10
20
50°C
45
40
35
30
25
20
15

Figure 6.6

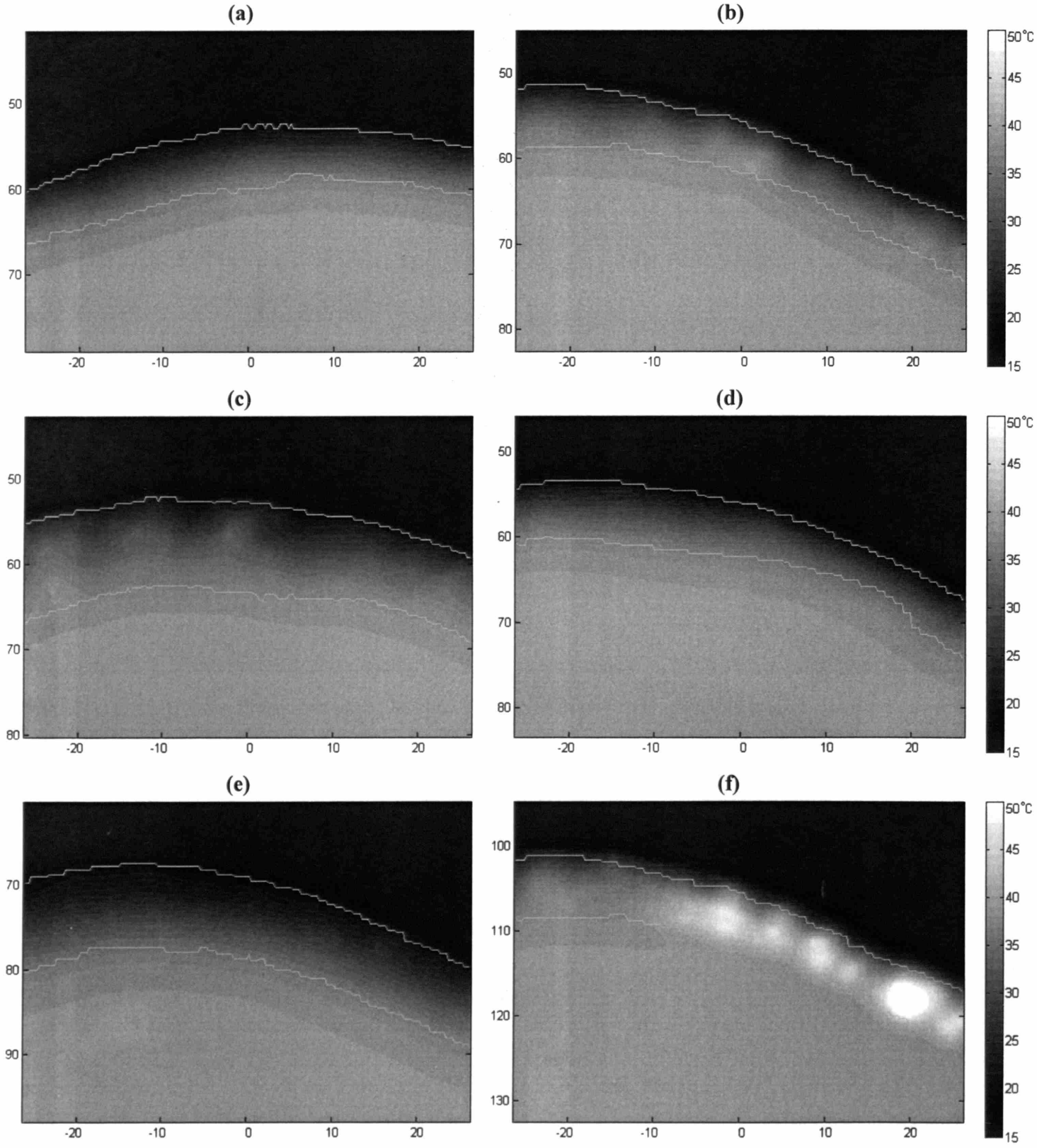

(a)
(b)
(c)
(d)
(e)
(f)
50
60
70
80
90
100
110
120
130
-20
-10
0
10
20
50°C
45
40
35
30
25
20
15

Figure 6.7

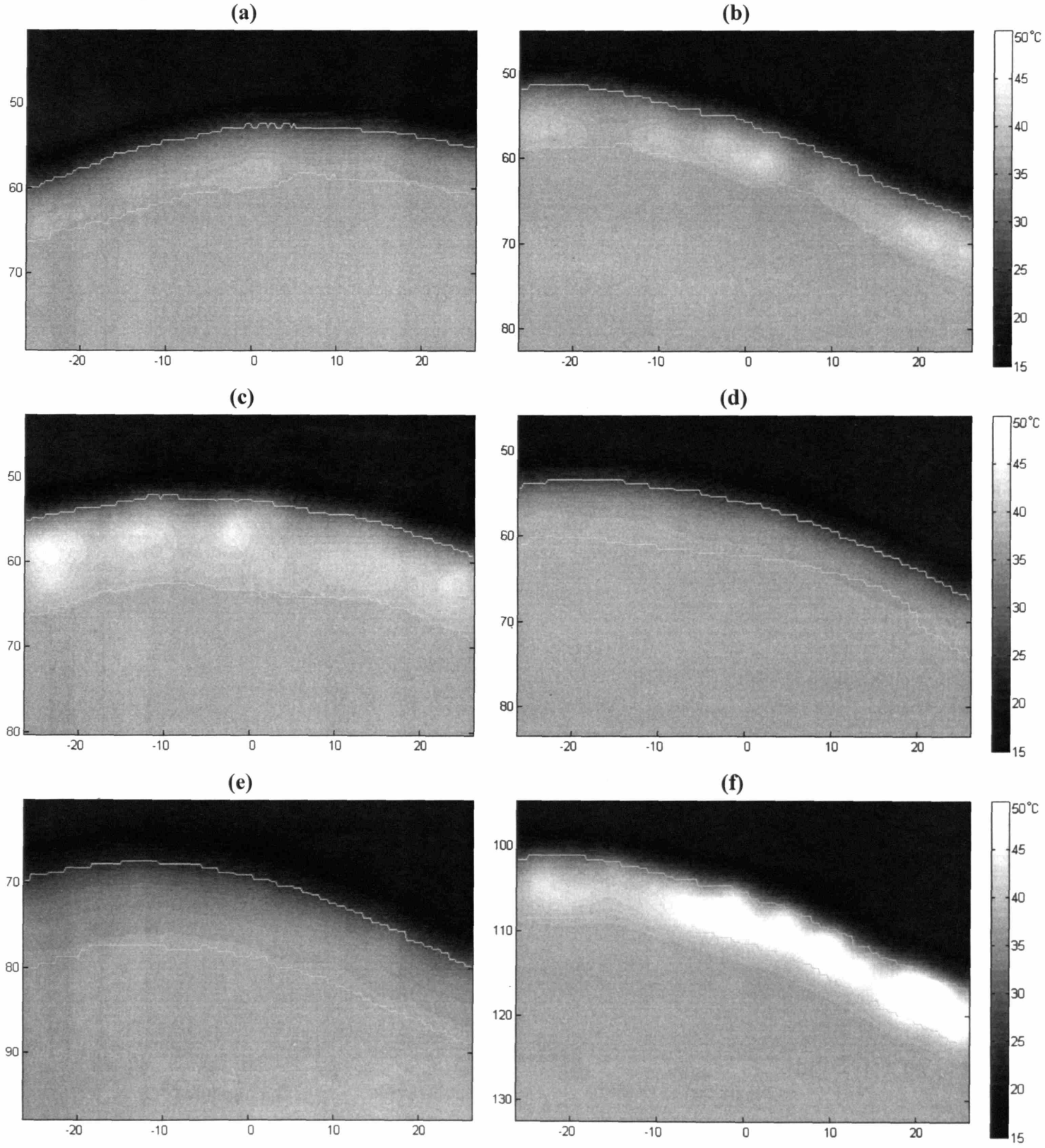

(a)
(b)
(c)
(d)
(e)
(f)
50
60
70
80
90
100
110
120
130
-20
-10
0
10
20
50°C
45
40
35
30
25
20
15

Figure 6.8

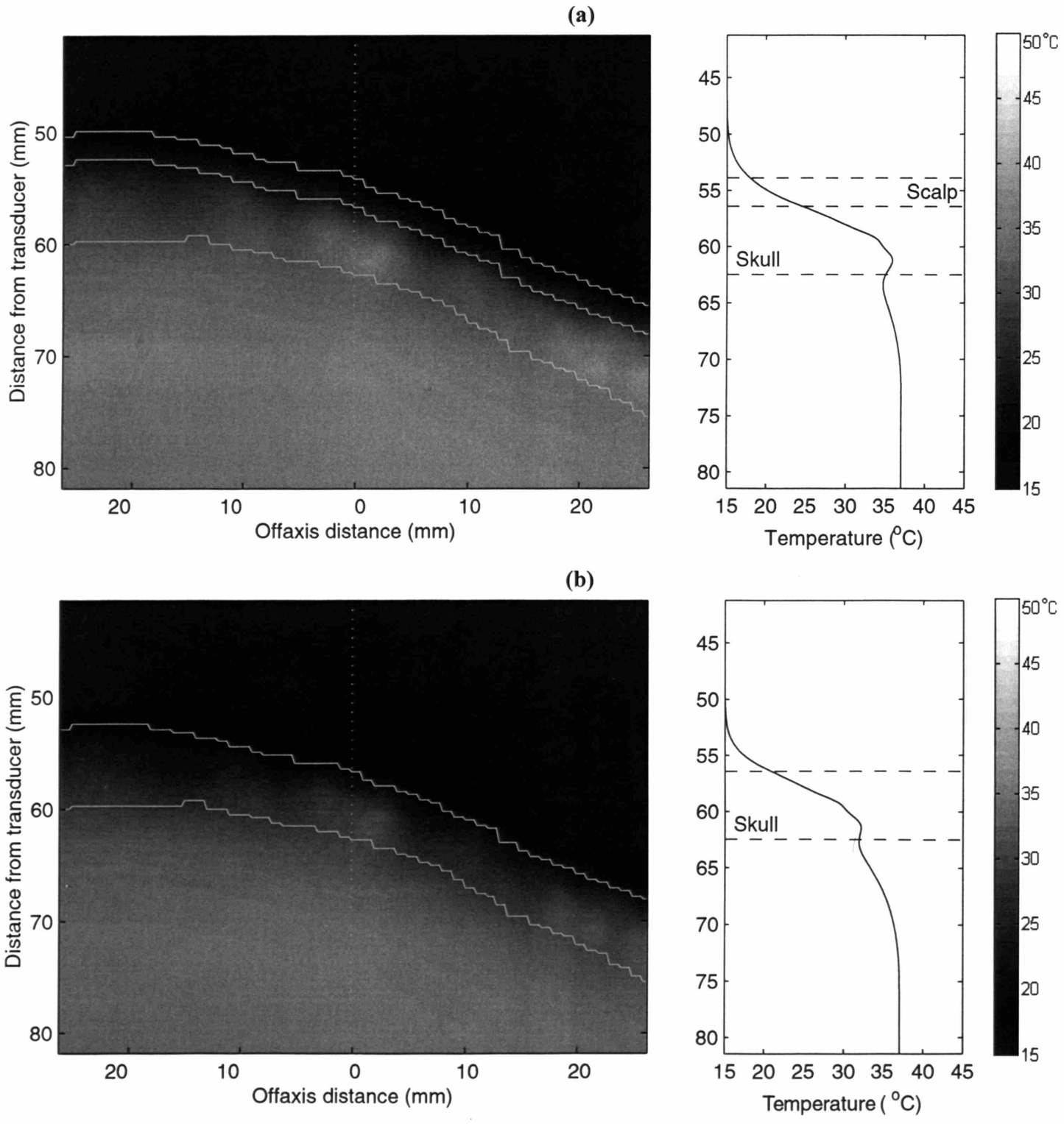

(a)
Distance from transducer (mm)
50
60
70
80
20
10
0
10
20
Offaxis distance (mm)
45
50
55
60
65
70
75
80
Scalp
Skull
15 20 25 30 35 40 45
Temperature (°C)
50°C
45
40
35
30
25
20
15
(b)
Distance from transducer (mm)
50
60
70
80
20
10
0
10
20
Offaxis distance (mm)
45
50
55
60
65
70
75
80
Skull
15 20 25 30 35 40 45
Temperature (°C)
50°C
45
40
35
30
25
20
15

Figure 6.9

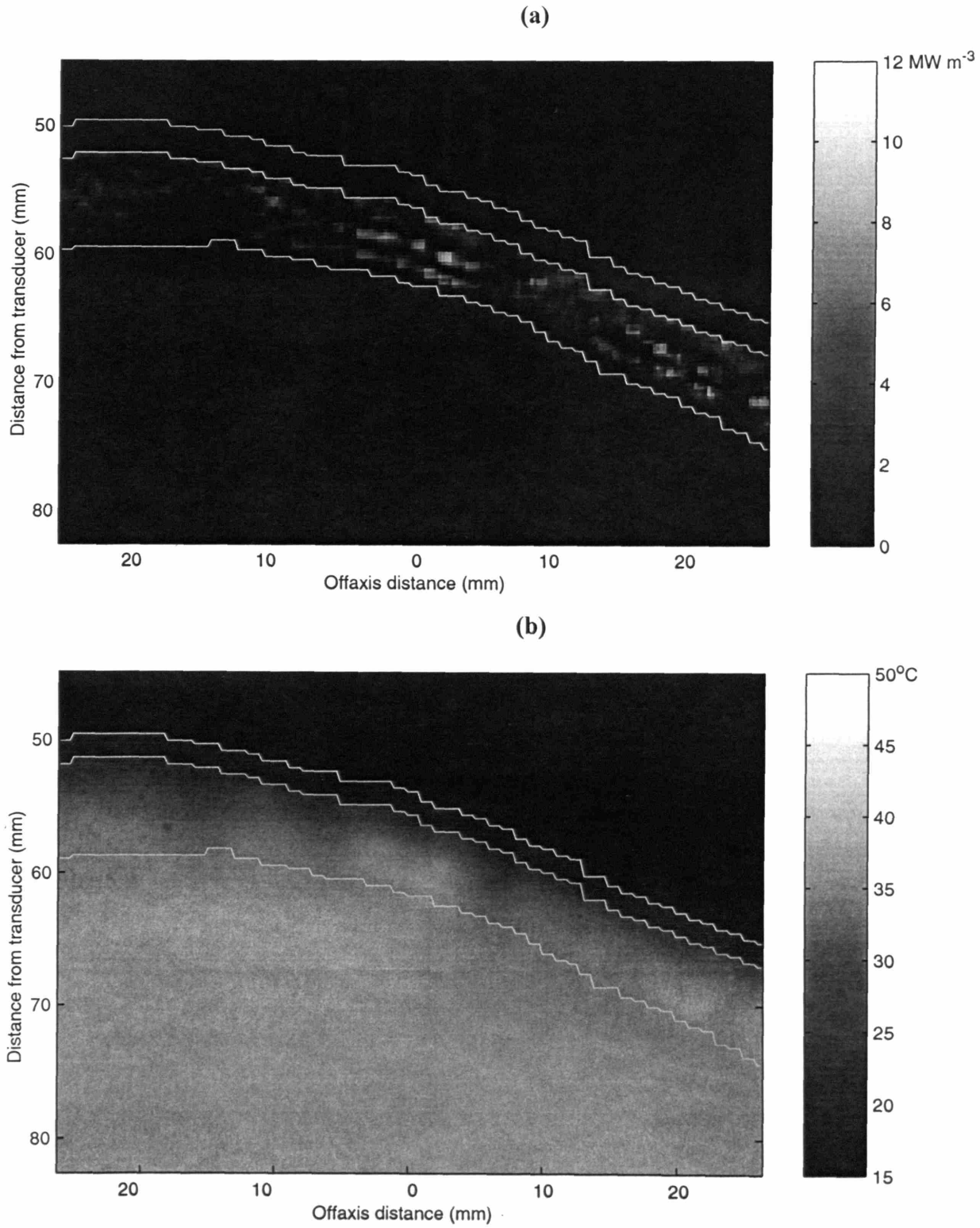

(a)
12 MW m-3
10
8
6
4
2
0
50
60
70
80
Distance from transducer (mm)
20
10
0
10
20
Offaxis distance (mm)
(b)
50°C
45
40
35
30
25
20
15
50
60
70
80
Distance from transducer (mm)
20
10
0
10
20
Offaxis distance (mm)

Figure 6.10

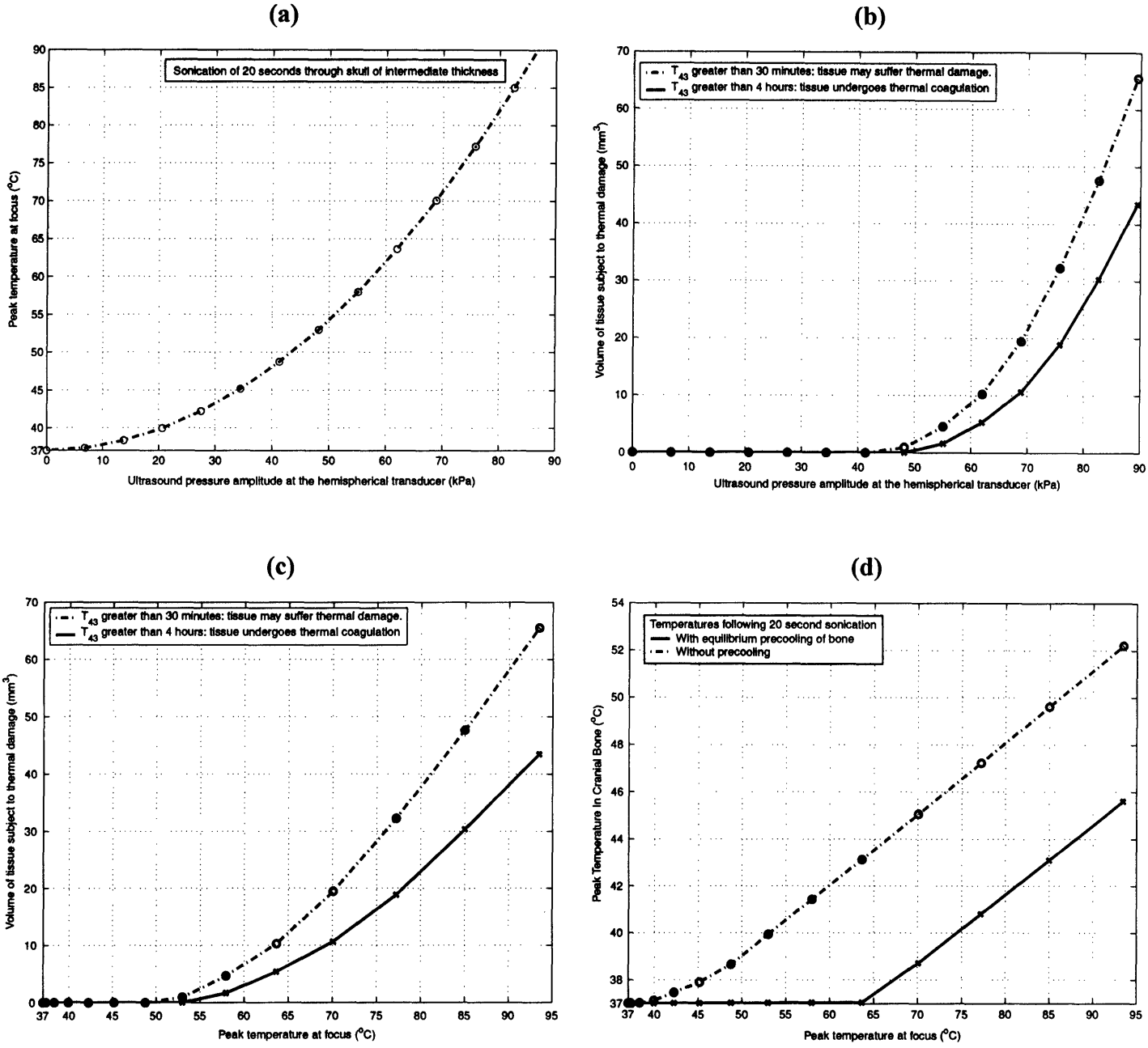
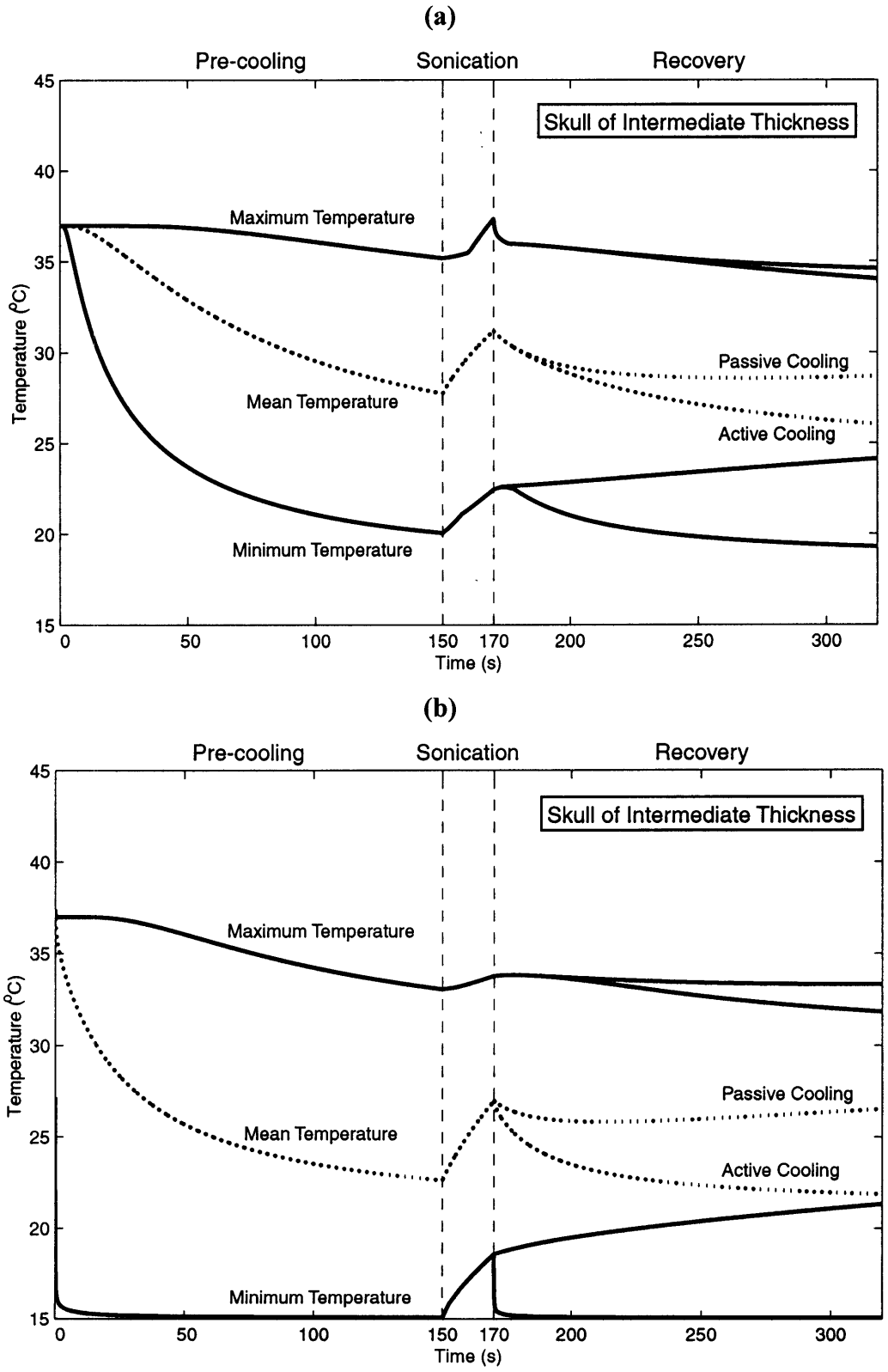

(a)
Sonication of 20 seconds through skull of intermediate thickness
Peak temperature at focus (°C)
Ultrasound pressure amplitude at the hemispherical transducer (kPa)
(b)
T43 greater than 30 minutes: tissue may suffer thermal damage.
T43 greater than 4 hours: tissue undergoes thermal coagulation
Volume of tissue subject to thermal damage (mm3)
Ultrasound pressure amplitude at the hemispherical transducer (kPa)
(c)
T43 greater than 30 minutes: tissue may suffer thermal damage.
T43 greater than 4 hours: tissue undergoes thermal coagulation
Volume of tissue subject to thermal damage (mm3)
Peak temperature at focus (°C)
(d)
Temperatures following 20 second sonication
With equilibrium precooling of bone
Without precooling
Peak Temperature in Cranial Bone (°C)
Peak temperature at focus (°C)

Figure 6.11


(a)
Pre-cooling
Sonication
Recovery
Skull of Intermediate Thickness
Maximum Temperature
Mean Temperature
Minimum Temperature
Passive Cooling
Active Cooling
Temperature (°C)
Time (s)
45
40
35
30
25
20
15
0
50
100
150
170
200
250
300
(b)
Pre-cooling
Sonication
Recovery
Skull of Intermediate Thickness
Maximum Temperature
Mean Temperature
Minimum Temperature
Passive Cooling
Active Cooling
Temperature (°C)
Time (s)

Chapter 7

Future Directions

The research presented in the previous chapters has suggested some possible lines of future academic enquiry.

7.1 VALIDATING AND INCREASING THE SPEED OF ANALYSIS TOWARDS ON-LINE USE OF 3D MODELS.

The speed of sound and acoustic attenuation models presented here have been validated under the well-controlled experimental conditions detailed in Chapter 4 and Chapter 5. These results anticipate application to full 3D simulation of transcranial acoustic propagation in a clinical setting. However, the performances of these models under the increased vagaries of a real clinical setting have yet to be ascertained. Current difficulties in this area include obtaining sufficiently robust registration, and in maintaining this registration through the long calculation time inherent in 3D modeling. Alleviating these difficulties remains an area of active research.

Much of the research presented here has made use of the technique of distributed computation to make the analysis of large acoustic and thermal problems tractable. The tools necessary to do this were developed by the author. Fortunately, robust public domain code libraries now exist for developing coarse-grained distributed computing on standard desktop hardware. In particular, the MPI library is implemented for both UNIX and PC environments and enjoys wide support in many academic supercomputing centers. Consequently, future development should probably be migrated to this platform instead of continuing to pursue a bespoke approach to distributed computing. Using unremarkable contemporary PC hardware, the acoustic simulations described in Chapter 6 take approximately one hour per acoustic element. Hence, an unremarkable collection of ten desktop PCs can calculate the acoustic field due to the five-hundred element hemispherical array in approximately two days. Given that the problem can be readily coarsely parallelized and distributed over as many computers as there are transducer elements, it is apparent that this time scale could be reduced to the point where it might eventually be practical to use 3D acoustic modeling to determine acoustic field corrections on-line in transcranial focused ultrasound surgery.

Improvements in processing speed are not only obtained by increasing the availability of computing power; it may be possible to achieve further increases in processing speed through the efficient coupling of analytical techniques to numerical methods. The combination of the hybrid Huygens-FDTD algorithm with multiresolutioning of the transducer (Section 6.3.4 and 6.3.5) produced an almost tenfold increase in processing speed over the FDTD algorithm alone. Similarly, the thermal analysis might be greatly increased in speed by replacing the current explicit implementation of the Pennes Bioheat Equation with an implicit formulation. While implicit algorithms tend to be computationally more complex, they allow much greater time steps without becoming unstable. Since the time scales involved in ultrasound sonication and cooling are of the order of hundreds of seconds, an implicit algorithm may present a substantial processing advantage.

7.2 THE VARIATION OF SPEED OF SOUND WITH FREQUENCY IN BONE.

Chapter 4 details the genetic optimization of the variation of speed of sound in bone dependent on apparent CT density. This relationship appears to have a dependence on the transition from trabecular to cortical structure in bone. The speed of sound curve in bone was determined for a single ultrasound frequency of 0.74 MHz. It is interesting to speculate how the relationship between speed of sound and apparent density might change with frequency. Any change is unlikely to be as pronounced as the variation in acoustic attenuation coefficient with frequency as seen in Chapter 5. However, it may be hypothesized that the apparent transition from trabecular to cortical behavior in the speed of sound curve would occur at a higher apparent density as the frequency of the ultrasound is increased. Bone is a cellular solid, composed of open and closed pores in a lattice framework. In regions of intermediate porosity, the pores can only be considered to be open or closed in relation to the wavelength of the ultrasound propagating through the material. Thus we would expect the transition density to increase with frequency since increased density corresponds to decreased pore size in the trabecular lattice, and increased frequency corresponds to decreased ultrasound wavelength in the cranial bone.

7.3 The variation of speed of sound and acoustic attenuation coefficient with apparent density in bone with orientational dependency.

The internal trabecular architecture of a bone is dynamically remodeled to provide maximum strength along the predominant weight-bearing axis of that bone. The bones of the skull, however, are not weight-bearing and so their trabeculae have no predominant orientation. This has simplified the analysis of the material properties of the bones of the skull since anisotropy in the speed of sound and acoustic absorption coefficient may then be neglected.

There is substantial research interest in using ultrasound to assess the material properties of bone non-invasively *in vivo*. This technique would be useful in such areas as screening for osteoporosis or assessing the healing of bone fractures, replacing the use of Dual X-Ray Absorption (DXA) scanning. However, this would require the sonication of bones such as the calcaneus or femur that are evidently weight-bearing, and in which one finds a predominant trabecular orientation and variation in acoustic properties with the direction of sound propagation. Towards this end, it might be interesting to attempt to determine how the speed of sound and acoustic absorption coefficient vary with apparent bone density when the role of orientation is significant. If this investigation were conducted with bone samples demonstrating various degrees of osteoporotic degeneration, it is not inconceivable that the results might ultimately provide a clinically useful index for staging the progress of osteoporosis.

7.4 Formulating an amplitude shading algorithm for transcranial ultrasound surgery

The studies in Chapter 6 used a uniform pressure amplitude excitation across the surface of the hemispherical transducer. It is clearly desirable to use the largest possible transducer for transcranial ultrasound surgery to obtain the largest geometric gain. However, it is not clear how such an array should be driven to produce the best thermal outcome. In Chapter 6, it was not possible to produce a therapeutic focus in the targeted

rostral position as the heating within the skull was too great. While a uniform excitation of the array might be expected to produce the least heating of the skull, this approach is overly simplistic on three counts:

1. The skull is not a uniform distance from the array even when mounted centrally to it as the skull is irregularly shaped. The above approach implicitly assumes radial symmetry in the skull, and is clearly flawed when the position of the skull is not central to the array.
2. Standing waves can be formed within the skull, causing substantial local variation in heating within the skull. The positions of these standing waves are difficult to predict since they are dependent on the interaction of ultrasound at a particular frequency with the variation in speed of sound and density within the skull.
3. The acoustic attenuation coefficient varies within the skull, producing local variation in the absorbed acoustic power.

The peak temperature reached in skull during a transcranial sonication of the brain is dependent on both the local heating of the skull and on the duration of the treatment. The duration of the treatment is dependent on the time it takes to heat the targeted focal brain tissue to the desired temperature; this is very much dependent on the rate at which energy can be transmitted to that focal point. Optimizing the efficiency with which energy is transmitted through the skull to the brain can therefore reduce the peak temperature reached within the skull. Consequently, the acoustic pressure amplitude would be increased in areas in which efficient transfer is occurring, and reduced in areas where there is inefficient transfer. This process of non-uniform amplitude excitation is known as amplitude shading. The development of a suitable amplitude-shading algorithm should increase the intracranial volume within which transcranial sonication can safely produce small thermally coagulated lesions. An optimal amplitude shading algorithm will need to consider both the local ultrasound intensity in the skull and also the rate of energy transmission across different parts of the skull. A balance between these two factors must be struck to optimize power transmission through the skull without creating excessive surface intensity in any particular place. This optimum balance is likely to be controlled by the maximum variation in acoustic intensity across the skull that can be tolerated before localized surface hotspots predominate the thermal case.

7.5 DETERMINING THERAPEUTIC LIMITS FOR TRANSCRANIAL FOCUSED ULTRASOUND SURGERY

As indicated in Chapter 6, a family of treatment curves can be produced for a particular skull and transducer configuration allowing peak temperatures, therapeutic effect and transducer excitation to be correlated and controlled.

As we approach the clinical use of transcranial ultrasound surgery, further studies of thermal dosimetry will be required to determine therapeutic limits and optimum treatment configuration in an on-line fashion. Furthermore, since most therapeutic transcranial ultrasound procedures will require multiple adjacent foci, these dosimetric studies will have to include modeling of repetitive treatments. It would also be very useful to be able to define the areas of the brain that are accessible to transcranial ultrasound therapy, although such a determination would evidently be dependent on the transducer geometry and on the pattern of excitation as described in Section 7.4 above.

The ultimate goal in the acoustic calculations associated with transcranial focused ultrasound surgery is to produce a perfectly coherent constructive interference at the targeted point within the brain. Indeed, this goal underpinned the genetic algorithm optimization of the speed of sound in bone in Chapter 4, in which the fitness of a particular speed of sound model was measured by its ability to produce the phase corrections that would have given this coherent interference. However, generating such a perfect coherence in clinical practice would require a very thorough CT scan of the patient's skull and extremely accurate knowledge of its positioning relative to the transducer. While these pieces of information can be gleaned in principle, it would almost certainly also be instructive to determine the theoretical performance of transcranial ultrasound surgery as this perfectly coherent interference is degraded.

Chapter 8

Acoustic Simulation Source Code

8.1 Source Code For A 3D Acoustic Solver Based Upon The Huygens-Fresnel Integral And The FDTD Formulation Of The Westervelt Equation.

Alpha3D.cpp Page 1/2

```
// Alpha3D.cpp
// Calculates the absorption coefficient at a mesh point given the
// mesh segmentation data and (depending on the material) the density.

#include "Alpha3D.h"

IFP Alpha3D(int index, float *rho, unsigned char *seg, float freq) {

  /*
    Choose the algorithm for determining the absorption coefficient
    The material types are:
    0: Outside the original data set
    1: Water
    2: Outer cortical bone
    3: Trabecular bone
    4: Inner cortical bone
    5: Brain
    6: User-specified - often fat
    7: User-specified - often liver
    8: User-specified

    This routine may be extended later to accommodate more materials.
  */

  IFP thisAlpha,bias;
  int lookup;

  // Use this line if you want absorptions specified per MHz
  // freq=(IFP)(freq/1e6); // Convert frequency to MHz

  // Use this line if you want to specify absorptions directly
  freq=1;

  switch (seg[index]) {
  case 0: // Outside the data set
    return (IFP)(alphaExtra*freq);
    break;

  case 1: // In water
    return (IFP)(alphaH2O*freq);
    break;

  case 2: // In bone (combined bone model)
  case 3:
  case 4:
    if (cBoneFunctionAlpha=0) {
      if (rho[index]≥1.6)
        // In cortical bone: from Fry, Barger (1978) and Lees (1986)
        thisAlpha=167;
      else
        // In trabecular bone, likewise Fry, Barger (1978)
        thisAlpha=305;
    } else {
      // Use the defined bone alpha function in the config file.
      // The function spans the range of rho from 1.0 to 3.4
      bias=(IFP)((rho[index]-1.0)/0.1);
      lookup=(int)floor((double)bias);
      bias=bias-lookup;
      if (lookup<0) {lookup=0; bias=0;}
      if (lookup≥24) {lookup=24; bias=0;}
      thisAlpha=(IFP)(cBoneAlpha[lookup]*(1-bias) + cBoneAlpha[lookup+1]*bias);
    }
    return thisAlpha;
    break;

  case 5: // In brain
    return (IFP)(alphaBrain*freq);
    break;

  // Other user-specified tissues
  case 6: //fat
    if (temperatureFlag=0)
      return (IFP)(alphaSeg6*freq);
    else
      //Ibrahim Hallaj's polynomial fit from Bamber/Hill, Damianou
      //From Hallaj's code: "the 0.3333 is to account for diff between atten and absorb"
      return (IFP)(freq*0.3333*( 1.2150e2
            -6.3162e0*temperature[index]
            +1.3516e-1*temperature[index]*temperature[index]
            -1.1639e-3*temperature[index]*temperature[index]*temperature[index]
            +3.6713e-6*temperature[index]*temperature[index]*temperature[index]*temperature[index])
);
    break;
```

```
  case 7: //liver
    if (temperatureFlag==0)
      return (IFP)(alphaSeg7*freq);
    else
      //Ibrahim Hallaj's polynomial fit from Bamber/Hill, Damianou
      return (IFP)(freq*(5.5367e0
            -2.9950e-1*temperature[index]
            +3.3357e-2*temperature[index]*temperature[index]
            -1.6058e-3*temperature[index]*temperature[index]*temperature[index]
            +3.4382e-5*temperature[index]*temperature[index]*temperature[index]*temperature[index]
            -3.2486e-7*temperature[index]*temperature[index]*temperature[index]*temperature[index]*
temperature[index]
            +1.1181e-9*temperature[index]*temperature[index]*temperature[index]*temperature[index]*
temperature[index]*temperature[index]));
    break;

  case 8: //buffer
    return (IFP)(alphaSeg8*freq);
    break;

  default:
    break;
  }

  // We should never actually reach here.
  return 0;
}
```

```
// Header file for Alpha3D

#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
```

```
// ArrayDefnHu3Dm.cpp
// Reads the file arrayFile that describes the shape and excitation
// of the sources to be used in this simulation.
// Also reads the file targetFile that earmarks points of special
// interest in the simulation volume for extra storage in the history
// array. This is necessary to do Fourier postprocessing on the pressure
// wave at these points.

#include "ArrayDefnHu3Dm.h"

int ArrayDefnHu3Dm(const char* arrayFile, const char* targetFile,
        const int globalBase, const int globalTop,
        int* iSource, int* jSource, int* kSource,
        float* ampSource, int* shiftSource, int* softSource,
        int *represents, int* Nsources,
        float* iTarget, float* jTarget, float* kTarget,
        int* Ntargets) {

  FILE* fileHandle;
  int   a, t;
  int   Ntruesources, Ntruetargets;

  if ((fileHandle=netOpen(arrayFile,"rb",_SH_DENYWR))≡NULL) return 1;

  for ( a=0 ; a<(*Nsources) ; a++ ) {
    if (¬(fread(iSource+a,sizeof(int),1,fileHandle)))   break;
    if (¬(fread(jSource+a,sizeof(int),1,fileHandle)))   break;
    if (¬(fread(kSource+a,sizeof(int),1,fileHandle)))   break;
    if (¬(fread(ampSource+a,sizeof(float),1,fileHandle))) break;
    if (¬(fread(shiftSource+a,sizeof(int),1,fileHandle))) break;
    if (¬(fread(softSource+a,sizeof(int),1,fileHandle)))  break;
    if (¬(fread(represents+a,sizeof(int),1,fileHandle)))  break;
  }

  fclose(fileHandle);

  if ((fileHandle=netOpen(targetFile,"rb",_SH_DENYWR))≡NULL) return 1;

  for ( t=0 ; t<(*Ntargets) ; t++ ) {
    if (¬(fread(iTarget+t,sizeof(float),1,fileHandle)))   break;
    if (¬(fread(jTarget+t,sizeof(float),1,fileHandle)))   break;
    if (¬(fread(kTarget+t,sizeof(float),1,fileHandle)))   break;
  }

  fclose(fileHandle);

  if ((a≠(*Nsources))∨(t≠(*Ntargets)))
    return 1; // Some data was missing somewhere.

  // Clear out the sources that are not in this simulation volume.
  Ntruesources=0;
  for (a=0 ; a<(*Nsources) ; a++) {
    if ((iSource[a]≥globalBase)∧(iSource[a]≤globalTop)) {
      iSource[Ntruesources]=iSource[a];
      jSource[Ntruesources]=jSource[a];
      kSource[Ntruesources]=kSource[a];
      ampSource[Ntruesources]=ampSource[a];
      shiftSource[Ntruesources]=shiftSource[a];
      softSource[Ntruesources]=softSource[a];
      represents[Ntruesources]=represents[a];
      Ntruesources++;
    }
  }

  // If we do have some array elements discarded due to being
  // outside the volume, we can't perform multiresolution bootup
  // because we can't be sure that necessary points haven't been
  // thrown out.
  if (Ntruesources<(*Nsources)) {
    for (a=0; a<Ntruesources; a++) {
      represents[a]=1;
    }
  }
  (*Nsources)=Ntruesources;

  // Clear out the targets that are not in this simulation volume.
  Ntruetargets=0;
  for (t=0 ; t<(*Ntargets) ; t++) {
    if ((iTarget[t]≥globalBase)∧(iTarget[t]≤globalTop)) {
      iTarget[Ntruetargets]=iTarget[t];
      jTarget[Ntruetargets]=jTarget[t];
      kTarget[Ntruetargets]=kTarget[t];
      Ntruetargets++;
    }
```

```
  }
  (*Ntargets)=Ntruetargets;

  return 0;
}
```

```
// Header file for ArrayDefnHu3Dm

#include <stdio.h>
#include <iostream.h>
//#include <windows.h>
#include <share.h>
#include "InternalPrecision.h"
#include "ioServices.h"

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// Beta3D.cpp
// Calculates the coefficient of nonlinearity at a mesh point given the
// mesh segmentation data and (depending on the material) the density.

#include "Beta3D.h"

IFP Beta3D(int index, float *rho, unsigned char *seg, float freq) {

  /*
    Choose the algorithm for determining the coefficient of nonlinearity
    The material types are:
    0: Outside the original data set
    1: Water
    2: Outer cortical bone
    3: Trabecular bone
    4: Inner cortical bone
    5: Brain
    6: User-specified - often fat
    7: User-specified - often liver
    8: User-specified

    This routine may be extended later to accommodate more materials.
  */

  switch (seg[index]) {
  case 0: // Outside the data set
    return (IFP)betaExtra;
    break;

  case 1: // In water
    return (IFP)betaH2O;
    break;

  case 2: // In bone (combined bone model)
  case 3:
  case 4:
    /*
    ---- Calculations/Models for Bone ------------------------------
    We are currently lacking a suitable model for the coefficient
    of nonlinearity in bone.
    */
    if (rho[index]≥1.6)
      // In cortical bone.
      return 0; // This still needs to be defined.
    else
      // In trabecular bone.
      return 0; // This still needs to be defined.
    break;

  case 5: // In brain
    return (IFP)betaBrain;
    break;

  // Other user-specified tissues
  case 6:
    return (IFP)betaSeg6;
    break;

  case 7:
    return (IFP)betaSeg7;
    break;

  case 8:
    return (IFP)betaSeg8;
    break;

  default:
    break;
  }

  // We should never actually reach here.
  return 0;
}
```

```
// Header file for Beta3D

#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
```

```
// BootFDTD3Dm.cpp
// Initialises an FDTD simulation using analytical linear methods.
// Designed to allow analytical methods to be used in water,
// switching to FDTD to cross the skull.
// Based on the Huygens-Fresnel integral equation.

#include "BootFDTD3Dm.h"

int BootFDTD3Dm( int *iSource, int *jSource, int *kSource,
         float *ampSource, int *shiftSource, int *softSource, int *represents,
         int Nsources, int initFrom, float lambda, int nptspercycle,
         IFP *pn, IFP *pnminus1, IFP *pnminus2,
         IFP *pnminus3, IFP *pnminus4, IFP *pnminus5,
         float dz, float dr, float dt, float trainmult, int Ntrain,
         int Imax, int Jmax, int Kmax, int globalBase,
         int gearRadius, int gearRatio, float *focus) {

  int   i,j,k,l,index;
  double  r1,rf,d,thisP,thisPhi,maxDist[6],dth,timePhi;
  double  thisCos,obliquity,radiansPerStep,radiansPerShift,stepOffset;

  double  *r0,*df;

  Disp("Performing Huygens-Fresnel boot-up.");

  // Allocate the memory for the r1 and df arrays.
  r0=(double*)malloc((unsigned)Nsources*sizeof(double));
  df=(double*)malloc((unsigned)Nsources*sizeof(double));
  if ((r0≡NULL)∨(df≡NULL)) {
    free(r0); free(df);
    return 1;
  }

  dth=2*pi/Kmax;
  radiansPerShift=2*pi/Ntrain;
  radiansPerStep=2*pi/nptspercycle;
  rf=dr*(focus[1]-0.5);
  if (rf<0) rf=0; // focusJ values less than 0.5 are meaningless.

  // Precalculate the r0 and df properties for each array element
  for (l=0; l<Nsources; l++) {
    r0[l]=dr*(jSource[l]-0.5);
    df[l]=sqrt(r0[l]*r0[l]+rf*rf-2*r0[l]*rf*cos(dth*(focus[2]-kSource[l]))+(dz*dz*(iSource[l]-focus[
0])*(iSource[l]-focus[0])));
  }

  for (i=0; i≤5; i++) {
    maxDist[i]=cH2O*dt*(initFrom-i);
  }

  obliquity=1;
  for (k=1; k≤Kmax; k++) {
    DispProgress(k,Kmax,0);
    for (j=1; j≤Jmax; j++) {
      //Determine whether gearing will be factor here.
      if ((gearRatio>0)∧(j≤gearRadius)) stepOffset=radiansPerStep/gearRatio;
      else stepOffset=radiansPerStep;
      for (i=1; i≤Imax; i++) {
        index=(i-1)+(j-1)*jInc+(k-1)*kInc;
        for (l=0; l<Nsources; l++) {
          if (ampSource[l]≠0) {
            thisCos=cos(dth*(k-kSource[l]));
            r1=dr*(j-0.5);
            d=sqrt(r0[l]*r0[l]+r1*r1-2*r0[l]*r1*thisCos+(dz*dz*(iSource[l]-globalBase+1-i)*(iSource[
l]-globalBase+1-i)));
            if (d≡0) break;
            if (d<maxDist[0]) {
              timePhi=radiansPerStep*initFrom+radiansPerShift*shiftSource[l];
              obliquity=(r0[l]*r0[l]-r0[l]*r1*thisCos-r0[l]*rf*cos(dth*(focus[2]-kSource[l]))+r1*rf*
cos(dth*(focus[2]-k))+dz*dz*(i-iSource[l]-globalBase+1)*(focus[0]-iSource[l]))/(d*df[l]);
              if (obliquity<0) obliquity=-obliquity;
              thisP=trainmult*ampSource[l]*r0[l]*dth*dr*obliquity*exp(-alphaH2O*d)/(lambda*d);
              thisPhi=2*pi*d/lambda;

              // Do multiresolutioning now
              if ((represents[l]>1)∧(d>(10*lambda))) {
                thisP*=represents[l];
                l+=(represents[l]-1);
              }

              pn[index]+=(IFP)(thisP*sin(timePhi-thisPhi));
              if (d<maxDist[1]) pnminus1[index]+=(IFP)(thisP*sin(timePhi-stepOffset-thisPhi));
              if (d<maxDist[2]) pnminus2[index]+=(IFP)(thisP*sin(timePhi-2*stepOffset-thisPhi));
              if (d<maxDist[3]) pnminus3[index]+=(IFP)(thisP*sin(timePhi-3*stepOffset-thisPhi));
```

```
            #if EXTENDED_ALPHA≠0
              if (d<maxDist[4]) pnminus4[index]+=(IFP)(thisP*sin(timePhi-4*stepOffset-thisPhi));

              if (d<maxDist[5]) pnminus5[index]+=(IFP)(thisP*sin(timePhi-5*stepOffset-thisPhi));
            #endif
          }
        }
      }
    }
  }

  // Do the coincident points now.
  for (l=0; l<Nsources; l++) {
    index=x(iSource[l]-globalBase+1,jSource[l],kSource[l]);
    if ((gearRatio>0)∧(jSource[l]≤gearRadius)) stepOffset=radiansPerStep/gearRatio;
    else stepOffset=radiansPerStep;
    timePhi=radiansPerStep*initFrom+radiansPerShift*shiftSource[l];
    pn[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi));
    pnminus1[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi-stepOffset));
    pnminus2[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi-2*stepOffset));
    pnminus3[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi-3*stepOffset));
    #if EXTENDED_ALPHA ≠ 0
      pnminus4[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi-4*stepOffset);
      pnminus5[index]=(IFP)(trainmult*ampSource[l]*sin(timePhi-5*stepOffset);
    #endif
  }

  free(r0); free(df);

  return 0;
}
```

```
// Header file for BootFDTD3Dm

#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include "ioServices.h"
extern int x(int i, int j, int k);
```

```
// Defines the global common material and mesh properties.
#include "InternalPrecision.h"

#define iInc 1
#define pi 3.14159265358979
extern int  Nprocessors, jInc, kInc, jIncFFR, kIncFFR, volSize, ffrVolSize;
extern float rhoH2O,cH2O,alphaH2O,betaH2O,cBrain,alphaBrain,betaBrain,
      cExtra,alphaExtra,betaExtra,cSeg6,alphaSeg6,betaSeg6,
      cSeg7,alphaSeg7,betaSeg7,cSeg8,alphaSeg8,betaSeg8;
extern int  temperatureFlag;
extern char temperatureFile[256];
extern IFP  *temperature;
extern int  cBoneFunction,cBoneFunctionAlpha;
extern float cBone[25],cBoneAlpha[25];
extern int x(int i, int j, int k);
```

```
// fdtd.cpp
// This routine handles the parallelizing of the FDTD algorithm
// over several processors, by splitting up the work and then
// calling FDTD3D_nonlinear_threaded_geared_iso.cpp.
// It serves an equivalent purpose to the thread driver routine
// in the original F90 code.

#include "fdtd.h"

void fdtd(IFP *pn,IFP *pnplus1,IFP *pnminus1,
  IFP *pnminus2, IFP *pnminus3, IFP *pnminus4, IFP *pnminus5, unsigned char *seg,
  int absorbTop, int absorbBase, float dz, float dr, float dt, float dth,
  int Imax, int Jmax, int Kmax, float *rho, float freq, float omega,
  int region, int Jinner, int Jouter, int gearRatio, int minorStep,
  int forceSingleThread, float *isoZ, float *isoR) {

  int     spread,t,Kfirst,Klast;
  DWORD   dummyID;
  fdtdStruct* parameters=new fdtdStruct[Nprocessors];

  if ((Nprocessors=1)∨(forceSingleThread≠0)) {
    // Single processor, or constrained to one processor.
    // Load up the structure and do the calculations.
    parameters[0].pn=pn; parameters[0].pnplus1=pnplus1;
    parameters[0].pnminus1=pnminus1; parameters[0].pnminus2=pnminus2;
    parameters[0].pnminus3=pnminus3; parameters[0].pnminus4=pnminus4; parameters[0].pnminus5=pnminus
5;
    parameters[0].seg=seg;
    parameters[0].absorbBase=absorbBase; parameters[0].absorbTop=absorbTop;
    parameters[0].dr=dr; parameters[0].dz=dz; parameters[0].dt=dt;
    parameters[0].dth=dth; parameters[0].freq=freq; parameters[0].omega=omega;
    parameters[0].Imax=Imax; parameters[0].Jmax=Jmax; parameters[0].Kmax=Kmax;
    parameters[0].Kfirst=1; parameters[0].Klast=Kmax;
    parameters[0].rho=rho; parameters[0].Jinner=Jinner; parameters[0].Jouter=Jouter;
    parameters[0].region=region; parameters[0].gearRatio=gearRatio;
    parameters[0].minorStep=minorStep;
    parameters[0].isoZ=isoZ; parameters[0].isoR=isoR;
    FDTD4s2t3d_nonlinear_threaded_geared_iso((LPVOID)parameters);
  } else {
    // Declare an array to hold the new thread handles.
    HANDLE*   threadHandles=new HANDLE[Nprocessors-1];
    spread=(Kmax+Nprocessors-1)/Nprocessors;

    for (t=0 ; t<Nprocessors ; t++) {
      // Establish the work boundaries for this thread.
      Kfirst=1+t*spread;
      Klast=__min(Kmax,Kfirst+spread-1);

      // Create the parameter block for this thread.
      parameters[t].pn=pn; parameters[t].pnplus1=pnplus1;
      parameters[t].pnminus1=pnminus1; parameters[t].pnminus2=pnminus2;
      parameters[t].pnminus3=pnminus3; parameters[t].pnminus4=pnminus4; parameters[t].pnminus5=pnmin
us5;
      parameters[t].seg=seg;
      parameters[t].absorbBase=absorbBase; parameters[t].absorbTop=absorbTop;
      parameters[t].dr=dr; parameters[t].dz=dz; parameters[t].dt=dt;
      parameters[t].dth=dth; parameters[t].freq=freq; parameters[t].omega=omega;
      parameters[t].Imax=Imax; parameters[t].Jmax=Jmax; parameters[t].Kmax=Kmax;
      parameters[t].Kfirst=Kfirst; parameters[t].Klast=Klast;
      parameters[t].rho=rho; parameters[t].Jinner=Jinner; parameters[t].Jouter=Jouter;
      parameters[t].region=region; parameters[t].gearRatio=gearRatio;
      parameters[t].minorStep=minorStep;
      parameters[t].isoZ=isoZ; parameters[t].isoR=isoR;
    }

    // Begin the threads.
    for (t=0 ; t<Nprocessors-1 ; t++) {
      threadHandles[t]=CreateThread(NULL,0,
        FDTD4s2t3d_nonlinear_threaded_geared_iso,(LPVOID)&parameters[t],
        0,(LPDWORD)&dummyID);
    }
    FDTD4s2t3d_nonlinear_threaded_geared_iso((LPVOID)&parameters[Nprocessors-1]);

    // Wait for threads to complete.
    WaitForMultipleObjects(Nprocessors-1,threadHandles,TRUE,INFINITE);
    for (t=0 ; t<Nprocessors-1; t++)
      CloseHandle(threadHandles[t]);
    delete threadHandles;
  }

  delete parameters;
}
```

```
// Header file for fdtd

#include <windows.h>
#include <process.h>
#include <stdlib.h>

#include "CommonMaterialProperties.h"
#include "fdtdStruct.h"
#include "InternalPrecision.h"

DWORD WINAPI FDTD4s2t3d_nonlinear_threaded_geared_iso (LPVOID parameters);
```

```
// FDTD4s2t3d_nonlinear_threaded_geared_iso.cpp
// This routine performs the 4th order in space, 2nd order in time,
// three dimensional finite difference time-domain calculations.

// This version was created on December 9,2000.
// It is descended from my original F90 implementation.

#include "FDTD4s2t3d_nonlinear_threaded_geared_iso.h"

DWORD WINAPI FDTD4s2t3d_nonlinear_threaded_geared_iso(LPVOID parameters) {

  // Unwrap the structure into local variables.
  fdtdStruct *setup = (fdtdStruct*) parameters;
  IFP *pn = setup->pn; IFP *pnplus1 = setup->pnplus1;
  IFP *pnminus1 = setup->pnminus1; IFP *pnminus2 = setup->pnminus2;
  IFP *pnminus3 = setup->pnminus3; IFP *pnminus4 = setup->pnminus4; IFP *pnminus5 = setup->pnminus5;
  unsigned char *seg = setup->seg;
  int absorbTop = setup->absorbTop; int absorbBase = setup->absorbBase;
  float dz = setup->dz; float dr = setup->dr; float dt = setup->dt;
  float dth = setup->dth; float freq = setup->freq; float omega=setup->omega;
  int Imax = setup->Imax; int Jmax=setup->Jmax; int Kmax=setup->Kmax;
  int Kfirst=setup->Kfirst; int Klast=setup->Klast;
  float *rho=setup->rho; int Jinner=setup->Jinner; int Jouter=setup->Jouter;
  int region=setup->region; int gearRatio=setup->gearRatio;
  int minorStep=setup->minorStep;
  float *isoZ=setup->isoZ; float *isoR=setup->isoR;

  // Variable declarations.

  int   i,j,k,opk,index,Jlimit;
  IFP weight[4],c,cAdj,r,ti,pnjplus1,pnjplus2,
    z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,z13,z14,
    q1,q2,q3,q4,q5,q6r,q6z,q6th,q7,dt_geared;
  unsigned char lastSeg;

  // Calculate properties for geared and iso operation.

  region--;
  dt_geared=dt/gearRatio;
  ti=(IFP)minorStep/(IFP)gearRatio;
  weight[0]=ti*(ti+1)*(ti+2);
  weight[1]=-3*(ti-1)*(ti+1)*(ti+2);
  weight[2]=3*(ti-1)*ti*(ti+2);
  weight[3]=-(ti-1)*ti*(ti+1);

  // We do the special cases first, and then the inner cells.

  if (Jouter ≡ Jmax) {
    // Special cases at the radial boundary.

    // Superabsorbing BC at cylindrical surface in the +radial direction:
    // ABC at r=Rmax, j=Jmax:
    for (k=Kfirst ; k≤Klast ; k++ ) {
      for (i=1 ; i≤Imax ; i++ ) {
        index=x(i,Jmax,k);
        c=Speed3D(index-jInc,rho,seg,freq)/isoR[region];
        cAdj=Speed3D(index-jInc-jInc,rho,seg,freq)/isoR[region];
        pnplus1[index]=pn[index]+(c/cAdj)*(pn[index-jInc]-pnminus1[index-jInc])-
          (c*dt_geared/dr)*(pn[index]-pn[index-jInc]-pnminus1[index-jInc]+pnminus1[index-2*jInc]);
      }
    }

    // FDTD O(dr^2, dz^2) at j=Jmax-1, i.e. one point inside from the cylindrical boundary
    // Explicitly calculates the cases involving wrap-around in k.
    j=Jmax-1;
    r=(IFP)((j-0.5)*dr);

    lastSeg=255;
    for (k=Kfirst ; k≤Klast ; k++ ) {
      for (i=2 ; i≤Imax-1 ; i++ ) {
        index=x(i,j,k);

        // Pressure field terms.
        z1=12*(pn[index+jInc]-2*pn[index]+pn[index-jInc]);
        z2=6*(pn[index+jInc]-pn[index-jInc]);
        z3=12*(pn[index+iInc]-2*pn[index]+pn[index-iInc]);
        z4=-2*pn[index]+pnminus1[index];

        // Absorption term in pressure field.
        #if EXTENDED_ALPHA ≡ 0
          z5=3*pn[index]-4*pnminus1[index]+pnminus2[index];
        #else
          z5=6*pn[index]-23*pnminus1[index]+34*pnminus2[index]-24*pnminus3[index]+8*pnminus4[index]-
pnminus5[index];
```

```
      #endif

      // Density field terms.
      z8=z2;
      z9=6*(rho[index+jInc]-rho[index-jInc]);
      z10=6*(pn[index+iInc]-pn[index-iInc]);
      z11=6*(rho[index+iInc]-rho[index-iInc]);

      // Nonlinear terms.
      z6=2*pn[index]-5*pnminus1[index]+4*pnminus2[index]-pnminus3[index];
      z7=3*pn[index]-4*pnminus1[index]+pnminus2[index];

      // Correct for the seam in k
      if (k≡1) {
        z12=12*(pn[index+kInc]-2*pn[index]+pn[x(i,j,Kmax)]);
        z13=6*(pn[index+kInc]-pn[x(i,j,Kmax)]);
        z14=6*(rho[index+kInc]-rho[x(i,j,Kmax)]);
      } else if (k≡Kmax) {
        z12=12*(pn[x(i,j,1)]-2*pn[index]+pn[index-kInc]);
        z13=6*(pn[x(i,j,1)]-pn[index-kInc]);
        z14=6*(rho[x(i,j,1)]-rho[index-kInc]);
      } else {
        z12=12*(pn[index+kInc]-2*pn[index]+pn[index-kInc]);
        z13=6*(pn[index+kInc]-pn[index-kInc]);
        z14=6*(rho[index+kInc]-rho[index-kInc]);
      }

      // Material properties in pressure field.
      if ( (temperatureFlag≠0)∨
        ((seg[index]≥2)∧(seg[index]≤4))∨(seg[index]≠lastSeg) ) {

        // Most of the time we don't have to recalculate these properties.
        lastSeg=seg[index];
        c=Speed3D(index,rho,seg,freq);
        q1=c*c*dt_geared*dt_geared/(12*dr*dr*isoR[region]*isoR[region]);
        q2=c*c*dt_geared*dt_geared/(12*dr*isoR[region]*isoR[region]);
        q3=c*c*dt_geared*dt_geared/(12*dz*dz*isoZ[region]*isoZ[region]);

        #if EXTENDED_ALPHA ≡ 0
          q4=-Alpha3D(index,rho,seg,freq)*c*dt_geared/2;
        #else
          q4=Alpha3D(index,rho,seg,freq)*c/(2*omega*omega*dt_geared);
        #endif

        q5=2*Beta3D(index,rho,seg,freq)/(1000*rho[index]*c*c);
        q6r=-c*c*dt_geared*dt_geared/(144*rho[index]*dr*dr*isoR[region]*isoR[region]);
        q6z=-c*c*dt_geared*dt_geared/(144*rho[index]*dz*dz*isoZ[region]*isoZ[region]);
        q6th=-c*c*dt_geared*dt_geared/(144*rho[index]*dth*dth);
        q7=c*c*dt_geared*dt_geared/(12*dth*dth);
      }

      pnplus1[index]=(IFP)(-z4+q1*z1+q2*z2/r+
        q3*z3+q7*z12/(r*r)+
        q4*z5+
        q5*(pn[index]*z6+0.25*z7*z7)+
        q6r*z8*z9+
        q6z*z10*z11+
        q6th*z13*z14/(r*r));

    }
  }
} // End of radial boundary special cases.

// The absorbing boundary at z=0 (i=1)
if (absorbBase≡1) {
  for (k=Kfirst ; k≤Klast ; k++) {
    for (j=Jinner ; j≤Jouter ; j++) {
      index=x(1,j,k);
      c=Speed3D(index+iInc,rho,seg,freq)/isoZ[region];
      cAdj=Speed3D(index+iInc+iInc,rho,seg,freq)/isoZ[region];
      pnplus1[index]=pn[index]+(c/cAdj)*(pn[index+iInc]-pnminus1[index+iInc])+
        (c*dt_geared/dz)*(pn[index+iInc]-pn[index]-pnminus1[index+2*iInc]+pnminus1[index+iInc]);
    }
  }
}

// The absorbing boundary at z=Zmax (i=Imax)
if (absorbTop≡1) {
  for (k=Kfirst ; k≤Klast ; k++) {
    for (j=Jinner ; j≤Jouter ; j++) {
      index=x(Imax,j,k);
      c=Speed3D(index-iInc,rho,seg,freq)/isoZ[region];
      cAdj=Speed3D(index-iInc-iInc,rho,seg,freq)/isoZ[region];
      pnplus1[index]=pn[index]+(c/cAdj)*(pn[index-iInc]-pnminus1[index-iInc])-
```

```
            (c*dt_geared/dz)*(pn[index]-pn[index-iInc]-pnminus1[index-iInc]+pnminus1[index-2*iInc]);
        }
      }
    }

    // Second order space, second order time at i=2:
    // Also explicitly includes wrap-around in k, when calculations
    // in j cross the centre, and when calculations in j cross the gearRadius.
    // Only need to do this if the base is absorbent.

    if (absorbBase==1) {
      i=2;
      if (Jouter == Jmax)
        Jlimit=Jmax-1;
      else
        Jlimit=Jouter;

      lastSeg=255;
      for (k=Kfirst ; k<=Klast ; k++) {
        for (j=Jinner ; j<=Jlimit ; j++) {
          r=(IFP)((j-0.5)*dr);
          index=x(i,j,k);

          if (j==1) {
            opk=(k+Kmax/2-1)%Kmax + 1;
            z1=12*(pn[index+jInc]-2*pn[index]+pn[x(i,j,opk)]);
            z2=6*(pn[index+jInc]-pn[x(i,j,opk)]);
            z8=z2;
            z9=6*(rho[index+jInc]-rho[x(i,j,opk)]);
          } else if (j==Jouter) {
            pnjplus1=(weight[0]*pnplus1[index+jInc]+weight[1]*pn[index+jInc]+
              weight[2]*pnminus1[index+jInc]+weight[3]*pnminus2[index+jInc])/6;
            z1=12*(pnjplus1-2*pn[index]+pn[index-jInc]);
            z2=6*(pnjplus1-pn[index-jInc]);
            z8=z2;
            z9=6*(rho[index+jInc]-rho[index-jInc]);
          } else {
            z1=12*(pn[index+jInc]-2*pn[index]+pn[index-jInc]);
            z2=6*(pn[index+jInc]-pn[index-jInc]);
            z8=z2;
            z9=6*(rho[index+jInc]-rho[index-jInc]);
          }

          z3=12*(pn[index+iInc]-2*pn[index]+pn[index-iInc]);
          z4=-2*pn[index]+pnminus1[index];

          #if EXTENDED_ALPHA == 0
            z5=3*pn[index]-4*pnminus1[index]+pnminus2[index];
          #else
            z5=6*pn[index]-23*pnminus1[index]+34*pnminus2[index]-24*pnminus3[index]+8*pnminus4[index]-
pnminus5[index];
          #endif

          z6=2*pn[index]-5*pnminus1[index]+4*pnminus2[index]-pnminus3[index];
          z7=3*pn[index]-4*pnminus1[index]+pnminus2[index];
          z10=6*(pn[index+iInc]-pn[index-iInc]);
          z11=6*(rho[index+iInc]-rho[index-iInc]);

          // Correct for the seam in k
          if (k==1) {
            z12=12*(pn[index+kInc]-2*pn[index]+pn[x(i,j,Kmax)]);
            z13=6*(pn[index+kInc]-pn[x(i,j,Kmax)]);
            z14=6*(rho[index+kInc]-rho[x(i,j,Kmax)]);
          } else if (k==Kmax) {
            z12=12*(pn[x(i,j,1)]-2*pn[index]+pn[index-kInc]);
            z13=6*(pn[x(i,j,1)]-pn[index-kInc]);
            z14=6*(rho[x(i,j,1)]-rho[index-kInc]);
          } else {
            z12=12*(pn[index+kInc]-2*pn[index]+pn[index-kInc]);
            z13=6*(pn[index+kInc]-pn[index-kInc]);
            z14=6*(rho[index+kInc]-rho[index-kInc]);
          }

          // Calculate the material coefficients.
          if ( (temperatureFlag!=0)∨
            ((seg[index]≥2)∧(seg[index]≤4))∨(seg[index]≠lastSeg) ) {

            // Most of the time we don't have to recalculate these properties.
            lastSeg=seg[index];
            c=Speed3D(index,rho,seg,freq);
            q1=c*c*dt_geared*dt_geared/(12*dr*dr*isoR[region]*isoR[region]);
            q2=c*c*dt_geared*dt_geared/(12*dr*isoR[region]*isoR[region]);
            q3=c*c*dt_geared*dt_geared/(12*dz*dz*isoZ[region]*isoZ[region]);
```

```
          #if EXTENDED_ALPHA ≡ 0
            q4=-Alpha3D(index,rho,seg,freq)*c*dt_geared/2;
          #else
            q4=Alpha3D(index,rho,seg,freq)*c/(2*omega*omega*dt_geared);
          #endif

          q5=2*Beta3D(index,rho,seg,freq)/(1000*rho[index]*c*c);
          q6r=-c*c*dt_geared*dt_geared/(144*rho[index]*dr*dr*isoR[region]*isoR[region]);
          q6z=-c*c*dt_geared*dt_geared/(144*rho[index]*dz*dz*isoZ[region]*isoZ[region]);
          q6th=-c*c*dt_geared*dt_geared/(144*rho[index]*dth*dth);
          q7=c*c*dt_geared*dt_geared/(12*dth*dth);
        }

        pnplus1[index]=(IFP)(-z4+q1*z1+q2*z2/r+
          q3*z3+q7*z12/(r*r)+
          q4*z5+
          q5*(pn[index]*z6+0.25*z7*z7)+
          q6r*z8*z9+
          q6z*z10*z11+
          q6th*z13*z14/(r*r));
      }
    }
  }

  // Second order space, second order time at i=Imax-1:
  // Also explicitly includes wrap-around in k, when calculations
  // in j cross the centre, and when calculations in j cross the gearRadius.
  // Only need to do this if the top is absorbent.

  if (absorbTop≡1) {
    i=Imax-1;
    if (Jouter ≡ Jmax)
      Jlimit=Jmax-1;
    else
      Jlimit=Jouter;

    lastSeg=255;
    for (k=Kfirst ; k≤Klast ; k++) {
      for (j=Jinner ; j≤Jlimit ; j++) {
        r=(IFP)((j-0.5)*dr);
        index=x(i,j,k);

        if (j≡1) {
          opk=(k+Kmax/2-1)%Kmax + 1;
          z1=12*(pn[index+jInc]-2*pn[index]+pn[x(i,j,opk)]);
          z2=6*(pn[index+jInc]-pn[x(i,j,opk)]);
          z8=z2;
          z9=6*(rho[index+jInc]-rho[x(i,j,opk)]);
        } else if (j≡Jouter) {
          pnjplus1=(weight[0]*pnplus1[index+jInc]+weight[1]*pn[index+jInc]+
            weight[2]*pnminus1[index+jInc]+weight[3]*pnminus2[index+jInc])/6;
          z1=12*(pnjplus1-2*pn[index]+pn[index-jInc]);
          z2=6*(pnjplus1-pn[index-jInc]);
          z8=z2;
          z9=6*(rho[index+jInc]-rho[index-jInc]);
        } else {
          z1=12*(pn[index+jInc]-2*pn[index]+pn[index-jInc]);
          z2=6*(pn[index+jInc]-pn[index-jInc]);
          z8=z2;
          z9=6*(rho[index+jInc]-rho[index-jInc]);
        }

        z3=12*(pn[index+iInc]-2*pn[index]+pn[index-iInc]);
        z4=-2*pn[index]+pnminus1[index];

        #if EXTENDED_ALPHA ≡ 0
          z5=3*pn[index]-4*pnminus1[index]+pnminus2[index];
        #else
          z5=6*pn[index]-23*pnminus1[index]+34*pnminus2[index]-24*pnminus3[index]+8*pnminus4[index]-
pnminus5[index];
        #endif

        z6=2*pn[index]-5*pnminus1[index]+4*pnminus2[index]-pnminus3[index];
        z7=3*pn[index]-4*pnminus1[index]+pnminus2[index];
        z10=6*(pn[index+iInc]-pn[index-iInc]);
        z11=6*(rho[index+iInc]-rho[index-iInc]);

        // Correct for the seam in k
        if (k≡1) {
          z12=12*(pn[index+kInc]-2*pn[index]+pn[x(i,j,Kmax)]);
          z13=6*(pn[index+kInc]-pn[x(i,j,Kmax)]);
          z14=6*(rho[index+kInc]-rho[x(i,j,Kmax)]);
        } else if (k≡Kmax) {
          z12=12*(pn[x(i,j,1)]-2*pn[index]+pn[index-kInc]);
```

```
        z13=6*(pn[x(i,j,1)]-pn[index-kInc]);
        z14=6*(rho[x(i,j,1)]-rho[index-kInc]);
      } else {
        z12=12*(pn[index+kInc]-2*pn[index]+pn[index-kInc]);
        z13=6*(pn[index+kInc]-pn[index-kInc]);
        z14=6*(rho[index+kInc]-rho[index-kInc]);
      }

      // Calculate the material coefficients.
      if ( (temperatureFlag≠0)∨
        ((seg[index]≥2)∧(seg[index]≤4))∨(seg[index]≠lastSeg) ) {

        // Most of the time we don't have to recalculate these properties.
        lastSeg=seg[index];
        c=Speed3D(index,rho,seg,freq);
        q1=c*c*dt_geared*dt_geared/(12*dr*dr*isoR[region]*isoR[region]);
        q2=c*c*dt_geared*dt_geared/(12*dr*isoR[region]*isoR[region]);
        q3=c*c*dt_geared*dt_geared/(12*dz*dz*isoZ[region]*isoZ[region]);

        #if EXTENDED_ALPHA ≡ 0
          q4=-Alpha3D(index,rho,seg,freq)*c*dt_geared/2;
        #else
          q4=Alpha3D(index,rho,seg,freq)*c/(2*omega*omega*dt_geared);
        #endif

        q5=2*Beta3D(index,rho,seg,freq)/(1000*rho[index]*c*c);
        q6r=-c*c*dt_geared*dt_geared/(144*rho[index]*dr*dr*isoR[region]*isoR[region]);
        q6z=-c*c*dt_geared*dt_geared/(144*rho[index]*dz*dz*isoZ[region]*isoZ[region]);
        q6th=-c*c*dt_geared*dt_geared/(144*rho[index]*dth*dth);
        q7=c*c*dt_geared*dt_geared/(12*dth*dth);
      }

      pnplus1[index]=(IFP)(-z4+q1*z1+q2*z2/r+
        q3*z3+q7*z12/(r*r)+
        q4*z5+
        q5*(pn[index]*z6+0.25*z7*z7)+
        q6r*z8*z9+
        q6z*z10*z11+
        q6th*z13*z14/(r*r));
    }
  }
}

// Corner elements: these are calculated by taking the average of their
// neighbours, since there is insufficient information to do anything else.
// This is only relevant if the point is against an absorbing border.

if (absorbBase≡1) {
  for (k=Kfirst ; k≤Klast ; k++) {
    if (Jouter ≡ Jmax) {
      pnplus1[x(1,Jmax,k)]=(IFP)(0.5*(pnplus1[x(1,Jmax-1,k)]+pnplus1[x(2,Jmax,k)]));
    }
  }
}

if (absorbTop≡1) {
  for (k=Kfirst ; k≤Klast ; k++) {
    if (Jouter ≡ Jmax) {
      pnplus1[x(Imax,Jmax,k)]=(IFP)(0.5*(pnplus1[x(Imax-1,Jmax,k)]+pnplus1[x(Imax,Jmax-1,k)]));
    }
  }
}

// The inner cells. These are calculated to order 4 in space, order 2 in time.
// Note the provisions for the seam in k, calculations in j that cross the centre,
// and calculations in j that cross the gearRadius.

if (Jouter≡Jmax)
  Jlimit=Jmax-2;
else
  Jlimit=Jouter;

lastSeg=255;
for ( k=Kfirst ; k≤Klast ; k++ ) {
  for ( j=Jinner ; j≤Jlimit ; j++ ) {
    for ( i=3 ; i≤Imax-2 ; i++ ) {
      index=x(i,j,k);
      r=(IFP)((j-0.5)*dr);

      // Calculate the spatial functions.
      if (j≡1) {
        opk=(k+Kmax/2-1)%Kmax + 1;
        z1=-pn[index+2*jInc]+16*pn[index+jInc]-30*pn[index]+16*pn[x(i,j,opk)]-pn[x(i,j+1,opk)];
        z2=-pn[index+2*jInc]+8*pn[index+jInc]-8*pn[x(i,j,opk)]+pn[x(i,j+1,opk)];
```

```
        z8=z2;
        z9=-rho[index+2*jInc]+8*rho[index+jInc]-8*rho[x(i,j,opk)]+rho[x(i,j+1,opk)];
      } else if (j≡2) {
        opk=(k+Kmax/2-1)%Kmax + 1;
        z1=-pn[index+2*jInc]+16*pn[index+jInc]-30*pn[index]+16*pn[index-jInc]-pn[x(i,j-1,opk)];
        z2=-pn[index+2*jInc]+8*pn[index+jInc]-8*pn[index-jInc]+pn[x(i,j-1,opk)];
        z8=z2;
        z9=-rho[index+2*jInc]+8*rho[index+jInc]-8*rho[index-jInc]+rho[x(i,j-1,opk)];
      } else if (j≡Jouter) {
        pnjplus2=(weight[0]*pnplus1[index+2*jInc]+weight[1]*pn[index+2*jInc]+
          weight[2]*pnminus1[index+2*jInc]+weight[3]*pnminus2[index+2*jInc])/6;
        pnjplus1=(weight[0]*pnplus1[index+jInc]+weight[1]*pn[index+jInc]+
          weight[2]*pnminus1[index+jInc]+weight[3]*pnminus2[index+jInc])/6;
        z1=-pnjplus2+16*pnjplus1-30*pn[index]+16*pn[index-jInc]-pn[index-2*jInc];
        z2=-pnjplus2+8*pnjplus1-8*pn[index-jInc]+pn[index-2*jInc];
        z8=z2;
        z9=-rho[index+2*jInc]+8*rho[index+jInc]-8*rho[index-jInc]+rho[index-2*jInc];
      } else if (j≡Jouter-1) {
        pnjplus2=(weight[0]*pnplus1[index+2*jInc]+weight[1]*pn[index+2*jInc]+
          weight[2]*pnminus1[index+2*jInc]+weight[3]*pnminus2[index+2*jInc])/6;
        z1=-pnjplus2+16*pn[index+jInc]-30*pn[index]+16*pn[index-jInc]-pn[index-2*jInc];
        z2=-pnjplus2+8*pn[index+jInc]-8*pn[index-jInc]+pn[index-2*jInc];
        z8=z2;
        z9=-rho[index+2*jInc]+8*rho[index+jInc]-8*rho[index-jInc]+rho[index-2*jInc];
      } else {
        z1=-pn[index+2*jInc]+16*pn[index+jInc]-30*pn[index]+16*pn[index-jInc]-pn[index-2*jInc];
        z2=-pn[index+2*jInc]+8*pn[index+jInc]-8*pn[index-jInc]+pn[index-2*jInc];
        z8=z2;
        z9=-rho[index+2*jInc]+8*rho[index+jInc]-8*rho[index-jInc]+rho[index-2*jInc];
      }

      z3=-pn[index+2*iInc]+16*pn[index+iInc]-30*pn[index]+16*pn[index-iInc]-pn[index-2*iInc];
      z4=-2*pn[index]+pnminus1[index];

      #if EXTENDED_ALPHA ≡ 0
        z5=3*pn[index]-4*pnminus1[index]+pnminus2[index];
      #else
        z5=6*pn[index]-23*pnminus1[index]+34*pnminus2[index]-24*pnminus3[index]+8*pnminus4[index]-
pnminus5[index];
      #endif

      z6=2*pn[index]-5*pnminus1[index]+4*pnminus2[index]-pnminus3[index];
      z7=3*pn[index]-4*pnminus1[index]+pnminus2[index];
      z10=-pn[index+2*iInc]+8*pn[index+iInc]-8*pn[index-iInc]+pn[index-2*iInc];
      z11=-rho[index+2*iInc]+8*rho[index+iInc]-8*rho[index-iInc]+rho[index-2*iInc];

      // Solve the problem of the seam in k here
      if (k≡1) {
        z12=-pn[index+2*kInc]+16*pn[index+kInc]-30*pn[index]+16*pn[x(i,j,Kmax)]-pn[x(i,j,Kmax-1)];
        z13=-pn[index+2*kInc]+8*pn[index+kInc]-8*pn[x(i,j,Kmax)]+pn[x(i,j,Kmax-1)];
        z14=-rho[index+2*kInc]+8*rho[index+kInc]-8*rho[x(i,j,Kmax)]+rho[x(i,j,Kmax-1)];
      } else if (k≡2) {
        z12=-pn[index+2*kInc]+16*pn[index+kInc]-30*pn[index]+16*pn[index-kInc]-pn[x(i,j,Kmax)];
        z13=-pn[index+2*kInc]+8*pn[index+kInc]-8*pn[index-kInc]+pn[x(i,j,Kmax)];
        z14=-rho[index+2*kInc]+8*rho[index+kInc]-8*rho[index-kInc]+rho[x(i,j,Kmax)];
      } else if (k≡Kmax-1) {
        z12=-pn[x(i,j,1)]+16*pn[index+kInc]-30*pn[index]+16*pn[index-kInc]-pn[index-2*kInc];
        z13=-pn[x(i,j,1)]+8*pn[index+kInc]-8*pn[index-kInc]+pn[index-2*kInc];
        z14=-rho[x(i,j,1)]+8*rho[index+kInc]-8*rho[index-kInc]+rho[index-2*kInc];
      } else if (k≡Kmax) {
        z12=-pn[x(i,j,2)]+16*pn[x(i,j,1)]-30*pn[index]+16*pn[index-kInc]-pn[index-2*kInc];
        z13=-pn[x(i,j,2)]+8*pn[x(i,j,1)]-8*pn[index-kInc]+pn[index-2*kInc];
        z14=-rho[x(i,j,2)]+8*rho[x(i,j,1)]-8*rho[index-kInc]+rho[index-2*kInc];
      } else {
        z12=-pn[index+2*kInc]+16*pn[index+kInc]-30*pn[index]+16*pn[index-kInc]-pn[index-2*kInc];
        z13=-pn[index+2*kInc]+8*pn[index+kInc]-8*pn[index-kInc]+pn[index-2*kInc];
        z14=-rho[index+2*kInc]+8*rho[index+kInc]-8*rho[index-kInc]+rho[index-2*kInc];
      }

      // Calculate the material coefficients.
      if ( (temperatureFlag≠0)∨
        ((seg[index]≥2)∧(seg[index]≤4))∨(seg[index]≠lastSeg) ) {

        // Most of the time we don't have to recalculate these properties.
        lastSeg=seg[index];
        c=Speed3D(index,rho,seg,freq);
        q1=c*c*dt_geared*dt_geared/(12*dr*dr*isoR[region]*isoR[region]);
        q2=c*c*dt_geared*dt_geared/(12*dr*isoR[region]*isoR[region]);
        q3=c*c*dt_geared*dt_geared/(12*dz*dz*isoZ[region]*isoZ[region]);

        #if EXTENDED_ALPHA ≡ 0
          q4=-Alpha3D(index,rho,seg,freq)*c*dt_geared/2;
        #else
          q4=Alpha3D(index,rho,seg,freq)*c/(2*omega*omega*dt_geared);
```

```
        #endif

        q5=2*Beta3D(index,rho,seg,freq)/(1000*rho[index]*c*c);
        q6r=-c*c*dt_geared*dt_geared/(144*rho[index]*dr*dr*isoR[region]*isoR[region]);
        q6z=-c*c*dt_geared*dt_geared/(144*rho[index]*dz*dz*isoZ[region]*isoZ[region]);
        q6th=-c*c*dt_geared*dt_geared/(144*rho[index]*dth*dth);
        q7=c*c*dt_geared*dt_geared/(12*dth*dth);
      }

      pnplus1[index]=(IFP)(-z4+q1*z1+q2*z2/r+
        q3*z3+q7*z12/(r*r)+
        q4*z5+
        q5*(pn[index]*z6+0.25*z7*z7)+
        q6r*z8*z9+
        q6z*z10*z11+
        q6th*z13*z14/(r*r));
    }
  }
}

return 0;
}
```

```
// Header file for FDTD4s2t3d_nonlinear_threaded_geared_iso
#include <windows.h>
#include "InternalPrecision.h"

extern IFP Alpha3D(int index, float *rho, unsigned char *seg, float freq);
extern IFP Speed3D(int index, float *rho, unsigned char *seg, float freq);
extern IFP Beta3D(int index, float *rho, unsigned char *seg, float freq);

#include "CommonMaterialProperties.h"
#include "fdtdStruct.h"
```

```
// Defines the structure for passing information
// to the FDTD worker threads.

#include "InternalPrecision.h"
typedef struct tag_fdtdStruct {
  IFP *pn,*pnplus1,*pnminus1,*pnminus2,*pnminus3,*pnminus4,*pnminus5;
  unsigned char *seg;
  float *rho;
  int absorbTop,absorbBase;
  float dz,dr,dt,dth,freq,omega;
  int Imax,Jmax,Kmax,Kfirst,Klast,Jinner,Jouter;
  int region,gearRatio,minorStep;
  float *isoZ,*isoR;
} fdtdStruct;
```

```
// FFRfilldown.cpp
// Completes an FDTD simulation using analytical linear methods.
// Designed to allow analytical methods to be used in brain,
// after FDTD methods have been used to cross the skull.
// Probably useful for other things as yet undetermined.
// Based on the Huygens-Fresnel equation.

#include "FFRfilldown.h"

int FFRfilldown(IFP *ffrA, IFP *ffrPhi, int Ifilldown, float lambdaID, float alphaID,
        int Ntargets, float *iTarget, float *jTarget, float *kTarget, float *ampTarget, float *phiT
arget,
        int Imax, int Jmax, int Kmax, int ffrExtent, float dr, float dz, int globalBase, int ffrFla
g) {

  int   i,j,k,l,jSource,kSource,index,indexSource;
  double  r0,r1,d,thisP,thisPhi,dth,obliquity;
  double  realPart, imagPart;

  dth=2*pi/Kmax;

  if (ffrFlag>0) {
    Disp("Performing Fourier filldown.");
    Ifilldown=Ifilldown-globalBase+1; // Do this upfront.
    for (k=1; k≤Kmax; k++) {
      DispProgress(k,Kmax,0);
      for (j=1; j≤Jmax; j++) {
        for (i=(Ifilldown+1); i≤ffrExtent; i++) {
          index=((k-1)*Jmax+(j-1))*ffrExtent+(i-1);
          realPart=0; imagPart=0;
          for (kSource=1; kSource≤Kmax; kSource++) {
            for (jSource=1; jSource≤Jmax; jSource++) {
              indexSource=(Ifilldown-1)+(jSource-1)*jIncFFR+(kSource-1)*kIncFFR;
              r0=dr*(jSource-0.5);
              r1=dr*(j-0.5);
              d=sqrt(r0*r0+r1*r1-2*r0*r1*cos(dth*(k-kSource))+(dz*dz*(i-Ifilldown)*(i-Ifilldown)));
              obliquity=(dz*(i-Ifilldown))/d;
              thisP=ffrA[indexSource]*r0*dth*dr*obliquity*exp(-alphaID*d)/(lambdaID*d);
              thisPhi=ffrPhi[indexSource]-(2*pi*d/lambdaID);
              realPart+=thisP*cos(thisPhi);
              imagPart+=thisP*sin(thisPhi);
            }
          }
          // Convert cartesian complex back to polar complex.
          ffrA[index]=(IFP)sqrt(realPart*realPart+imagPart*imagPart);
          ffrPhi[index]=(IFP)atan2(imagPart,realPart);
        }
      }
    }
    Ifilldown=Ifilldown+globalBase-1; // Undo this.
  }

  if (Ntargets>0) {
    // Produce filldown to the specified target points.
    Disp("Performing Fourier filldown to targets.");
    for (l=0; l<Ntargets; l++) {
      DispProgress(l+1,Ntargets,0);
      realPart=0; imagPart=0;
      for (kSource=1; kSource≤Kmax; kSource++) {
        for (jSource=1; jSource≤Jmax; jSource++) {
          indexSource=(Ifilldown-globalBase)+(jSource-1)*jIncFFR+(kSource-1)*kIncFFR;
          r0=dr*(jSource-0.5);
          r1=dr*(jTarget[l]-0.5);
          d=sqrt(r0*r0+r1*r1-2*r0*r1*cos(dth*(kTarget[l]-kSource))+(dz*dz*(iTarget[l]-Ifilldown)*(iT
arget[l]-Ifilldown)));
          obliquity=(dz*(iTarget[l]-Ifilldown))/d;
          thisP=ffrA[indexSource]*r0*dth*dr*obliquity*exp(-alphaID*d)/(lambdaID*d);
          thisPhi=ffrPhi[indexSource]-(2*pi*d/lambdaID);
          realPart+=thisP*cos(thisPhi);
          imagPart+=thisP*sin(thisPhi);
        }
      }
      // Convert cartesian complex back to polar complex.
      ampTarget[l]=(IFP)sqrt(realPart*realPart+imagPart*imagPart);
      phiTarget[l]=(IFP)atan2(imagPart,realPart);
    }
  }
  return 0; // No failure case for this function
}
```

```
// Header file for FFRfilldown

#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include "ioServices.h"
```

```
// This is the Rainfall Process function.
// It implements the HuFDTD3D Rainfall method.

// HuFDTD3D - The 3D Trans-Cranial Acoustic Propagation Solver
// Begun by Chris Connor on Apr 30, 2002.
// Solves for acoustic propagation in 3D using a combination of
// the Huygens-Fresnel integral equation to handle regions of simple
// linear acoustic propagation and employing a 3D Finite Difference
// Time Domain method to handle propagation across the skull.

#define DIAGNOSTIC
#include "HuFDTD3D.H"

// Declare global properties of materials and of the mesh.
// GLOBAL VARIABLES
float rhoH2O,cH2O,alphaH2O,betaH2O,cBrain,alphaBrain,betaBrain,
    cExtra,alphaExtra,betaExtra,cSeg6,alphaSeg6,betaSeg6,
    cSeg7,alphaSeg7,betaSeg7,cSeg8,alphaSeg8,betaSeg8;
int   Nprocessors, jInc, kInc, jIncFFR, kIncFFR, volSize, ffrVolSize, temperatureFlag;
IFP   *temperature;
char  temperatureFile[256];
int   cBoneFunction,cBoneFunctionAlpha;
float cBone[25],cBoneAlpha[25];
// END OF GLOBAL VARIABLES

extern "C" int RainfallProcess(char *inFile) {

  // Variable definitions (non-array)
  int   Imax,Jmax,Kmax,Ntrain,Nsources,Nmax,Ntargets,i,j,k,n,minorStep,
      nptspercycle,decimFlag,decimSpace,decimTime,nendtoend,index,indexFFR,
      absorbTop,absorbBase,planeFlag,termPulse,ffrExtent,
      globalTop,globalBase,forceSingleThread,gearRadius,gearRatio,
      initFrom,restoreFlag,QFlag,peakFlag,failFlag,ffrFlag,waveTrap,bootStep;

  char  trainFile[256],historyFile[256],initFile[256],restoreFile[256],
      arrayFile[256],targetFile[256],peakFile[256],ffrFile[256],
      segFile[256],rhoFile[256],decimProto[256],QFile[256],
      planeProto[256],netPath[256],seqIdent[256],buffer[256];

  float dz,dr,dt,dth,zshock,zmax,rmax,tmax,period,freq,
      omega,kwave,lambda,tendtoend,Umax,Mach,Gold,fourierP,fourierQ,
      trainmult,CFL,isoZ[2],isoR[2],focus[3],thisAmp,thisPhi;

  FILE  *fileHandle;
  gzFile  gzFileHandle;

  // Declare pointers to dynamic arrays.
  // IFP is the internal floating point precision, which is either defined
  // as double or float in InternalPrecision.h
  // pnminus4 and pnminus5 are only allocated if EXTENDED_ALPHA is set.
  int       *iSource,*jSource,*kSource,*shiftSource,*softSource,*represents;
  float       *iTarget,*jTarget,*kTarget,*ampTarget,*phiTarget;
  float     *fourierA,*fourierB;
  unsigned char *seg;
  IFP       *train;
  float     *rho,*ampSource;
  IFP       *pnplus1,*pn,*pnminus1,*pnminus2,*pnminus3,*ppeak,*Q;
  IFP       *pnminus4, *pnminus5;

  // Obtain number of processors.
  // Available in Windows NT/2000 in an environment variable.
  Nprocessors = atoi(getenv("NUMBER_OF_PROCESSORS"));
  sprintf(buffer,"Number of processors: %i",Nprocessors);
  Disp(buffer);

  // Declare the internal precision of this solver.
  // Only solvers of the same internal precision may collaborate.
  if (sizeof(IFP)==sizeof(double))
    Disp("Using double precision internal arithmetic.");
  else
    Disp("Using single precision internal arithmetic.");

  // Name the job we're doing.
  sprintf(buffer,"Beginning job %s",inFile);
  Disp(buffer);

  // Read in the configuration file for this job.
  if (ReadConfig3D(inFile, &Imax, &Jmax, &Kmax, &bootStep, &Nmax,
      &dr, &dz, &dt,
      &Nsources, &arrayFile[0],
      &waveTrap, &Ntargets, &targetFile[0],
      &focus[0],
      &historyFile[0],
      &peakFlag,&peakFile[0],
```

```
      &ffrFlag,&ffrExtent,&ffrFile[0],
      &segFile[0],&rhoFile[0],
      &absorbBase, &absorbTop,
      &globalBase, &globalTop,
      &netPath[0], &seqIdent[0],
      &decimFlag, &decimSpace, &decimTime,
      &decimProto[0],
      &planeFlag, &planeProto[0],
      &Ntrain, &trainFile[0],
      &QFlag,&QFile[0],
      &nptspercycle, &trainmult, &termPulse,
      &forceSingleThread, &gearRadius, &gearRatio,
      &isoZ[0], &isoR[0],
      &initFrom, &initFile[0],
      &restoreFlag, &restoreFile[0])) {
    Disp("Couldn't obtain the configuration file.");
    Disp("Non-existent file or unreachable network path.");
    return 3;
  }

  #ifdef DIAGNOSTIC
    sprintf(buffer,"Imax, Jmax, Kmax, bootStep, Nmax: %i,%i,%i,%i,%i",Imax,Jmax,Kmax,bootStep,Nmax); Disp(buffe
r);
    sprintf(buffer,"dr, dz, dt: %g,%g,%g",dr,dz,dt); Disp(buffer);
    sprintf(buffer,"Nsources, arrayFile: %i, %s",Nsources,arrayFile); Disp(buffer);
    sprintf(buffer,"waveTrap, Ntargets, targetFile: %i,%i, %s",waveTrap,Ntargets,targetFile); Disp(buffer);
    sprintf(buffer,"historyFile: %s",historyFile); Disp(buffer);
    sprintf(buffer,"peakFlag, peakFile: %i, %s",peakFlag,peakFile); Disp(buffer);
    sprintf(buffer,"ffrFlag, ffrExtent, ffrFile: %i, %i, %s",ffrFlag,ffrExtent,ffrFile); Disp(buffer);
    sprintf(buffer,"segFile, rhoFile: %s, %s",segFile,rhoFile); Disp(buffer);
    sprintf(buffer,"absorbBase, absorbTop: %i, %i",absorbBase,absorbTop); Disp(buffer);
    sprintf(buffer,"globalBase, globalTop: %i, %i",globalBase,globalTop); Disp(buffer);
    sprintf(buffer,"netPath, seqIdent: %s, %s",netPath,seqIdent); Disp(buffer);
    sprintf(buffer,"decimFlag, decimSpace, decimTime: %i, %i, %i",
      decimFlag,decimSpace,decimTime); Disp(buffer);
    sprintf(buffer,"decimProto: %s",decimProto); Disp(buffer);
    sprintf(buffer,"planeFlag, planeProto: %i, %s",planeFlag,planeProto); Disp(buffer);
    sprintf(buffer,"cH2O, alphaH2O, betaH2O, rhoH2O: %f,%f,%f,%f",
      cH2O,alphaH2O,betaH2O,rhoH2O); Disp(buffer);
    sprintf(buffer,"cBrain, alphaBrain, betaBrain: %f,%f,%f",
      cBrain,alphaBrain,betaBrain); Disp(buffer);
    sprintf(buffer,"cSeg6, alphaSeg6, betaSeg6: %f,%f,%f",
      cSeg6,alphaSeg6,betaSeg6); Disp(buffer);
    sprintf(buffer,"cSeg7, alphaSeg7, betaSeg7: %f,%f,%f",
      cSeg7,alphaSeg7,betaSeg7); Disp(buffer);
    sprintf(buffer,"cSeg8, alphaSeg8, betaSeg8: %f,%f,%f",
      cSeg8,alphaSeg8,betaSeg8); Disp(buffer);
    sprintf(buffer,"Ntrain, trainFile: %i, %s",Ntrain,trainFile); Disp(buffer);
    sprintf(buffer,"temperatureFlag, temperatureFile: %i, %s",temperatureFlag,temperatureFile); Disp(buffer);
    sprintf(buffer,"QFlag, QFile: %i, %s",QFlag,QFile); Disp(buffer);
    sprintf(buffer,"nptspercycle, trainmult, termPulse: %i, %f, %i",
      nptspercycle,trainmult,termPulse); Disp(buffer);
    sprintf(buffer,"forceSingleThread, gearRadius, gearRatio: %i,%i,%i",
      forceSingleThread,gearRadius,gearRatio); Disp(buffer);
    sprintf(buffer,"isoZ(outer), isoZ(inner): %f, %f",isoZ[0],isoZ[1]); Disp(buffer);
    sprintf(buffer,"isoR(outer), isoR(inner): %f, %f",isoR[0],isoR[1]); Disp(buffer);
    sprintf(buffer,"initFrom, initFile: %i, %s",initFrom,initFile); Disp(buffer);
    sprintf(buffer,"restoreFlag, restoreFile: %i, %s",restoreFlag,restoreFile); Disp(buffer);
  #endif

  // Basic Sanity Check on configurations
  if ((Kmax % 2)==1) {
    Disp("Parameter Error: Kmax must be an even number.");
    if ((absorbTop==1)∧(absorbBase==1))
      return 1; // This is an isolated failure.
    else
      return 2; // Other jobs will fail due to this failure.
  }

  if ((Imax<5) ∧ ((absorbBase==0)∨(absorbTop==0))) {
    Disp("Parameter Error: Imax must be at least 5 for distributed processing.");
    if ((absorbTop==1)∧(absorbBase==1))
      return 1; // This is an isolated failure.
    else
      return 2; // Other jobs will fail due to this failure.
  }

  if (gearRadius > (Jmax-1)) {
    Disp("Parameter Error: gearRadius cannot be that close to Jmax.");
    if ((absorbTop==1)∧(absorbBase==1))
      return 1; // This is an isolated failure.
    else
      return 2; // Other jobs will fail due to this failure.
  }
```

```
if ((gearRadius>0) ∧ (gearRadius<3)) {
  Disp("Parameter Error: gearRadius must be at least 3.");
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

if (Nmax<nptspercycle) {
  Disp("Duration Error: can't run the simulation for less than one wave period.");
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

if (ffrExtent<Imax) {
  Disp("FFR Error: ffrExtent can't be less than Imax.");
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

// Calculate notable acoustic properties.
UltrasoundProperties3D(&zmax,&dz,&Imax,&Jmax,&Kmax,&rmax,&dr,&tmax,
            &dt,&dth,&Nmax,&period,&nptspercycle,&freq,&omega,
            &lambda,&tendtoend,&nendtoend,&trainmult,&CFL,&kwave,
            &zshock,&Mach,&Gold,&Umax);

#ifdef DIAGNOSTIC
  sprintf(buffer,"Zmax, Rmax, Tmax: %f,%f,%f",zmax,rmax,tmax); Disp(buffer);
  sprintf(buffer,"Period, Frequency, Omega: %f,%f,%f",period,freq,omega); Disp(buffer);
  sprintf(buffer,"Zshock, Umax: %f,%f",zshock,Umax); Disp(buffer);
  sprintf(buffer,"Mach, Goldberg: %f,%f",Mach,Gold); Disp(buffer);
  sprintf(buffer,"CFL: %f",CFL); Disp(buffer);
  Disp("");
#endif

// Array dynamic allocation. (This was much easier in Fortran 90!)
// A note on the structure of the arrays:
// In the 3D arrays, the i co-ordinate is the most rapidly changing,
// followed by the j co-ordinate, and finally the k co-ordinate.
// This makes FDTD3DC compatible with the original data order in the original
// F90 implementation, and also with MatLab. This allows whole arrays to read
// in and dumped out using a single file read or write operation.

jInc = Imax;
kInc = Imax*Jmax;
volSize = Imax*Jmax*Kmax;
jIncFFR = ffrExtent;
kIncFFR = ffrExtent*Jmax;
ffrVolSize = ffrExtent*Jmax*Kmax;

iSource   = (int *)malloc((unsigned)Nsources*sizeof(int));
jSource   = (int *)malloc((unsigned)Nsources*sizeof(int));
kSource   = (int *)malloc((unsigned)Nsources*sizeof(int));
ampSource = (float *)malloc((unsigned)Nsources*sizeof(float));
shiftSource = (int *)malloc((unsigned)Nsources*sizeof(int));
softSource  = (int *)malloc((unsigned)Nsources*sizeof(int));
represents  = (int *)malloc((unsigned)Nsources*sizeof(int));
iTarget   = (float *)malloc((unsigned)Ntargets*sizeof(float));
jTarget   = (float *)malloc((unsigned)Ntargets*sizeof(float));
kTarget   = (float *)malloc((unsigned)Ntargets*sizeof(float));
ampTarget = (float *)malloc((unsigned)Ntargets*sizeof(float));
phiTarget = (float *)malloc((unsigned)Ntargets*sizeof(float));
train   = (IFP *)malloc((unsigned)Ntrain*sizeof(IFP));
rho     = (float *)malloc((unsigned)volSize*sizeof(float));
seg     = (unsigned char *)malloc((unsigned)volSize*sizeof(unsigned char));
pnplus1   = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
pn      = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
pnminus1  = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
pnminus2  = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
pnminus3  = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
ppeak   = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
fourierA  = (float *)malloc((unsigned)ffrVolSize*sizeof(float));
fourierB  = (float *)malloc((unsigned)ffrVolSize*sizeof(float));

if (EXTENDED_ALPHA≠0) {
  pnminus4= (IFP *)malloc((unsigned)volSize*sizeof(IFP));
  pnminus5= (IFP *)malloc((unsigned)volSize*sizeof(IFP));
} else {
  pnminus4= NULL;
```

```
  pnminus5= NULL;
}
if (temperatureFlag≠0)
  temperature = (IFP *)malloc((unsigned)volSize*sizeof(IFP));
if (QFlag≠0)
  Q = (IFP *)malloc((unsigned)volSize*sizeof(IFP));

// Check that all requested memory was successfully acquired.
if ((iSource≡NULL)∨(jSource≡NULL)∨(kSource≡NULL)∨
  (ampSource≡NULL)∨(shiftSource≡NULL)∨(softSource≡NULL)∨
  (iTarget≡NULL)∨(jTarget≡NULL)∨(kTarget≡NULL)∨
  (ampTarget≡NULL)∨(phiTarget≡NULL)∨
  (fourierA≡NULL)∨(fourierB≡NULL)∨(represents≡NULL)∨
  (seg≡NULL)∨(rho≡NULL)∨(train≡NULL)∨
  (pnplus1≡NULL)∨(pn≡NULL)∨(pnminus1≡NULL)∨
  (pnminus2≡NULL)∨(pnminus3≡NULL)∨(ppeak≡NULL)∨
  ((EXTENDED_ALPHA≠0)∧((pnminus4≡NULL)∨(pnminus5≡NULL)))∨
  ((temperature≡NULL)∧(temperatureFlag≠0))∨
  ((Q≡NULL)∧(QFlag≠0))) {
  Disp("");
  Disp("Failed to allocate sufficient memory.");
  RELEASE_RESOURCES;
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

// Obtain excitation waveforms, material properties, array configuration

if (ReadTrain3D(train,Ntrain,trainFile)) {
  Disp("The waveform data is incomplete.");
  RELEASE_RESOURCES;
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

if (Materials3D(rhoFile,segFile,rho,seg)) {
  Disp("The materials data is incomplete.");
  RELEASE_RESOURCES;
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

if (ArrayDefnHu3Dm(arrayFile,targetFile,globalBase,globalTop,
      iSource,jSource,kSource,ampSource,shiftSource,softSource,represents,&Nsources,
      iTarget,jTarget,kTarget,&Ntargets)) {
  Disp("The array or target definitions were incomplete.");
  RELEASE_RESOURCES;
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

for (i=0; i<Ntargets; i++) {
  if (iTarget[i]<waveTrap) {
    Disp("Wavetrap error: Can't have any targets above the wavetrap.");
    if ((absorbTop≡1)∧(absorbBase≡1))
      return 1; // This is an isolated failure.
    else
      return 2; // Other jobs will fail due to this failure.
  }
}

// Initialize the memory space, either from a previously stored file, or set
// the entire working area to zeros.

failFlag=Initialize3D(pnplus1,pn,pnminus1,pnminus2,pnminus3,pnminus4,pnminus5,
  ppeak,fourierA,fourierB,initFrom,initFile,QFlag,Q);
if (failFlag) {
  Disp("The simulation could not be initialized as requested.");
  RELEASE_RESOURCES;
  if ((absorbTop≡1)∧(absorbBase≡1))
    return 1; // This is an isolated failure.
  else
    return 2; // Other jobs will fail due to this failure.
}

// Initialise the simulation volume by Huygens-Fresnel if requested.
```

```
if (bootStep>0) {
  if (initFrom<0) {
    if (BootFDTD3Dm(iSource,jSource,kSource,ampSource,shiftSource,softSource,represents,Nsources,
      bootStep,cH2O*dt*nptspercycle,nptspercycle,pn,pnminus1,pnminus2,pnminus3,pnminus4,
      pnminus5,dz,dr,dt,trainmult,Ntrain,Imax,Jmax,Kmax,globalBase,gearRadius,gearRatio,focus)) {
      Disp("Boot Error: Couldn't obtain enough memory.");
      RELEASE_RESOURCES;
      if ((absorbTop=1)∧(absorbBase=1))
        return 1; // This is an isolated failure.
      else
        return 2; // Other jobs will fail due to this failure.
    }
  } else {
    Disp("Cannot both initialize and perform Huygens-Fresnel boot-up.");
    RELEASE_RESOURCES;
    if ((absorbTop=1)∧(absorbBase=1))
      return 1; // This is an isolated failure.
    else
      return 2; // Other jobs will fail due to this failure.
  }
}

// Determine the step from which the simulation will begin
if (initFrom<0) {
  if (bootStep>0) initFrom=bootStep;
  else initFrom=0;
}
// Report the source driving conditions.
if (termPulse<0) {
  Disp("Source excitation continues indefinitely.");
  termPulse=initFrom+Nmax;}
else {
  sprintf(buffer,"Source excitation terminates after %i steps.",termPulse);
  Disp(buffer);
}

// Comment on the bone function.
if (cBoneFunction≠0)
  Disp("Using customized bone speed function.");
else
  Disp("Using built-in bone speed function.");

// Comment on the bone absorption function.
if (cBoneFunctionAlpha≠0)
  Disp("Using customized bone absorption function.");
else
  Disp("Using built-in bone absorption function.");

// Report the material models being used.
if (temperatureFlag≠0)
  Disp("Using temperature-varying tissue models.");
else
  Disp("Using constant property tissue models.");


// Test single plane dump here.
if ((planeFlag ≠ 0)∧(planeFlag≤Kmax)) {
  sprintf(buffer,"%s%05i.dpl.gz",planeProto,initFrom);
  if ((fileHandle = netOpen(buffer, "wb", _SH_DENYWR)) ≠ NULL) {
    gzFileHandle = gzdopen((_fileno)(fileHandle),"wb");
    gzwrite(gzFileHandle,&pn[x(1,1,planeFlag)],sizeof(IFP)*Imax*Jmax);
    gzwrite(gzFileHandle,&pn[x(1,1,(planeFlag+Kmax/2-1)%Kmax + 1)],sizeof(IFP)*Imax*Jmax);
    gzclose(gzFileHandle);
    fclose(fileHandle);}
  else {
    Disp("This file could not be created.");
  }
}

// Place the first sources.
SetSource3D(pn,(IFP)(initFrom+1),termPulse,trainmult,dt,freq,
  Nsources,iSource,jSource,kSource,ampSource,shiftSource,softSource,
  Ntrain,train,0,gearRadius,1,globalBase);

// Begin the simulation.
Disp("Beginning the simulation.");
for ( n=(initFrom+1); n≤(initFrom+Nmax) ; n++ ) {
  // Show the step-by-step progress
  DispProgress(n,Nmax,initFrom);

  // Calculate the pressure field with FDTD4s2t3d
  // Computes to fourth order spatial, second order time accuracy in 3D.
  if (gearRadius≤0) {
    // Ungeared operation. This is straightforward.
```

```
    // Advance a time step
    fdtd(pn,pnplus1,pnminus1,pnminus2,pnminus3,pnminus4,pnminus5,
      seg,absorbTop,absorbBase,dz,dr,dt,dth,Imax,Jmax,Kmax,
      rho,freq,omega,1,1,Jmax,1,0,forceSingleThread,isoZ,isoR);

    // Put in the new sources.
    SetSource3D(pnplus1,(IFP)(n+1),termPulse,trainmult,dt,freq,
      Nsources,iSource,jSource,kSource,ampSource,shiftSource,softSource,
      Ntrain,train,0,gearRadius,1,globalBase);

    // Update the arrays to new values in time
    Update3D(Imax,Jmax,Kmax,pn,pnplus1,pnminus1,pnminus2,pnminus3,pnminus4,pnminus5,n,0,
      netPath,seqIdent,absorbBase,absorbTop,globalBase,globalTop,1,Jmax);

  } else {
    // Geared operation. Differential time-stepping of the mesh.

    // Advance the outer region first.
    fdtd(pn,pnplus1,pnminus1,pnminus2,pnminus3,pnminus4,pnminus5,
      seg,absorbTop,absorbBase,dz,dr,dt,dth,Imax,Jmax,Kmax,
      rho,freq,omega,1,gearRadius+1,Jmax,1,0,forceSingleThread,isoZ,isoR);

    // Make sure the sources are set correctly for future times.
    SetSource3D(pnplus1,(IFP)(n+1),termPulse,trainmult,dt,freq,
      Nsources,iSource,jSource,kSource,ampSource,shiftSource,softSource,
      Ntrain,train,1,gearRadius,1,globalBase);

    // Now step the inner portion of the mesh forward in smaller time-steps.
    for ( minorStep=0 ; minorStep<(gearRatio-1) ; minorStep++ ) {
      //Possibly display substep number here if needed for debugging

      // Calculate this minor step with FDTD4s2t3d.
      fdtd(pn,pnplus1,pnminus1,pnminus2,pnminus3,pnminus4,pnminus5,
        seg,absorbTop,absorbBase,dz,dr,dt,dth,Imax,Jmax,Kmax,rho,freq,omega,
        2,1,gearRadius,gearRatio,minorStep,forceSingleThread,isoZ,isoR);

      // Set any sources within the geared volume.
      SetSource3D(pnplus1,(IFP)n+((minorStep+1)/gearRatio),termPulse,trainmult,dt,freq,
        Nsources,iSource,jSource,kSource,ampSource,shiftSource,softSource,
        Ntrain,train,2,gearRadius,gearRatio,globalBase);

      // Update the arrays for the next minor step.
      if (minorStep<(gearRatio-1))
        Update3D(Imax,Jmax,Kmax,pn,pnplus1,pnminus1,
          pnminus2,pnminus3,pnminus4,pnminus5,
          n-1,minorStep+1,netPath,seqIdent,
          absorbBase,absorbTop,globalBase,globalTop,
          1,gearRadius);
      else
        Update3D(Imax,Jmax,Kmax,pn,pnplus1,pnminus1,
          pnminus2,pnminus3,pnminus4,pnminus5,
          n,0,netPath,seqIdent,
          absorbBase,absorbTop,globalBase,globalTop,
          1,gearRadius);

      // Check for peak values in the inner volume.
      if ((initFrom+Nmax-n)<(NcycleAvg*nptspercycle)) {
        for (k=1 ; k<=Kmax ; k++ ) {
          for (j=1 ; j<=gearRadius ; j++) {
            for (i=1 ; i<=Imax ; i++ ) {
              ppeak[x(i,j,k)]=__max(ppeak[x(i,j,k)],(IFP)fabs(pn[x(i,j,k)]));
            }
          }
        }
      }

    } // End of minorStep loop for the inner portion of the geared mesh.

    // Update the outer volume to new values in time.
    Update3D(Imax,Jmax,Kmax,pn,pnplus1,pnminus1,
      pnminus2,pnminus3,pnminus4,pnminus5,
      n,0,netPath,seqIdent,
      absorbBase,absorbTop,globalBase,globalTop,
      gearRadius+1,Jmax);

  } // End of ungeared/geared decision.

  // Check the pressure array to see whether the solution has become unstable.
  for (i=0 ; i<volSize ; i++) {
    if (_isnan(pnplus1[i])) {
      sprintf(buffer,"NaN at index %i",i);
      Disp(buffer);
      Disp("Remember: arrays are zero-referenced in C.");
```

```
      Disp("Aborting due to instability.");
      RELEASE_RESOURCES;
      if ((absorbTop=1)∧(absorbBase=1))
        return 1; // This is an isolated failure.
      else
        return 2; // Other jobs will fail due to this failure.
    }
  }

  // Check if the simulation has been paused externally
  // Again, only of relevance within Rainfall.
  while (checkPaused()) {
    Sleep(10000);
  }

  // Check if this run has been aborted externally.
  // This depends on the wrapping code. When running from the command line
  // this never happens. Rainfall can, however, externally abort a run.
  if (checkRunAborted()) {
    Disp("The simulation run is being terminated.");
    RELEASE_RESOURCES;
    return 1;
  }

  // Store peak values (if appropriate) across array
  if ((initFrom+Nmax-n)<(NcycleAvg*nptspercycle)) {
    for (i=0 ; i<volSize ; i++ ) {
      ppeak[i]=__max(ppeak[i],(IFP)fabs(pn[i]));
    }
  }

  // Store Q data if requested
  if ((n+nptspercycle)>(Nmax+initFrom)) {
    if (QFlag≠0) {
      for (i=0 ; i<volSize ; i++) {
        // Creates the progressive sum in Q of (4*dt*dt)*(dp/dt)*(dp/dt)
        Q[i]+=(IFP)((3*pn[i]-4*pnminus1[i]+pnminus2[i])*(3*pn[i]-4*pnminus1[i]+pnminus2[i]));
      }
    }
  }

  // Do wavetrapping
  if ((n+nptspercycle)>(Nmax+initFrom)) {
    if (waveTrap>0) {
      // fourierP and fourierQ are the Phase and Quadrature components
      // needed to do this transform.
      fourierP=WaveGen3D((IFP)n,dt,freq,0,Ntrain,train);
      fourierQ=WaveGen3D((IFP)(n+0.25*nptspercycle),dt,freq,0,Ntrain,train);
      for (k=1; k≤Kmax; k++) {
        for (j=1; j≤Jmax; j++) {
          index=x(1,j,k);
          indexFFR=((k-1)*Jmax+(j-1))*ffrExtent;
          for (i=1 ; i≤Imax ; i++) {
            // Sum the orthogonal fourier components.
            fourierA[indexFFR]+=pn[index]*fourierP;
            fourierB[indexFFR++]+=pn[index++]*fourierQ;
          }
        }
      }
    }
  }

  // Write out the single plane data if this was requested.
  if ((planeFlag ≠ 0)∧(planeFlag≤Kmax)) {
    sprintf(buffer,"%s%05i.dpl.gz",planeProto,n);
    if ((fileHandle = netOpen(buffer, "wb", _SH_DENYWR)) ≠ NULL) {
      gzFileHandle = gzdopen((_fileno)(fileHandle),"wb");
      gzwrite(gzFileHandle,&pn[x(1,1,planeFlag)],sizeof(IFP)*Imax*Jmax);
      gzwrite(gzFileHandle,&pn[x(1,1,(planeFlag+Kmax/2-1)%Kmax + 1)],sizeof(IFP)*Imax*Jmax);
      gzclose(gzFileHandle);
      fclose(fileHandle);}
    else {
      Disp("This file could not be created.");
    }
  }

  // Write out decimated data if this was requested.
  if ((decimFlag ≠ 0) ∧ (n % decimTime ≡ 0)) {
    Disp("Writing out decimated file... ");
    sprintf(buffer,"%s%05i.dcm.gz",decimProto,n);
    if ((fileHandle = netOpen(buffer, "wb", _SH_DENYWR)) ≠NULL) {
      gzFileHandle = gzdopen((_fileno)(fileHandle),"wb");
      if (decimSpace>1) {
        for (k=1 ; k≤Kmax ; k+=decimSpace) {
```

```
            for (j=1 ; j≤Jmax ; j+=decimSpace) {
              for (i=1 ; i≤Imax ; i+=decimSpace) {
                gzwrite(gzFileHandle,&pn[x(i,j,k)],sizeof(IFP));
              }
            }
          }
        } else {
          gzwrite(gzFileHandle,pn,sizeof(IFP)*volSize);
        }
        gzclose(gzFileHandle);
        fclose(fileHandle);
      }
      else {
        Disp("This file could not be created.");
      }
    }

    // Announce overall progress
    if ((n%(int)(ceil((float)Nmax/20))) ≡ 0) {
      sprintf(buffer,"%i%% complete.",(int)(100*(n-initFrom)/Nmax));
      Disp(buffer);
    }
  } // End of loop for step number n.

  if (waveTrap>0) {
    // Do the fourier post-processing for targets.
    // Convert the wavetrap data into amplitude and phase form.
    for (i=0 ; i<ffrVolSize ; i++) {
      thisAmp=(float)(2*sqrt(fourierA[i]*fourierA[i]+fourierB[i]*fourierB[i])/nptspercycle);
      thisPhi=(float)(atan2(fourierB[i],fourierA[i]));
      fourierA[i]=thisAmp;
      fourierB[i]=thisPhi;
    }

    FFRfilldown(fourierA,fourierB,waveTrap,Speed3D(x(waveTrap-globalBase+1,1,1),rho,seg,freq)*dt*npt
spercycle,
      Alpha3D(x(waveTrap-globalBase+1,1,1),rho,seg,freq),Ntargets,iTarget,jTarget,kTarget,ampTarget,
phiTarget,
      Imax,Jmax,Kmax,ffrExtent,dr,dz,globalBase,ffrFlag);

    // Write out the target history file.
    if (Ntargets>0) {
      if ((fileHandle = netOpen(historyFile, "wt", _SH_DENYWR)) ≠ NULL) {
        for (i=0 ; i<Ntargets ; i++) {
          fprintf(fileHandle,"[%f %f %f][%f %f]\n",iTarget[i],jTarget[i],kTarget[i],ampTarget[i],phiTarg
et[i]);
        }
        fclose(fileHandle);
      } else {
        Disp("This file could not be created.");
      }
    }

    // Write out the FFR file if requested
    if ((fileHandle = netOpen(ffrFile, "wb", _SH_DENYWR)) ≠ NULL) {
      if ( (gzFileHandle = gzdopen((_fileno)(fileHandle),"wb")) ≠ NULL) {
        gzwrite(gzFileHandle,fourierA,sizeof(IFP)*ffrVolSize);
        gzwrite(gzFileHandle,fourierB,sizeof(IFP)*ffrVolSize);
        gzclose(gzFileHandle);
      } else {
        Disp("Couldn't allocate the compression state...");
      }
      fclose(fileHandle);
    }
    else {
      Disp("This file could not be created.");
    }
  }

  // Write out the peak pressure file with decimation.
  if (peakFlag≠0) {
    if ((fileHandle = netOpen(peakFile, "wb", _SH_DENYWR)) ≠ NULL) {
      gzFileHandle = gzdopen((_fileno)(fileHandle),"wb");
      if (gzFileHandle=NULL) {
        Disp("Could not allocate the compression state...");
        Disp("Writing Peak Pressure file uncompressed.");
      }
      if (decimSpace>1) {
        if (gzFileHandle≠NULL) {
          for (k=1 ; k≤Kmax ; k+=decimSpace) {
            for (j=1 ; j≤Jmax ; j+=decimSpace) {
              for (i=1 ; i≤Imax ; i+=decimSpace) {
                gzwrite(gzFileHandle,&ppeak[x(i,j,k)],sizeof(IFP));
              }
```

```
            }
          }
        } else {
          for (k=1 ; k≤Kmax ; k+=decimSpace) {
            for (j=1 ; j≤Jmax ; j+=decimSpace) {
              for (i=1 ; i≤Imax ; i+=decimSpace) {
                fwrite(&ppeak[x(i,j,k)],sizeof(IFP),1,fileHandle);
              }
            }
          }
        }
      } else {
        if (gzFileHandle≠NULL)
          gzwrite(gzFileHandle,ppeak,sizeof(IFP)*volSize);
        else
          fwrite(ppeak,sizeof(IFP),volSize,fileHandle);
      }
      if (gzFileHandle≠NULL) gzclose(gzFileHandle);
      fclose(fileHandle);}
    else {
      Disp("This file could not be created.");
    }
  }

  // Write out the Q file if requested.
  if (QFlag≠0) {
    for (i=0 ; i<volSize ; i++) {
      // Multiplies the accumulated Q to produce the overall equation.
      // Accumulated Q is (4*dt*dt)*(dp/dt)*(dp/dt)
      // Result is ((2*alpha)/(rho*c*w^2))*mean((dp/dt)^2)
      Q[i]*=Alpha3D(i,rho,seg,freq)/(2*1000*rho[i]*Speed3D(i,rho,seg,freq)*omega*omega*dt*dt*nptsper
cycle);
    }
    if (gearRadius>0) {
      // Geared operation was performed... we must correct for
      // the difference in time-step.
      for (k=1 ; k≤Kmax ; k++) {
        for (j=1 ; j≤gearRadius ; j++) {
          for (i=1 ; i≤Imax ; i++) {
            Q[x(i,j,k)]*=gearRatio*gearRatio;
          }
        }
      }
    }
    if ((fileHandle = netOpen(QFile, "wb", _SH_DENYWR)) ≠ NULL) {
      if ( (gzFileHandle = gzdopen((_fileno)(fileHandle),"wb")) ≠ NULL) {
        gzwrite(gzFileHandle,Q,volSize*sizeof(IFP));
        gzclose(gzFileHandle);}
      else {
        Disp("Couldn't allocate the compression state...");
        Disp("Writing Q file uncompressed.");
        fwrite(Q,sizeof(IFP),volSize,fileHandle);
      }
      fclose(fileHandle);}
    else {
      Disp("The Q file could not be created.");
    }
  }

  // Write out the restore file if requested.
  if (restoreFlag≠0) {
    if ((fileHandle = netOpen(restoreFile, "wb", _SH_DENYWR)) ≠ NULL) {
      if ( (gzFileHandle = gzdopen((_fileno)(fileHandle),"wb")) ≠ NULL) {
        gzwrite(gzFileHandle,pn,sizeof(IFP)*volSize);
        gzwrite(gzFileHandle,pnminus1,sizeof(IFP)*volSize);
        gzwrite(gzFileHandle,pnminus2,sizeof(IFP)*volSize);
        gzwrite(gzFileHandle,pnminus3,sizeof(IFP)*volSize);
        if (EXTENDED_ALPHA≠0) {
          gzwrite(gzFileHandle,pnminus4,sizeof(IFP)*volSize);
          gzwrite(gzFileHandle,pnminus5,sizeof(IFP)*volSize);
        }
        gzwrite(gzFileHandle,ppeak,sizeof(IFP)*volSize);
        gzclose(gzFileHandle);
      } else {
        Disp("Couldn't allocate the compression state...");
        Disp("Writing Restore file uncompressed.");
        fwrite(pn,sizeof(IFP),volSize,fileHandle);
        fwrite(pnminus1,sizeof(IFP),volSize,fileHandle);
        fwrite(pnminus2,sizeof(IFP),volSize,fileHandle);
        fwrite(pnminus3,sizeof(IFP),volSize,fileHandle);
        if (EXTENDED_ALPHA≠0) {
          fwrite(pnminus4,sizeof(IFP),volSize,fileHandle);
          fwrite(pnminus5,sizeof(IFP),volSize,fileHandle);
        }
```

```
        fwrite(ppeak,sizeof(IFP),volSize,fileHandle);
      }
      fclose(fileHandle);
    }
    else
      Disp("This file could not be created.");
  }

  // Finished successfully.
  // Release the memory being used for arrays, signal and return.
  RELEASE_RESOURCES;
  Disp("Done.");

  return 0;
};

int x(int i, int j, int k) {
  return (i-1)+(j-1)*jInc+(k-1)*kInc;
}
```

```
LIBRARY HuFDTD3Dm.dll
EXPORTS RainfallProcess
```

```
// Header file for HuFDTD3D

// Includes

#include <iostream.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <float.h>
#include <math.h>
#include <stdio.h>
#include <share.h>
#include <memory.h>
#include "InternalPrecision.h"
#include "ioServices.h"
#include "Zlib113/zlib.h"

// Defines
#define RELEASE_RESOURCES \
    {free(iSource); free(jSource); free(kSource);\
    free(ampSource); free(shiftSource); free(softSource);\
    free(iTarget); free(jTarget); free(kTarget);\
    free(ampTarget); free(phiTarget); free(represents);\
    free(seg); free(rho); free(train); \
    free(pnplus1); free(pn); free(pnminus1);\
    free(pnminus2); free(pnminus3); free (ppeak);\
    free(fourierA); free(fourierB); \
    if (EXTENDED_ALPHA≠0) {free(pnminus4); free(pnminus5);} \
    if (temperatureFlag≠0) free(temperature);\
    if (QFlag≠0) free(Q);}

#define NcycleAvg 10

// Procedure definitions

extern int ReadConfig3D(char* inFile, int* Imax, int* Jmax, int* Kmax, int* bootStep, int* Nmax,
          float* dr, float* dz, float* dt,
          int* Nsources, char* arrayFile,
          int* waveTrap, int* Ntargets, char* targetFile,
          float *focus,
          char* historyFile,
          int* peakFlag, char* peakFile,
          int* ffrFlag, int* ffrExtent, char* ffrFile,
          char* segFile, char* rhoFile,
          int* absorbBase, int* absorbTop,
          int* globalBase, int* globalTop,
          char* netPath, char* seqIdent,
          int* decimFlag, int* decimSpace, int* decimTime,
          char* decimProto,
          int* planeFlag, char* planeProto,
          int* Ntrain, char* trainFile,
          int* QFlag, char* QFile,
          int* nptspercycle, float* trainmult, int* termPulse,
          int* forceSingleThread, int* gearRadius, int* gearRatio,
          float* isoZ, float* isoR,
          int *initFrom, char *initFile,
          int *restoreFlag, char *restoreFile);

extern void UltrasoundProperties3D(float *zmax, float *dz, int *Imax, int *Jmax,
              int *Kmax, float *rmax, float *dr, float *tmax,
              float *dt, float *dth, int *Nmax, float *period,
              int *nptspercycle, float *freq, float *omega,
              float *lambda, float *tendtoend, int *nendtoend,
              float *trainmult, float *CFL, float *kwave,
              float *zshock, float *Mach, float *Gold, float *Umax);

extern int ReadTrain3D(IFP *train, const int Ntrain, const char *trainFile);

extern int Materials3D( const char *rhoFile, const char *segFile,
            float *rho,unsigned char *seg);

extern int Initialize3D(IFP *pnplus1,IFP *pn,
            IFP *pnminus1,IFP *pnminus2,
            IFP *pnminus3,
            IFP *pnminus4,IFP *pnminus5,
            IFP *ppeak, float *fourierA, float *fourierB,
            int initFrom, char *initFile,
            int QFlag, IFP *Q);

extern int ArrayDefnHu3Dm(const char *arrayFile, const char *targetFile,
        const int globalBase, const int globalTop,
        int *iSource, int *jSource, int *kSource,
        float *ampSource, int *shiftSource, int *softSource,
        int *represents, int *Nsources,
```

```
        float *iTarget, float *jTarget, float *kTarget,
        int *Ntargets);

int x(int i, int j, int k);

extern void Update3D(int Imax, int Jmax, int Kmax, IFP *pn, IFP *pnplus1,
        IFP *pnminus1, IFP *pnminus2, IFP *pnminus3, IFP *pnminus4, IFP *pnminus5,
        int n, int minorStep, char *netPath, char *seqIdent,
        int absorbBase, int absorbTop,
        int globalBase, int globalTop,
        int Jinner, int Jouter);

extern void fdtd(IFP *pn,IFP *pnplus1,
  IFP *pnminus1, IFP *pnminus2, IFP *pnminus3, IFP *pnminus4, IFP *pnminus5,
  unsigned char *seg,
  int absorbTop, int absorbBase, float dz, float dr, float dt, float dth,
  int Imax, int Jmax, int Kmax, float *rho, float freq, float omega,
  int region, int Jinner, int Jouter, int gearRatio, int minorStep,
  int forceSingleThread, float *isoZ, float *isoR);

extern void SetSource3D(IFP *pThis, IFP step, int termPulse,
         float trainmult, float dt, float freq, int Nsources,
         int *iSource, int *jSource, int *kSource,
         float *ampSource, int *shiftSource, int *softSource,
         int Ntrain,IFP *train,int region,int gearRadius,
         int ratio,int globalBase);

extern int BootFDTD3Dm( int *iSource, int *jSource, int *kSource,
         float *ampSource, int *shiftSource, int *softSource, int *represents,
         int Nsources, int initFrom, float lambda, int nptspercycle,
         IFP *pn, IFP *pnminus1, IFP *pnminus2,
         IFP *pnminus3, IFP *pnminus4, IFP *pnminus5,
         float dz, float dr, float dt, float trainmult, int Ntrain,
         int Imax, int Jmax, int Kmax, int globalBase,
         int gearRadius, int gearRatio, float *focus);

extern int FFRfilldown(IFP *ffrA, IFP *ffrPhi, int Ifilldown, float lambdaID, float alphaID,
         int Ntargets, float *iTarget, float *jTarget, float *kTarget, float *ampTarget, float *phiT
arget,
         int Imax, int Jmax, int Kmax, int ffrExtent, float dr, float dz, int globalBase, int ffrFla
g);

extern IFP WaveGen3D(IFP step, float dt, float freq, int shift, int Ntrain, IFP *train);

extern IFP Alpha3D(int index, float *rho, unsigned char *seg, float freq);
extern IFP Speed3D(int index, float *rho, unsigned char *seg, float freq);

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// Initialize3D.cpp
// Zeros the memory allocated to the pressure arrays. That's all.
// The feature to restart a simulation part way through from stored
// data would be implemented here. This feature does not exist right now.

#include "Initialize3D.h"

int Initialize3D(IFP *pnplus1, IFP *pn,
         IFP *pnminus1,IFP *pnminus2,
         IFP *pnminus3,IFP *pnminus4,IFP *pnminus5,
         IFP *ppeak, float *fourierA, float *fourierB,
         int initFrom, char *initFile,
         int QFlag, IFP *Q) {

  // Create local variables.
  int   dataRead,dataNeeded;
  FILE* fileHandle;
  gzFile  gzFileHandle;

  // Clear the array for Q, if Q is being calculated.
  if (QFlag≠0) memset(Q,0,volSize*sizeof(IFP));

  if (initFrom < 0) {
    memset(pnplus1,0,volSize*sizeof(IFP));
    memset(pn,0,volSize*sizeof(IFP));
    memset(pnminus1,0,volSize*sizeof(IFP));
    memset(pnminus2,0,volSize*sizeof(IFP));
    memset(pnminus3,0,volSize*sizeof(IFP));
    memset(ppeak,0,volSize*sizeof(IFP));
    if (EXTENDED_ALPHA≠0) {
      memset(pnminus4,0,volSize*sizeof(IFP));
      memset(pnminus5,0,volSize*sizeof(IFP));
    }
  }
  else {
    // Set pnplus1 values to zero.
    memset(pnplus1,0,volSize*sizeof(IFP));

    // We are initialising the simulation from stored pressure data.
    if ((fileHandle=netOpen(initFile,"rb",_SH_DENYWR))≡NULL) return 1;
    gzFileHandle = gzdopen((_fileno)(fileHandle),"rb");
    dataRead =gzread(gzFileHandle,pn,sizeof(IFP)*volSize);
    dataRead+=gzread(gzFileHandle,pnminus1,sizeof(IFP)*volSize);
    dataRead+=gzread(gzFileHandle,pnminus2,sizeof(IFP)*volSize);
    dataRead+=gzread(gzFileHandle,pnminus3,sizeof(IFP)*volSize);
    if (EXTENDED_ALPHA≠0) {
      dataRead+=gzread(gzFileHandle,pnminus4,sizeof(IFP)*volSize);
      dataRead+=gzread(gzFileHandle,pnminus5,sizeof(IFP)*volSize);
    }
    dataRead+=gzread(gzFileHandle,ppeak,sizeof(IFP)*volSize);
    gzclose(gzFileHandle);
    fclose(fileHandle);

    dataNeeded=(signed)(5*volSize*sizeof(IFP));
    if (EXTENDED_ALPHA≠0) dataNeeded+=(signed)(2*volSize*sizeof(IFP));
    if (dataRead ≠ dataNeeded)
      return 1; // Some of the data was missing.
  }

  // Read in the temperature data if some has been specified
  if (temperatureFlag≠0) {
    if ((fileHandle=netOpen(temperatureFile,"rb",_SH_DENYWR))≡NULL) return 1;
    gzFileHandle = gzdopen((_fileno)(fileHandle),"rb");
    dataRead=gzread(gzFileHandle,temperature,sizeof(IFP)*volSize);
    gzclose(gzFileHandle);
    fclose(fileHandle);

    if (dataRead ≠ (signed)(volSize*sizeof(IFP)))
      return 1; // Some of the temperature data was missing.
  }

  memset(fourierA,0,ffrVolSize*sizeof(float));
  memset(fourierB,0,ffrVolSize*sizeof(float));

  return 0; // Completed successfully.
}
```

```
// Header file for Initialize3D

#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include <share.h>
#include "ioServices.h"
#include "Zlib113/zlib.h"
#include <stdio.h>
#include <string.h>
//#include <windows.h>

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// Sets the internal precision used for FDTD3DC
// Can either be set to float or double
// IFP stands for internal floating point.

//#define IFP double
#define IFP float

// EXTENDED_ALPHA defines whether the absorption characteristic is based on
// the third derivative calculation of absorption or on the first derivative.
// EXTENDED_ALPHA forces the use of the 3rd derivative, with the penalty of a
// greater memory footprint.
// #define EXTENDED_ALPHA 0
#define EXTENDED_ALPHA 0
```

```
// Defines the I/O to the user services available to the FDTD3DC core.
#include "..\RainfallServerDLL\StdAfx.h"
#define MAX_RETRIES 10

#ifdef __cplusplus
extern "C" {
#endif

__declspec( dllimport ) void Disp(CString text);
__declspec( dllimport ) void DispProgress(int n, int Nmax, int initFrom);
__declspec( dllimport ) int checkRunAborted();
__declspec( dllimport ) int checkPaused();

#ifdef __cplusplus
}
#endif
```

```
// Materials3D.cpp
// Reads the density and segmentation files into the arrays
// rho and seg from rhoFile and segFile respectively.

#include "Materials3D.h"

int Materials3D(const char* rhoFile, const char* segFile,
        float* rho,unsigned char* seg) {

  FILE* fileHandle;
  gzFile  gzFileHandle;
  int   rhoRead, segRead;

  if ((fileHandle=netOpen(rhoFile,"rb",_SH_DENYWR))==NULL) return 1;
  gzFileHandle = gzdopen((_fileno)(fileHandle),"rb");
  rhoRead=gzread(gzFileHandle,rho,sizeof(float)*volSize);
  gzclose(gzFileHandle);
  fclose(fileHandle);

  if ((fileHandle=netOpen(segFile,"rb",_SH_DENYWR))==NULL) return 1;
  gzFileHandle = gzdopen((_fileno)(fileHandle),"rb");
  segRead=gzread(gzFileHandle,seg,sizeof(unsigned char)*volSize);
  gzclose(gzFileHandle);
  fclose(fileHandle);

  if   ((rhoRead==(signed)(volSize*sizeof(float)))
    ^(segRead==(signed)(volSize*sizeof(unsigned char))))
    return 0; // Data read successfully
  else
    return 1; // Some of the material data was missing.
}
```

```
// Header file for Materials3D

#include <stdio.h>
#include <iostream.h>
#include <share.h>
//#include <windows.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include "ioServices.h"
#include "Zlib113/zlib.h"

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// Performs safe network-wide retry-and-quit file opening.
// Called by all other functions in FDTD3DC that want to open files.

#include "netOpen.h"

FILE *netOpen(const char *fileName, const char *mode, int share) {

  int   retries = MAX_RETRIES;
  FILE  *fileHandle;
  char  buffer[256];

  while ((¬(fileHandle = _fsopen(fileName, mode, share)))∧(retries>0)) {
    retries--;
    if (retries>0) {
      sprintf(buffer,"Retrying access to %s",fileName);
      Disp(buffer);
      Sleep(5000);
    } else {
      sprintf(buffer,"Failed to access %s",fileName);
      Disp(buffer);
    }
  }

  if (retries≡0)
    return NULL;
  else
    return fileHandle;
}
```

```
// Header file for netOpen

#include <stdio.h>
#include <share.h>
#include "ioServices.h"
#include <windows.h>
```

```
// ReadConfig3D.cpp
// Reads in the configuration file for the FDTD routines.

#include "ReadConfig3D.h"

int ReadConfig3D(char* inFile, int* Imax, int* Jmax, int* Kmax, int* bootStep, int* Nmax,
          float* dr, float* dz, float* dt,
          int* Nsources, char* arrayFile,
          int* waveTrap, int* Ntargets, char* targetFile,
          float *focus,
          char* historyFile,
          int* peakFlag, char* peakFile,
          int* ffrFlag, int* ffrExtent, char* ffrFile,
          char* segFile, char* rhoFile,
          int* absorbBase, int* absorbTop,
          int* globalBase, int* globalTop,
          char* netPath, char* seqIdent,
          int* decimFlag, int* decimSpace, int* decimTime,
          char* decimProto,
          int* planeFlag, char* planeProto,
          int* Ntrain, char* trainFile,
          int* QFlag, char* QFile,
          int* nptspercycle, float* trainmult, int* termPulse,
          int* forceSingleThread, int* gearRadius, int* gearRatio,
          float* isoZ, float* isoR,
          int *initFrom, char *initFile,
          int *restoreFlag, char *restoreFile) {

  FILE* fileHandle;
  char  buffer[256];

  if ((fileHandle=netOpen(inFile,"rt",_SH_DENYWR))==NULL) return 1;

  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i,%i,%i,%i",Imax,Jmax,Kmax,bootStep,Nmax);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",dz,dr,dt);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",Nsources,arrayFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i,'%[^']",waveTrap,Ntargets,targetFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%f,%f,%f",&focus[0],&focus[1],&focus[2]);
  fgets(buffer,256,fileHandle); sscanf(buffer,"'%[^']",historyFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",peakFlag,peakFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i,'%[^']",ffrFlag,ffrExtent,ffrFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"'%[^']','%[^']'",segFile,rhoFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i",absorbBase,absorbTop);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i",globalBase,globalTop);
  fgets(buffer,256,fileHandle); sscanf(buffer,"'%[^']','%[^']'",netPath,seqIdent);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i,%i",decimFlag,decimSpace,decimTime);
  fgets(buffer,256,fileHandle); sscanf(buffer,"'%[^']'",decimProto);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']'",planeFlag,planeProto);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e",&cH2O,&alphaH2O,&betaH2O,&rhoH2O);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",&cBrain,&alphaBrain,&betaBrain);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",&cExtra,&alphaExtra,&betaExtra);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",&cSeg6,&alphaSeg6,&betaSeg6);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",&cSeg7,&alphaSeg7,&betaSeg7);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e",&cSeg8,&alphaSeg8,&betaSeg8);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",Ntrain,trainFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",&temperatureFlag,&temperatureFile[0]);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",QFlag,QFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%e,%i",nptspercycle,trainmult,termPulse);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,%i,%i",forceSingleThread,gearRadius,gearRatio);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e",&isoZ[0],&isoZ[1]);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e",&isoR[0],&isoR[1]);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",initFrom,initFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i,'%[^']",restoreFlag,restoreFile);
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i",&cBoneFunction);
  if (cBoneFunction>0) {
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBone[0],&cBone[1],&cBone[2],&cBon
e[3],&cBone[4]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBone[5],&cBone[6],&cBone[7],&cBon
e[8],&cBone[9]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBone[10],&cBone[11],&cBone[12],&c
Bone[13],&cBone[14]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBone[15],&cBone[16],&cBone[17],&c
Bone[18],&cBone[19]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBone[20],&cBone[21],&cBone[22],&c
Bone[23],&cBone[24]);
  }
  fgets(buffer,256,fileHandle); sscanf(buffer,"%i",&cBoneFunctionAlpha);
  if (cBoneFunctionAlpha>0) {
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBoneAlpha[0],&cBoneAlpha[1],&cBon
eAlpha[2],&cBoneAlpha[3],&cBoneAlpha[4]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBoneAlpha[5],&cBoneAlpha[6],&cBon
eAlpha[7],&cBoneAlpha[8],&cBoneAlpha[9]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBoneAlpha[10],&cBoneAlpha[11],&cB
oneAlpha[12],&cBoneAlpha[13],&cBoneAlpha[14]);
```

```
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBoneAlpha[15],&cBoneAlpha[16],&cB
oneAlpha[17],&cBoneAlpha[18],&cBoneAlpha[19]);
    fgets(buffer,256,fileHandle); sscanf(buffer,"%e,%e,%e,%e,%e",&cBoneAlpha[20],&cBoneAlpha[21],&cB
oneAlpha[22],&cBoneAlpha[23],&cBoneAlpha[24]);
  }

  fclose(fileHandle);
  return 0;
}
```

```
// Header file for ReadConfig3D

#include <iostream.h>
#include <stdio.h>
//#include <windows.h>
#include <share.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include "ioServices.h"

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// ReadTrain3D.cpp
// Reads in the excitation waveform from trainFile.

#include "ReadTrain3D.h"

int ReadTrain3D(IFP *train, const int Ntrain, const char* trainFile) {
  int   numRead;
  float tempFloat;
  FILE* fileHandle;

  if ((fileHandle=netOpen(trainFile,"rt",_SH_DENYWR))==NULL) return 1;
  for (numRead=0 ; numRead<Ntrain ; numRead++) {
    if (fscanf(fileHandle,"%f\n",&tempFloat) == EOF) break;
    train[numRead]=(IFP)tempFloat;
  }
  fclose(fileHandle);

  if (numRead==Ntrain)
    return 0; // Got all the asked-for waveform data.
  else
    return 1; // Uh-oh... some data must be missing.
}
```

```
// Header file for ReadTrain3D

#include <stdio.h>
#include <iostream.h>
//#include <windows.h>
#include <share.h>
#include "InternalPrecision.h"
#include "ioServices.h"

extern FILE *netOpen(const char *fileName, const char *mode, int share);
```

```
// SetSource3D.cpp
// SetSource3D puts the sources of excitation into the array it's given
// according to whether they're hard or soft sources and where they
// are in the volume.

#include "SetSource3D.h"

void SetSource3D(IFP *pThis, IFP step, int termPulse,
         float trainmult, float dt, float freq, int Nsources,
         int *iSource, int *jSource, int *kSource,
         float *ampSource, int *shiftSource, int *softSource,
         int Ntrain,IFP *train,int region,int gearRadius,
         int ratio,int globalBase) {

  IFP thisWave;
  int   s;

  for (s=0 ; s<Nsources ; s++) {
    if ((region≡0) ∨
       ((region≡1)∧(jSource[s]>gearRadius)) ∨
       ((region≡2)∧(jSource[s]≤gearRadius))) {

      if (floor(step)≤termPulse)
        thisWave = ampSource[s]*trainmult*
          WaveGen3D(step,dt,freq,shiftSource[s],Ntrain,train);
      else
        thisWave = 0;

      if (softSource[s]≡0)
        // Hard Source
        pThis[x(iSource[s]-globalBase+1,jSource[s],kSource[s])] = thisWave;
      else
        // Soft Source
        pThis[x(iSource[s]-globalBase+1,jSource[s],kSource[s])] = thisWave +
          pThis[x(iSource[s]-globalBase+1,jSource[s],kSource[s])];

    }
  }
}
```

```
// Header file for SetSource3D

#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"

extern IFP WaveGen3D(IFP step, float dt, float freq,
         int shift, int Ntrain, IFP *train);
```

```
// Speed3D.cpp
// Calculates the speed of sound at a mesh point given the
// mesh segmentation data and (depending on the material) the density.

#include "Speed3D.h"

IFP Speed3D(int index, float *rho, unsigned char *seg, float freq) {

  /*
    Choose the algorithm for determining the speed of sound
    The material types are:
    0: Outside the original data set
    1: Water
    2: Outer cortical bone
    3: Trabecular bone
    4: Inner cortical bone
    5: Brain
    6: User-specified - often fat
    7: User-specified - often liver
    8: User-specified

    This routine may be extended later to accommodate more materials.
  */

  IFP thisSpeed,bias;
  int lookup;

  switch (seg[index]) {
  case 0: // Outside the data set
    return (IFP)cExtra;
    break;

  case 1: // In water
    if (temperatureFlag==0)
      return (IFP)cH2O;
    else
      // Greenspan M, Tschiegg C (1959) JASA 31:75
      return (IFP)( 1402.736
             +5.03358*temperature[index]
             -0.0579506*temperature[index]*temperature[index]
             +3.31636e-04*temperature[index]*temperature[index]*temperature[index]
             -1.45262e-06*temperature[index]*temperature[index]*temperature[index]*temperature[index
]
             +3.0449e-09*temperature[index]*temperature[index]*temperature[index]*temperature[index]
*temperature[index]);
    break;

  case 2: // In bone (combined bone model)
  case 3: // There is no temperature dependence included for this.
  case 4: // No model is available.
    /*
    // ---- Calculations/Models for Bone --------------------------------
    // In cortical bone (both inner and outer) and trabecular
    // This model was derived on August 31, 2000. It fuses the
    // low volume fraction model for trabecular bone with a square-root
    // curve for intermediate fractions in line with cellular solid theory,
    // coupled with a transition into the linear cortical region.

    // A strictly monotonic quintic function is fitted over density values
    // below 2.21 g/cc, above that we return to the linear cortical fit.

    IFP thisRho = (IFP)rho[index];
    if (rho[index]<=2.21) {
      // The original expression in Fortran (which is clearer) is:
      // Speed3D=1000*((3.5885*rho(i,j,k)**5)+(-33.0008*rho(i,j,k)**4)+&
      //        (119.1308*rho(i,j,k)**3)+(-210.1748*rho(i,j,k)**2)+&
      //        (182.0340*rho(i,j,k))-60.1703)
      //Unfortunately, C lacks the integer power expression.

      return 1000*(IFP)
            ((3.5885*thisRho*thisRho*thisRho*thisRho*thisRho)+
             (-33.0008*thisRho*thisRho*thisRho*thisRho)+
             (119.1308*thisRho*thisRho*thisRho)+
            (-210.1748*thisRho*thisRho)+
             (182.0340*thisRho)+
             (-60.1703));
    } else
      return (IFP)(1882*thisRho-700);
    */

    // We now introduce a model derived on March 20, 2001 which is
    // more compliant with Fry's speeds in bone, and also trends with
    // increasing density in a way that Greg and I feel is more natural.
    /*
```

```
    if (rho[index]<=1) {
      return (IFP)1500; // I claim this is actually void water.
    } else if (rho[index]>3.1) {
      return (IFP)3314.8; // Few points have greater density than this.
    } else {
      // This is a compact way of writing an order 7 polynomial fit...
      return (IFP)(((((((623.97*rho[index]-9572.49)*rho[index]
                        +62154.34)*rho[index]
                        -221375.35)*rho[index]
                        +467106.54)*rho[index]
                        -583996.70)*rho[index]
                        +401193.06)*rho[index]
                        -114612.22);
    }
    */

    // This is Greg's model for effective speed based on mean density.
    //thisSpeed=(IFP)(2063.2*rho[index]-1659.4+120);
    //if (thisSpeed<1500) thisSpeed=1500;
    //return thisSpeed;

    if (cBoneFunction==0) {
      // This is Greg's model for effective speed based on mean density.
      thisSpeed=(IFP)(2063.2*rho[index]-1659.4+120);
      if (thisSpeed<1500) thisSpeed=1500;
    } else {
      // Use the defined bone function in the config file.
      // The function spans the range of rho from 1.0 to 3.4
      bias=(IFP)((rho[index]-1.0)/0.1);
      lookup=(int)floor((double)bias);
      bias=bias-lookup;
      if (lookup<0) {lookup=0; bias=0;}
      if (lookup>=24) {
        thisSpeed=(IFP)cBone[24];
      } else {
        thisSpeed=(IFP)(cBone[lookup]*(1-bias) + cBone[lookup+1]*bias);
      }
    }
    return thisSpeed;
    break;

    // Kullervo's Test Case
    //return (IFP)cH2O;
    //break;

  case 5: // In brain
    return (IFP)cBrain;
    break;

  // Other user-specified tissues
  case 6: //fat
    if (temperatureFlag==0)
      return (IFP)cSeg6;
    else
      //Ibrahim Hallaj's polynomial fit from Bamber/Hill, Damianou
      return (IFP)( 1.7461e3
            -3.9308e0*temperature[index]
            -4.1282e-1*temperature[index]*temperature[index]
              +1.1373e-2*temperature[index]*temperature[index]*temperature[index]
            -1.1010e-4*temperature[index]*temperature[index]*temperature[index]*temperature[index]
            +3.7308e-7*temperature[index]*temperature[index]*temperature[index]*temperature[index]*
temperature[index]);
    break;

  case 7: //liver
    if (temperatureFlag==0)
      return (IFP)cSeg7;
    else
      //Ibrahim Hallaj's polynomial fit from Bamber/Hill, Damianou
      return (IFP)( 1.5293e3
            +1.6856e0*temperature[index]
            +6.1131e-2*temperature[index]*temperature[index]
            -2.2967e-3*temperature[index]*temperature[index]*temperature[index]
            +2.2657e-5*temperature[index]*temperature[index]*temperature[index]*temperature[index]
            -7.1795e-8*temperature[index]*temperature[index]*temperature[index]*temperature[index]*
temperature[index]);
    break;

  case 8: //buffer
    return (IFP)cSeg8;
    break;

  default:
    break;
```

```
  }

  // We should never actually reach here.
  return 0;
}
```

```
// Header file for Speed3D

#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
```

StdAfx.h


```
// stdafx.h : include file for standard system include files,
//  or project specific include files that are used frequently, but
//      are changed infrequently
//

#if ¬defined(AFX_STDAFX_H__EC048FEE_1C5E_11D4_A80C_00010229A291__INCLUDED_)
#define AFX_STDAFX_H__EC048FEE_1C5E_11D4_A80C_00010229A291__INCLUDED_

#if _MSC_VER ≥ 1000
#pragma once
#endif // _MSC_VER >= 1000

#define VC_EXTRALEAN    // Exclude rarely-used stuff from Windows headers

#include <afxwin.h>         // MFC core and standard components
#include <afxext.h>         // MFC extensions

#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>     // MFC support for Windows Common Controls
#endif // _AFX_NO_AFXCMN_SUPPORT


//{{AFX_INSERT_LOCATION}}
// Microsoft Developer Studio will insert additional declarations immediately before the previous li
ne.

#endif // !defined(AFX_STDAFX_H__EC048FEE_1C5E_11D4_A80C_00010229A291__INCLUDED_)
```

```
// UltrascundProperties3D.cpp
// UltrascundProperties3D calculates a number of useful physical
// quantities relevant to FDTD processing and to ultrasound in general.

#include "UltrasoundProperties3D.h"

void UltrasoundProperties3D(float* zmax, float* dz, int* Imax, int* Jmax,
              int* Kmax, float* rmax, float* dr, float* tmax,
              float* dt, float* dth, int* Nmax, float* period,
              int* nptspercycle, float* freq, float* omega,
              float* lambda, float* tendtoend, int* nendtoend,
              float* trainmult, float* CFL, float* kwave,
              float* zshock, float* Mach, float* Gold, float* Umax) {

//  const double pi = 4*atan(1);

  (*zmax)      = (*dz) * (*Imax);
  (*rmax)      = (*dr) * ((*Jmax)+float(0.5));
  (*tmax)      = (*dt) * (*Nmax);
  (*tendtoend)  = (*dt)/cH2O;
  (*nendtoend)  = (int)floor((*tendtoend)/(*dt));
  (*dth)       = (float)(2*pi/(*Kmax));
  (*period)    = (*dt) * (*nptspercycle);
  (*freq)      = 1/(*period);
  (*omega)     = (float)(2*pi*(*freq));
  (*lambda)    = cH2O*(*period);
  (*kwave)     = (*omega)/cH2O;
  (*Umax)      = (*trainmult)/(1000*rhoH2O*cH2O);
  (*Mach)      = (*Umax)/cH2O;
  (*Gold)      = betaH2O*(*Mach)/(alphaH2O*cH2O/(*omega));
  (*zshock)    = 1/(betaH2O*(*Mach)*(*kwave));
  (*CFL)       = cH2O*(*dt)/(*dz);
}
```

```
// Header file for UltrasoundProperties3D

#include <iostream.h>
#include <math.h>
#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
```

```
// Update3D.cpp
// Does array to array copying at the end of each processing step.
// This is straightforward for non-distributed methods.
// This subroutine also handles the flipping of array volumes across
// the network for distributed processing, which is more complicated.

#include "Update3D.h"

void Update3D(int Imax, int Jmax, int Kmax, IFP *pn, IFP *pnplus1,
        IFP *pnminus1, IFP *pnminus2, IFP *pnminus3, IFP *pnminus4, IFP *pnminus5,
        int n, int minorStep, char *netPath, char *seqIdent,
        int absorbBase, int absorbTop,
        int globalBase, int globalTop,
        int Jinner, int Jouter) {

  int   j,k,index;
  char  buffer[256];
  FILE  *fileHandle;

  // Data writing - present data at exchange interfaces.

  if ((absorbBase≠1)∧(¬checkRunAborted())) {
    // Write-Exchange at the volume base.

    // Create the filename, and create the file.
    sprintf(buffer,"%s%s-%i-%i-%i-%i-%i-%i.int",netPath,seqIdent,
      n,minorStep,Jinner,Jouter,globalBase+2,globalBase+3);
    while (¬(fileHandle = _fsopen(buffer, "wb", _SH_DENYRW))) {
      Sleep(5000);}

    // Write the data to the file.
    for (k=1 ; k≤Kmax; k++) {
      for (j=Jinner ; j≤Jouter ; j++) {
        index=x(3,j,k);
        fwrite(&pnplus1[index],sizeof(IFP),2,fileHandle);
      }
    }

    // We're done with this file.
    fclose(fileHandle);
  }

  if ((absorbTop≠1)∧(¬checkRunAborted())) {
    // Write-Exchange at the volume top.

    // Create the filename, and create the file.
    sprintf(buffer,"%s%s-%i-%i-%i-%i-%i-%i.int",netPath,seqIdent,
      n,minorStep,Jinner,Jouter,globalTop-3,globalTop-2);
    while (¬(fileHandle = _fsopen(buffer, "wb", _SH_DENYRW))) {
      Sleep(5000);}

    // Write the data to the file.
    for (k=1 ; k≤Kmax; k++) {
      for (j=Jinner ; j≤Jouter ; j++) {
        index=x(Imax-3,j,k);
        fwrite(&pnplus1[index],sizeof(IFP),2,fileHandle);
      }
    }

    // We're done with this file.
    fclose(fileHandle);
  }

  // Data reading - retrieve data at exchange interfaces.

  if ((absorbBase≠1)∧(¬checkRunAborted())) {
    // Read-Exchange at the volume base.

    // Create the filename, and look for the file.
    sprintf(buffer,"%s%s-%i-%i-%i-%i-%i-%i.int",netPath,seqIdent,
      n,minorStep,Jinner,Jouter,globalBase,globalBase+1);
    while (¬(fileHandle = _fsopen(buffer, "rb", _SH_DENYRW))) {
      Sleep(5000);}

    // Read the data from the file.
    for (k=1 ; k≤Kmax; k++) {
      for (j=Jinner ; j≤Jouter ; j++) {
        index=x(1,j,k);
        fread(&pnplus1[index],sizeof(IFP),2,fileHandle);
      }
    }

    // We're done with this file - close and delete it.
    fclose(fileHandle); remove(buffer);
```

```
  }

  if ((absorbTop≠1)∧(¬checkRunAborted())) {
    // Read-Exchange at the volume top.

    // Create the filename, and look for the file.
    sprintf(buffer,"%s%s-%i-%i-%i-%i-%i-%i.int",netPath,seqIdent,
      n,minorStep,Jinner,Jouter,globalTop-1,globalTop);
    while (¬(fileHandle = _fsopen(buffer, "rb", _SH_DENYRW))) {
      Sleep(5000);}

    // Read the data from the file.
    for (k=1 ; k≤Kmax; k++) {
      for (j=Jinner ; j≤Jouter ; j++) {
        index=x(Imax-1,j,k);
        fread(&pnplus1[index],sizeof(IFP),2,fileHandle);
      }
    }

    // We're done with this file - close and delete it.
    fclose(fileHandle); remove(buffer);
  }

  // Now we perform the standard updating of arrays.
  // Copying from one array to another.
  // This is a little neater in F90: C lacks implicit loops.
  for (k=1 ; k≤Kmax; k++) {
    for (j=Jinner ; j≤Jouter ; j++) {
      index=x(1,j,k);
      if (EXTENDED_ALPHA≠0) {
        memcpy(pnminus5+index,pnminus4+index,Imax*sizeof(IFP));
        memcpy(pnminus4+index,pnminus3+index,Imax*sizeof(IFP));
      }
      memcpy(pnminus3+index,pnminus2+index,Imax*sizeof(IFP));
      memcpy(pnminus2+index,pnminus1+index,Imax*sizeof(IFP));
      memcpy(pnminus1+index,pn+index,Imax*sizeof(IFP));
      memcpy(pn+index,pnplus1+index,Imax*sizeof(IFP));
    }
  }
}
```

```
// Header file for Update3D

#include <stdio.h>
#include <iostream.h>
#include <string.h>
//#include <windows.h>
#include <share.h>

#include "CommonMaterialProperties.h"
#include "InternalPrecision.h"
#include "ioServices.h"
```

```
// WaveGen3D.cpp
// Given an array which defines one cycle of a periodic
// waveform, this routine returns the value at a particular
// FDTD step number by interpolating into the array.

#include "WaveGen3D.h"

IFP WaveGen3D(IFP step, float dt, float freq,
         int shift, int Ntrain, IFP *train) {

  IFP index, bias;

  index = (step-1)*(IFP)dt*(IFP)freq*(IFP)Ntrain;
  index += (IFP)shift;
  index = ((int)floor(index) % Ntrain) + index - (IFP)floor(index);

  bias = index - (IFP)floor(index);

  if (index≥(Ntrain-1))
    return (((1-bias)*train[Ntrain-1])+(bias*train[0]));
  else
    return (((1-bias)*train[(int)floor(index)])+
      (bias*train[(int)ceil(index)]));
}
```

```
// Header file for WaveGen3D

#include <math.h>
#include "InternalPrecision.h"
```

Example Configuration File


```
370, 56, 48, 1149, 602 / Imax,Jmax,Kmax,bootStep,Nmax
2.529513e-004, 2.529513e-004, 3.465003e-008 / dz,dr,dt
2544, '\\fdtdServer\C\SkullStudy\skull6\source-100.arr' / nSources,arrayFile
357, 0, '\\fdtdServer\C\SkullStudy\skull6\source-100.tgt' / waveTrap,nTargets,targetFile
653.999889, 0.500360, 45.853687 / iFocus,jFocus,kFocus
'\\fdtdServer\C\SkullStudy\skull6\volume100-1.hst' / historyFile
0, '\\fdtdServer\C\SkullStudy\skull6\volume100-1.ppk.gz' / peakFlag, peakFile
1, 720, '\\fdtdServer\C\SkullStudy\skull6\volume100-1.ffr.gz' / ffrFlag, ffrExtent, ffrFile
'\\fdtdServer\C\SkullStudy\skull6\volume100-1.seg.gz', '\\fdtdServer\C\SkullStudy\skull6\volume100-1.rho.gz' / segFile, rhoFile
1, 1 / absorbBase, absorbTop
-2, 717 / globalBase, globalTop
'\\fdtdServer\C\SkullStudy\skull6\', 'DF2F7BD421' / netPath, seqIdent
0, 0, 0 / decimFlag, decimspace, decimtime
'\\fdtdServer\C\SkullStudy\skull6\volume100-1-' / decimProto
0, '\\fdtdServer\C\SkullStudy\skull6\volume100-1-OnePlane' / planeFlag, planeProto
1.500000e+003, 2.880000e-004, 3.500000e+000, 1.000000e+000 / cH2O,alphaH2O,betaH2O,rhoH2O (rho in g/cc)
1.545000e+003, 2.960000e+000, 0.000000e+000 / cBrain, alphaBrain, betaBrain
1.500000e+003, 2.880000e-004, 3.500000e+000 / cExtra, alphaExtra, betaExtra
1.430000e+003, 6.919400e+000, 1.050000e+001 / cSeg6,alphaSeg6,betaSeg6 (fat)
1.596000e+003, 3.561400e+000, 6.000000e+000 / cSeg7,alphaSeg7,betaSeg7 (liver)
1.500000e+003, 3.050000e+002, 0.000000e+000 / cSeg8,alphaSeg8,betaSeg8 (buffer)
3600, '\\quincy\d\sin3600-periodic.trn' / Ntrain, trainFile
0, 'none' / temperatureFlag, temperatureFile
0, 'none' / QFlag, QFile
39, 3.000000e+005, -1 / nptspercycle, trainmult, termPulse
0, 9, 16 / forceSingleThread, gearRadius, gearRatio
9.990834e-001, 9.979990e-001 / isoZ-region1, isoZ-region2
9.990834e-001, 9.979990e-001 / isoR-region1, isoR-region2
-1, 'none' / initFrom, initFile
0, 'none' / restoreFlag, restoreFile
1 / cBoneFunction follows below
1.815800e+003, 2.010300e+003, 2.173600e+003, 2.274600e+003, 2.305400e+003
2.299400e+003, 2.294000e+003, 2.318300e+003, 2.373500e+003, 2.455600e+003
2.560400e+003, 2.683800e+003, 2.821400e+003, 2.969400e+003, 3.123300e+003
3.279200e+003, 3.432900e+003, 3.580100e+003, 3.716800e+003, 3.839000e+003
3.946400e+003, 4.041800e+003, 4.128400e+003, 4.208900e+003, 4.286500e+003
1 / cBoneFunctionAlpha follows below
5.120869e+002, 4.556639e+002, 4.001435e+002, 3.464283e+002, 2.954210e+002
2.480242e+002, 2.051405e+002, 1.676725e+002, 1.365229e+002, 1.125942e+002
9.678913e+001, 9.001027e+001, 9.290427e+001, 1.037747e+002, 1.196682e+002
1.376294e+002, 1.547697e+002, 1.682281e+002, 1.755066e+002, 1.765931e+002
1.725165e+002, 1.643095e+002, 1.530048e+002, 1.396348e+002, 1.252322e+002
```

8.2 A Command Line Routine For Testing Rainfall Applications On A Lone Workstation.

main.cpp Page 1/2

```
// main.cpp
// Provides a wrapper for FDTD3DC that allows it to be run from
// the command line. By doing it this way, the core of FDTD3DC
// can be built into any application without further modification.
// Also provides the Disp routine for displaying text.

#include "main.h"

int main (int argc, char *argv[]) {

  int       status, retries;
  char      dllFile[256],inFile[256];
  HINSTANCE   hAlgorithm;
  RAINFALLPROCESS pRainfallProcess=0;
  FILE      *rfxFile;

  printf("%s : Rainfall Test Wrapper\n",argv[0]);
  printf("Written by Chris Connor. [cwc@mit.edu]\n\n");

  if ((argc ≠ 3)∧(argc ≠ 2)) {
    printf("Wrong number of parameters.\n");
    printf("Syntax: %s <application dll> <configuration file>\n", argv[0]);
    printf("    or %s <rainfall RFX file>\n",argv[0]);
    return 1;
  }

  if (argc=3) {
    strcpy(dllFile,argv[1]);
    strcpy(inFile,argv[2]);
  } else {
    if ( (rfxFile=fopen(argv[1],"rt")) = NULL) {
      printf("Couldn't open the specified RFX file.\n");
      return 1;
    }
  }

  for (;;) {
    if (argc=2) {
      if ( readNextLine(rfxFile,dllFile,inFile) = 1 ) {
        fclose(rfxFile);
        break;
      }
    }
    // Attempt to load the DLL specified in the process.
    printf("Applying %s to %s.\n",dllFile,inFile);
    retries = 5;
    while (((hAlgorithm = LoadLibrary((char *)dllFile)) = NULL)∧(retries>0)) {
      retries--;
      Sleep(5000);
    }

    if (retries>0) {
      pRainfallProcess = (RAINFALLPROCESS)GetProcAddress(hAlgorithm,"RainfallProcess");
      if (pRainfallProcess=0) retries=0;
    }

    if (retries=0) {
      printf("Couldn't load the specified algorithm: %s\n",dllFile);
      printf("Syntax: %s <application dll> <configuration file>\n", argv[0]);
      return 1;
    } else
      printf("Using algorithm: %s\n",dllFile);

    status=pRainfallProcess((char *)inFile);
    FreeLibrary(hAlgorithm);
    if (argc=3) break;
  }

  if (status=0)
    printf("Processing completed. OK.\n");
  else
    printf("Processing terminated irregularly.\n");
  return status;
}

int readNextLine(FILE *rfxFile,char *dllFile, char *inFile) {
  char buffer[256];

  fgets(buffer,256,rfxFile);
  if (¬feof(rfxFile)) {
    sscanf(buffer,"%*[^#]#%[^#]#%[^\n]",dllFile,inFile);
    return 0;
  } else {
    return 1;
```

```
  }
}

extern "C" void Disp(char *text) {
  char buffer[256], *s, *d;
  s=text; d=buffer;
  while ((*s)≠'\0') {
    *d=*s++;
    if ((*d)≡'%') {
      *(d+1)='%';
      d=d+2;
    }
    else
      d++;
  }
  *d='\0';

  printf(buffer);
  printf("\n");
}

extern "C" int checkRunAborted() {
  return 0;
}

extern "C" int checkPaused() {
  return 0;
}

extern "C" void DispProgress(int n, int Nmax, int initFrom) {
  char buffer[256];
  sprintf(buffer,"Progress: %i / %i",n,Nmax);
  Disp(buffer);
}
```

```
// Header file for main.cpp

#include <stdio.h>
#include <windows.h>
#include <stdlib.h>

int readNextLine(FILE *rfxFile,char *dllFile, char *inFile);

extern "C" void Disp(char *text);
extern "C" int checkPaused();
extern "C" int checkRunAborted();
extern "C" void DispProgress(int n, int Nmax, int initFrom);

// declare extern to remote procedure here.
extern "C" int RainfallProcess(char *inFile);
typedef int (*RAINFALLPROCESS)(char *inFile);
```

```
NAME RainfallTest.exe
EXPORTS
Disp
DispProgress
checkRunAborted
checkPaused
```