THE GENERAL SOLUTION FOR ESTIMATING ULTRASONICALLY INDUCED TISSUE HEATING

BY

DONOVAN SCOTT ELLIS

B.S., University of Illinois, 1991

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1992

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

MAY 1992

WE HEREBY RECOMMEND THAT THE THESIS BY

DONOVAN SCOTT ELLIS

ENTITLED THE GENERAL SOLUTION FOR ESTIMATING ULTRASONICALLY INDUCED TISSUE HEATING

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

W O'Brien
Director of Thesis Research

N. Narayana Rao
Head of Department

Committee on Final Examination†

Chairperson

† Required for doctor's degree but not for master's.

O-517

DEDICATION

This thesis is dedicated to my family. Their continuing encouragement and support have allowed me to achieve my goals in school as well as in life.

ACKNOWLEDGMENTS

I would like to thank my research advisor, Professor William D. O'Brien, Jr., for his encouragement, support, and advice throughout my college career. Special thanks are given to Eric Chen, Ilmar Hein, Linda Nostwick, and Nadine Smith for their advice and friendship. Thanks are given to Bob Cicone for his assistance with the preparation of the figures of this thesis and his continuing advice and support. Also, I must thank Kathy Bates and Wanda Elliott for their daily support and encouragement. The Beckman Institute System Service Staff is also thanked for their patience and understanding during the preparation of this thesis. Finally I would like to thank the rest of my research group and the many others who have made an impact on this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The study of tissue heating, or hyperthermia, and possible damaging effects due to diagnostic ultrasound is very important. When tissues are exposed to ultrasound, a portion of the acoustic energy is absorbed. The energy absorbed by the tissue is converted into thermal energy which results in a temperature increase in the tissue. A significant increase in the temperature above ambient levels can cause damage to the tissue. The fact that bioeffects could be due to both thermal and nonthermal mechanisms is recognized, but the consensus is that a majority of the observed bioeffects that have been reported are a result of the temperature increase in the tissues [1].

There are two significant applications of the study of temperature increases in tissues exposed to ultrasound, the first being that of fetal imaging. For over three decades, diagnostic ultrasound imaging has been used to examine noninvasively the human fetus [2]. The exposure of the fetus to ultrasound power level used in the clinical setting is assumed to be safe. The American Institute of Ultrasound in Medicine (AIUM) has published guidelines for the level of ultrasonic energy, based upon the temperature increase in the tissue, that are considered the safe limit for use on the human fetus [3]. However, due to the widespread use of ultrasound and the development of new imaging instrumentation and techniques, such as Doppler ultrasound, studies must be continued to better understand the process of tissue heating and to guarantee the safety of exposure levels. The second application of tissue heating is that of cancer treatment. While the possibility of damage to fetal tissue depends on the limiting of tissue heating, in cancer treatment, by substantially increasing the temperature in a tumor, studies have shown that the cancerous tissues are destroyed. While this treatment is relatively new, the potential for the noninvasive treatment of tumors via ultrasound is possible.

In order to determine the temperature increase produced by diagnostic ultrasound equipment, several methods have been published. However, these methods have been restricted to simplified models of ultrasound propagation in tissue. In addition, the tissue models are simplified versions of the actual tissues encountered by the ultrasound beam.

This thesis presents a new method for computing the temperature increase in tissue due to diagnostic ultrasound equipment. The method is unique in that it is applicable to all transducer geometries as well as tissue models. Chapter 2 presents background information on ultrasound, and the thermal mechanism of tissue heating due to ultrasound. In addition, the bioeffects due to tissue heating and a brief review of previously published methods of calculating the temperature increase in tissue are presented. Chapter 3 develops the theory of the general solution for a homogeneous medium. Chapter 4 presents the results of the general solution for both circular and rectangular apertures and compares the results for the circular aperture with those for the heated disc solution [3], a published method for estimating the temperature increase. Chapter 5 presents the theory and results of the application of the general solution to a layered tissue medium, typical of fetal imaging as well as imaging of small parts and the eye. Finally, Chapter 6 reviews the applicability of the general solution as well as summarizing the results obtained and the direction of future work.

CHAPTER 2

BIOLOGICAL EFFECTS OF ULTRASOUND

2.1 Principles of Ultrasound

Ultrasound is defined as sound that is not audible to human beings because its frequency is outside of and greater than the range detectable by the human ear. The following mathematical basis for waves is applicable to ultrasound as it is a wave phenomenon. A more detailed derivation can be found in [4].

The linearized wave equation, upon which the general solution is based, is given as

$$\nabla^2 p = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} \tag{2.1}$$

where ∇^2 is the second-order LaPlacian operator, c is the speed of sound in the medium, p is the acoustic pressure and t is time. This equation relates pressure with both the speed of sound in the medium and time. A harmonic plane wave is a function of only one spatial coordinate. For example, for a wave that propagates only along the x-axis, Equation (2.1) is reduced to

$$\frac{\partial^2 p}{\partial x^2} = \frac{1}{c^2}\frac{\partial p^2}{\partial t^2} \tag{2.2}$$

The solution to this differential equation is

$$p(x,t) = A_1 \exp[j(\omega t - kx)] + A_2 \exp[j(\omega t + kx)] \tag{2.3}$$

where ω is the angular frequency and k is the wave number ($=\omega/c$). The term with coefficient A_1 represents a wave travelling in the +x direction and the term with coefficient A_2 represents a wave travelling in the -x direction. Note that the pressure is a function of the two values, distance, x, and time, t. Figure 2.1 illustrates a plane wave travelling in the +x direction (i.e., $A_2 = 0$). Figure 2.1(a) shows the wave for a fixed time, t, while x varies. Figure 2.1(b) shows the wave for a fixed distance, x, while t is allowed to vary.

From Equation (2.3) and Figure 2.1 several parameters can be defined. The wavelength, λ, as shown in Figure 2.1(a), is the distance between two points is space for which the term $kx=2\pi$. Therefore, the wave number is related to the wavelength as

$$\lambda = \frac{2\pi}{k} \tag{2.4}$$

Similarly, for $\omega t = 2\pi$, the value T, the period of the wave, is defined as the value satisfying the periodicity of the wave, i.e.,

$$\omega = \frac{2\pi}{T} = 2\pi f \tag{2.5}$$

where f is defined as the frequency of the wave. The quantities, c, f, and λ are related as

$$c = f\lambda \tag{2.6}$$

Ultrasound is typically generated by converting electrical energy into acoustic energy using an ultrasonic transducer. Transducers are the basis for ultrasound wave generation and detection when used in imaging systems. Briefly, imaging systems operate by first emitting an ultrasound wave and then measuring the amount of energy reflected back to the transducer by the tissues being scanned. Through repeated emission, or pulses,

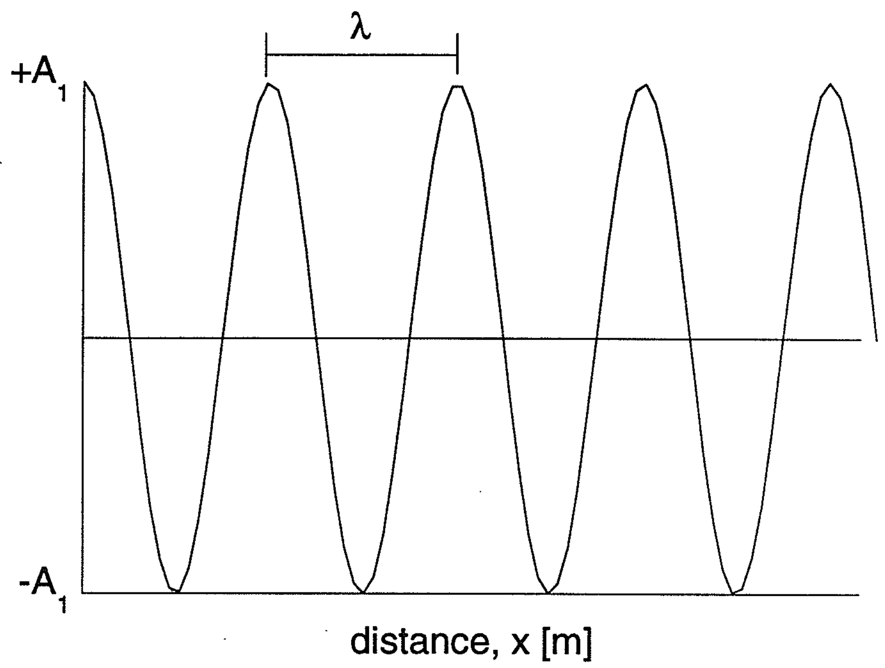

Figure 2.1(a):Pressure vs. distance, x, for a fixed time, t. λ is the wavelength of the wave.

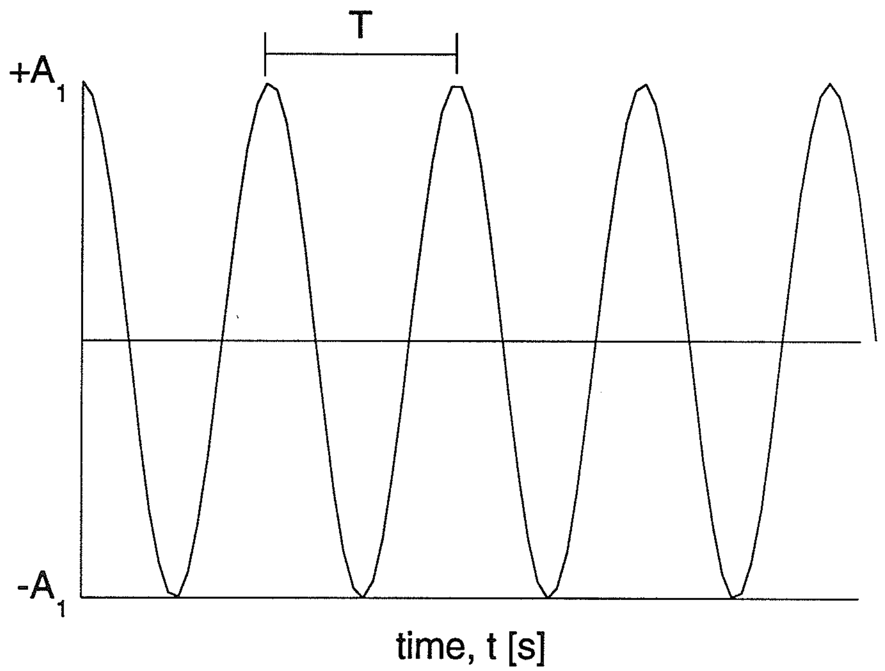

Figure 2.1(b): Pressure vs. time, t , for a fixed distance, x. T is the period of the wave.

of ultrasound signals in different directions, the data gathered can be converted into an image that can be used to determine the various structures present in the tissue.

A specific example of a transducer and the resulting ultrasound field is that of the plane-piston or unfocussed transducer. The plane-piston transducer consists of the circular layer of material vibrating with simple harmonic motion. Figure 2.2 shows the type of ultrasound field generated by the plane-piston transducer. As can be seen in the figure, the width of the beam is equal to the diameter of the circular disc in the near field. At a distance $D^2/4\lambda$, the beam begins to diverge. This distance, which defines the end of the near field and the beginning of the far field, is called the transition distance.

The plane-piston transducer is not of very much value in diagnostic imaging due to its large beam width. To improve resolution and to increase the intensity of the beam at a point of interest, focussed transducers are used. Figure 2.3 shows a beam profile similar to that of the plane-piston transducer. Here, the beam width narrows to a small distance in the focal region of the transducer. This concentration of energy results in a much larger intensity over a small region of interest. This allows for a better signal to be received by the

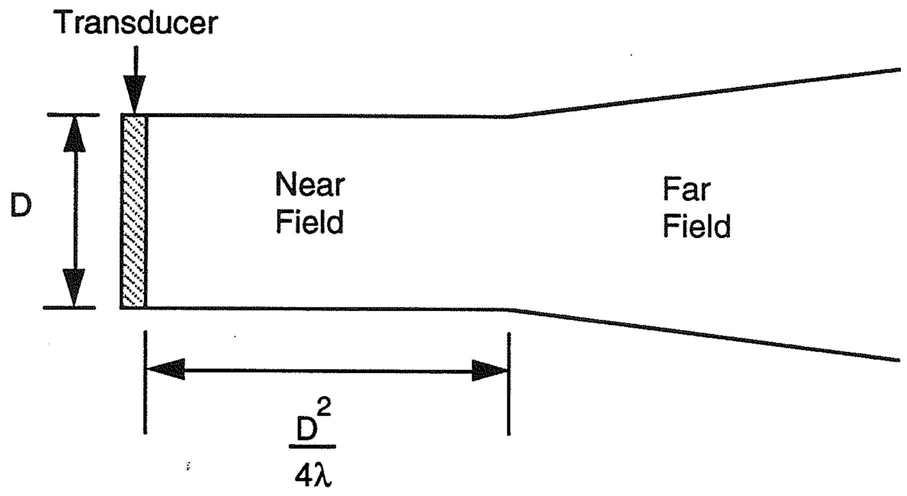

Figure 2.2: Ultrasound beam generated by a plane-piston transducer. D is the diameter of the transducer. The value $D^2/4\lambda$ is the transition distance [5].

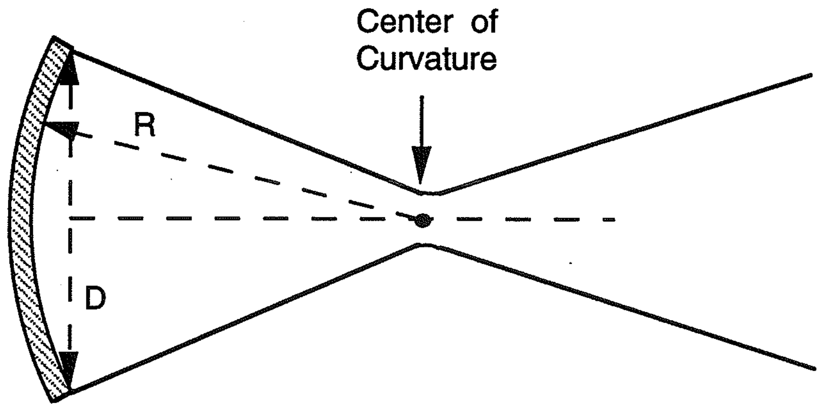

Figure 2.3: Ultrasound beam generated by a focussed transducer. D is the diameter of the transducer and R is the radius of curvature [5].

transducer when generating an image. Typical circular aperture focussed transducers have a diameter less than 2 cm, operate between 1 MHz and 5 MHz and have a f-number (=R/D) of between 3 and 5.

2.2 Absorption

Absorption is the conversion of acoustic energy into thermal energy. Typical tissues encountered by diagnostic ultrasound absorb the acoustic energy and convert it into heat. Mathematically, absorption can be defined as the amount of time required for the system to reach equilibrium after the application of a sudden change in pressure. In Equations (2.1)-(2.6), any change in pressure was instantaneous. The modified linear wave equation which accounts for this time lag is given as

$$\left(1 + \tau\frac{\partial}{\partial t}\right)\nabla^2 p = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} \qquad (2.7)$$

where τ is the relaxation time of the medium.

If the analysis is restricted to acoustic quantities that behave as $\exp(j\omega t)$, Equation (2.7) reduces to

$$(\nabla^2 + \mathbf{k}^2)\,\mathbf{p} = 0 \qquad (2.8)$$

where

$$\mathbf{k} = \frac{\omega}{c\sqrt{1 + j\omega\tau}} \qquad (2.9)$$

If **k** is defined as

$$\mathbf{k} = k - j\alpha \qquad (2.10)$$

and is substituted into Equation (2.9), Equations (2.11) and (2.12) are obtained by collecting the real and imaginary parts.

$$(k^2 - \alpha^2) + 2\omega\tau\alpha k = \left(\frac{\omega}{c}\right)^2 \qquad (2.11)$$

$$2\alpha k = \omega t\,(k^2 - \alpha^2) \qquad (2.12)$$

By solving this system of equations, k and α are given as

$$\alpha = \frac{\omega}{c}\frac{1}{\sqrt{2}}\left[\frac{\sqrt{1+(\omega\tau)^2} - 1}{1 + (\omega\tau)^2}\right] \qquad (2.13)$$

$$k = \frac{\omega}{c}\frac{1}{\sqrt{2}}\left[\frac{\sqrt{1+(\omega\tau)^2}+1}{1+(\omega\tau)^2}\right] \tag{2.14}$$

Rewriting Equation (2.3) with the new definition of **k** (Equation (2.10)), the pressure waveform in the +x direction is given as

$$p = P_0 \exp(-\alpha x)\exp[j(\omega t - kx)] \tag{2.15}$$

where P_0 is the peak pressure amplitude. Graphically, Equation (2.15) is presented in Figure 2.4. The solid represents the pressure as calculated by Equation (2.15). The dashed line represents the exp (-αx) term.

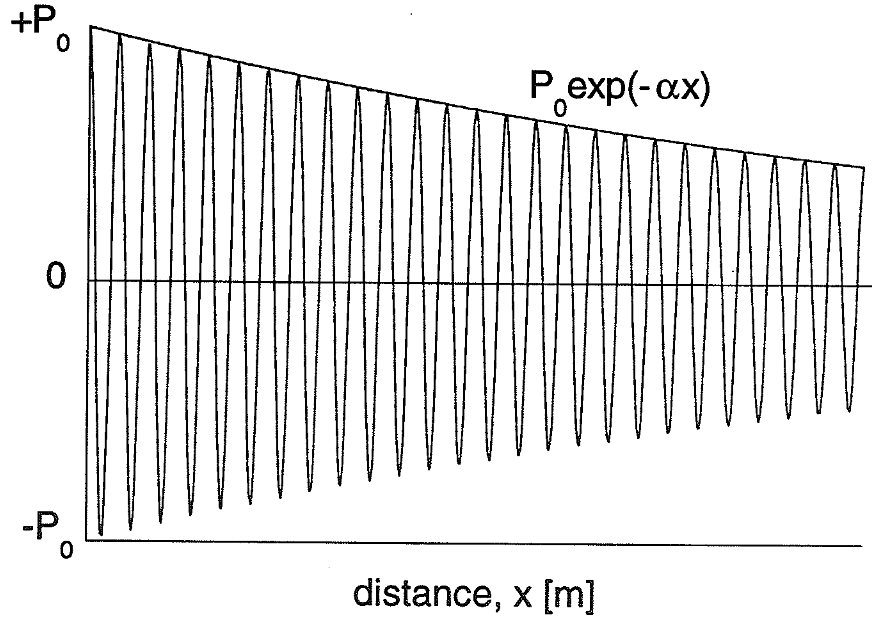

Figure 2.4: Pressure waveform, p, being attenuated at a rate of exp(-αx).

The process of absorption can be placed into three categories [4]. The first type of absorption is with viscous losses. Viscous losses are due to the interaction of adjacent portions of the tissue when subjected to movement created by the sound wave. This type of loss can be considered as a frictional loss. The absorption coefficient associated with this type of loss is

$$\alpha_S \approx \frac{2}{3}\frac{\omega^2}{\rho_0\, c^3}\eta \tag{2.16}$$

where ρ_0 is the density of the medium and η is the shear viscosity coefficient. Equation (2.13) is a low frequency approximation and is valid except for extremely high ultrasonic frequencies.

The second type of mechanism that produces absorption is heat conduction. Heat conduction losses result from the conduction of thermal energy from areas of higher temperature to areas of lower temperature. The thermal conduction absorption coefficient is

$$\alpha_K \approx \frac{\omega^2}{2\rho_0 c^3}(\gamma\text{-}1)\frac{\kappa}{C_P} \tag{2.17}$$

where γ is the ratio of heat capacities (C_P/C_V), C_P is the heat capacity at a constant pressure, C_V is the heat capacity at a constant volume and κ is the thermal conductivity of the fluid. Again, this equation is an approximation and is valid for low frequencies only.

The third process of absorption is due to molecular thermal relaxation. Molecular thermal relaxation accounts for the conversion of kinetic energy of molecules into stored potential energy, internal rotational energy, or energies of association and dissociation.

The total absorption coefficient of a medium is the summation of each of the individual absorption coefficients, that is,

$$\alpha = \sum_{i=1} \alpha_i \tag{2.18}$$

2.3 Thermal Mechanism

The amount of heat generated is dependent on the medium characteristics and the ultrasound beam. The time average rate at which heat is generated, q_v, is

$$q_v = 2\alpha I \tag{2.19}$$

where α is the absorption coefficient of the medium and I is the time average intensity. This equation is valid for a plane travelling wave [6]. As Equation (2.19) indicates, the rate of heat generation is directly proportional to the absorption coefficient of the medium as well as the time average intensity of the ultrasound beam. This dependence of q_v on α and I is significant when examining the types of tissues encountered with ultrasound. For example, a high intensity ultrasound beam in amniotic fluid, a low absorbing medium, will generate heat at a slower rate than if in bone, which absorbs 60-80% of the ultrasound beam's energy [7].

While the process of absorption acts to generate a temperature increase in tissue, there are natural mechanisms of tissues that counteract such an increase. One mechanism is perfusion, which is the removal of heat from tissue via blood flow. Because ultrasound generates a localized area of tissue heating, the blood flowing throughout the tissue is able to carry heat away from the insonified tissue. The rate of perfusion varies from tissue to tissue. Examples of highly perfused tissues are the kidney and heart. In these tissues, blood flows throughout the tissue at a relatively rapid rate. An example of a tissue with low perfusion is bone [7]. The second process of heat loss is heat conduction. Heat conduction is dependent on the thermal conductivity coefficient of the tissue, which is an indication of the rate that heat will travel from a point with a higher temperature to another point with a lower relative temperature.

As shown in the preceding example, bone is a low perfused, high absorbing medium. Thus, there is the potential for a larger temperature increase near a bone-tissue interface than in the homogeneous tissue case. Due to the fact that during ultrasound exams of the human fetus, fetal bone is being insonified, there is a great deal of research into the temperature increase at the tissue-bone interface [7,8,9,10]. This will be further discussed in Section 2.4.

In the mathematical modeling of the temperature increase generated by ultrasound, the bio-heat transfer equation is used to combine the processes of absorption, perfusion and heat conduction into one equation. The bio-heat transfer equation [11] is

$$\dot{T} = \kappa\nabla^2 T - T/\tau + q_v/c_v \qquad (2.20)$$

where $\dot{T}$ is the time rate of temperature rise, κ is the thermal diffusivity, T is the temperature increase above the ambient level, τ is the perfusion time constant, q_v is the rate of heat generation per unit volume, and c_v is the volume specific heat of the medium.

A solution to the bio-heat transfer equation (Equation (2.20)) developed by Nyborg [12] is used as the basis for the general solution. Equation (2.21) gives the temperature rise at a point a distance r from a small heat source of volume dv. The source has been generating heat at a rate $q_v dv$ (Equation (2.19)) for time t.

$$T(t) = \frac{C}{r}\{\, E[2 - \mathrm{erfc}\,(r^* - R)] + E^{-1}\,\mathrm{erfc}\,(t^*+R)\} \qquad (2.21)$$

where

$$C = q_v dv/8\pi K \qquad (2.22)$$

$$K = c_v\kappa \qquad (2.23)$$

$$E = \exp(-r/L) \qquad (2.24)$$

$$L = \sqrt{\kappa\tau} \qquad (2.25)$$

$$t^* = \sqrt{t/\tau} \qquad (2.26)$$

$$R = r/\sqrt{4\kappa t} \qquad (2.27)$$

c_v is the volume specific heat for the tissue and erfc is the complex error function.

As discussed earlier, perfusion is the process of heat loss via blood carrying heat away from the localized source of heat generation. As can be seen from Equation (2.24), the value L takes on the mathematical form of a time constant. The perfusion length, L, and the perfusion time constant, t, are related by Equation (2.25). An example of the calculation of the perfusion length, L, for moderate perfusion, $\tau = 1000$ s, is L = 11.8 mm. For vigorous perfusion, $\tau = 100$ s and L = 3.7 mm. These values are valid for the thermal diffusivity, κ, of water ($\kappa = 0.14$ mm^2/s) [3].

2.4 Bioeffects

The significance of tissue heating due to ultrasound is apparent when normal body temperature conditions are reviewed. The temperature in the human body is a near constant. The body regulates the internal temperature within a few degrees Celsius. This consistency is due to a complex mechanism which balances the internal generation of body heat loss [1]. This consistency is important for the body to function normally. Table 2.1 indicates the various classifications of body temperature, and shows the significance of a change in temperature of a few degrees Celsius.

Table 2.1 Human body temperature as related to fever [1].

Classification	Celsius	Fahrenheit
Hypothermia	less than 36.0	96.8
Normal	37.0	98.6
Mild Fever	38.5	101.3
Average Fever	39.5	103.1
High Fever	40.5	104.9
Severe Fever	more than 42.0	107.6

These changes in temperature are sometimes due to illness or increased physical activity. However, the upper limits are rarely approached except in the case of heat stroke, which is potentially fatal. These temperature increases can also be caused by exposures to ultrasound in a localized region. While the entire body is not affected, the localized area may be subjected to dangerous temperature increases.

Not only are the variations in temperature a concern, but also the duration of such variations. In research on cell activity and growth, it has been found that there is virtually no effect on human cells for temperatures below 40°C. However, when cells are exposed to a temperature of 42°C, bioeffects are apparent when the subject has been exposed for 120 minutes. At 46°C it only takes 7.5 minutes for bioeffects to be generated [13]. This relationship between temperature increase and length of exposure is very important. Researchers have used the fact that tissue damage occurs at higher temperatures as a treatment for destroying cancerous tumors without invasive surgery. By exposing tumors to higher temperatures above 45°C, but for very short durations, cancerous tissues were damaged [1]. While this treatment is advantageous in that it does not require surgery, the fact still remains that healthy tissue will also be exposed to these high temperatures and collateral damage may occur. Researchers have found this to be a problem as temperatures around 41°C show signs of enhancing cell growth and may also enhance *tumor* growth. On the other extreme, temperatures above 45°C begin to cause damage to the surrounding healthy tissue, as discussed earlier. Therefore, the treatment is limited to a very specific temperature range [14].

An example of the effect of temperature on the cell cycle rate is shown in Figure 2.5, where the cell cycle rate is plotted against growth temperature. As can be seen, the cycle rate reaches a maximum near 35°C. While the cell cycle rate is diminished as the temperature decreases from 35°C, there is still a substantial growth rate until 0°C (freezing) is reached. However, when progressing in the opposite direction, a small increase in growth results in a rapid decrease in the cell cycle rate. An example for mammalian cells is shown in Figure 2.6, where the growth of 5178Y cells is plotted for different temperatures. As can be seen from the graph, the growth rate is a maximum for 37°C. As the temperature decreases or increases from 37°C, the growth rate decreases.

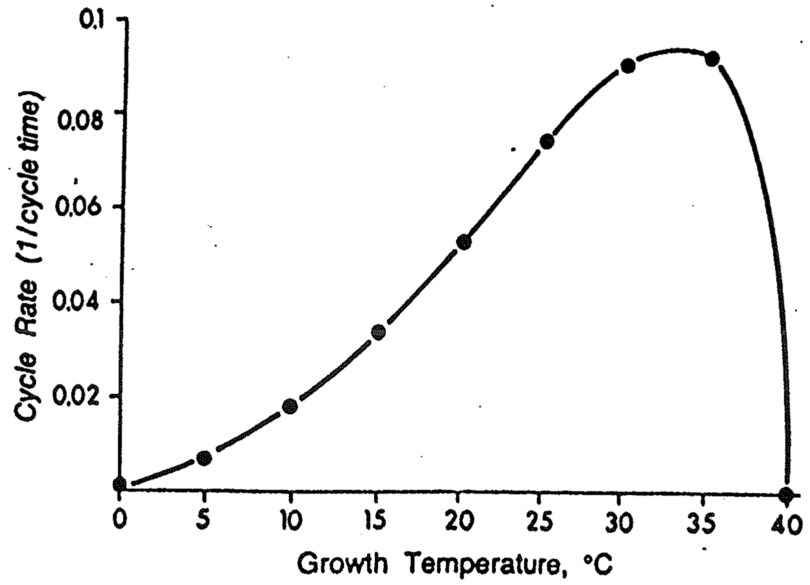

Figure 2.5: Cycle rate for onion cells for different growth temperatures. The maximum cycle rate occurs for temperatures between 30-35°C. There is no cell progression at 40°C [1].

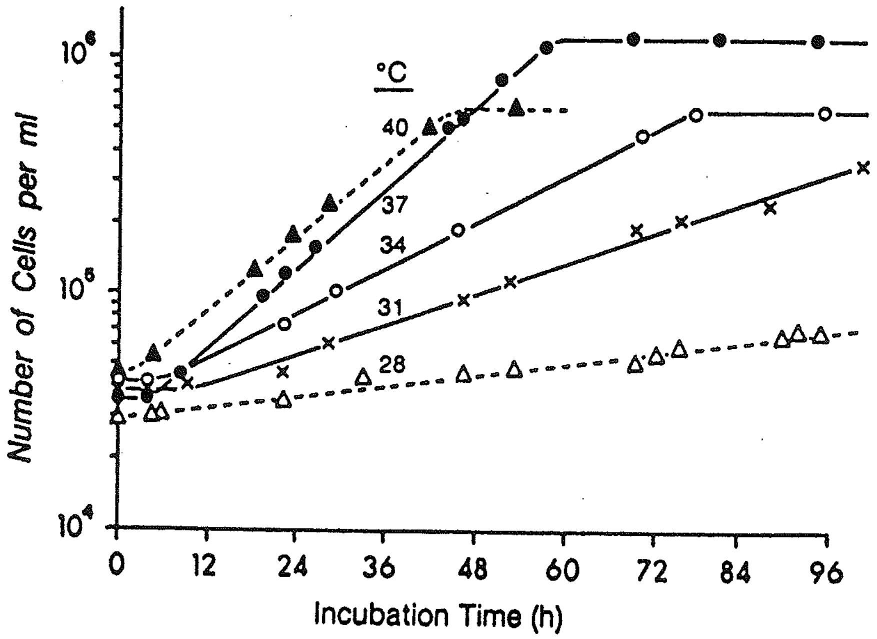

Figure 2.6: Growth curves for in vitro 5178Y cells at different temperatures [1].

This shows the fragile biological environment that exists for tissue. A small change in temperature, i.e., via ultrasound, may lead to significant cellular damage.

Miller and Ziskin [1] have reviewed data from various published research experiments which have claimed the observation of thermal bioeffects. The data were combined to create a composite view of the trends of observed thermal bioeffects. Figure 2.7 is a graphical representation of the collected data.

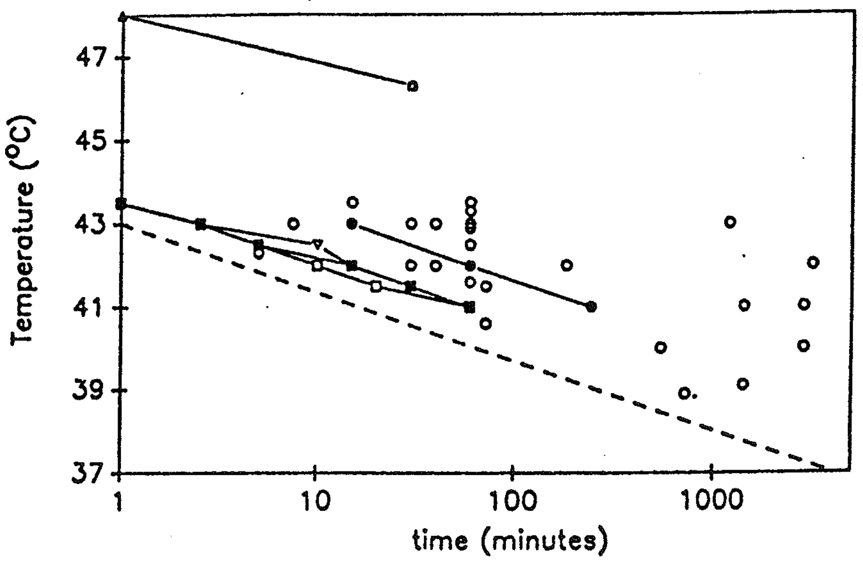

Figure 2.7: Graphical summary of reported thermal bioeffects for which the temperature increase and exposure durations have been reported. The solid lines represent data relating to a single effect. The dashed line (t_{43} =1) is the lower boundary for observed thermal biological effects [1].

The t_{43} (dashed line) represents the threshold level for which bioeffects have not been observed. Any combination of exposure duration and temperature elevation below the line indicates that there have been no observed thermal bioeffects.

The review by Miller and Ziskin [1] of bioeffects due to hyperthermia has collected all of the published data on bioeffects. From their review, the authors make the following conclusion with respect to hyperthermia and diagnostic ultrasound,

> A conservative estimation of a suggested relationship between fetal abnormalities and hyperthermia suggests that it appears reasonably certain that short exposures to an elevated temperature of 39°C would pose no adverse effects. At higher temperatures the duration of the exposure becomes important and must be considered in evaluating the probability of harm. Scientific evidence supports the conclusion that ultrasound exposure need not be withheld because of a concern for thermally mediated adverse effects if the combination of temperature elevation and exposure duration are below the t_{43} boundary line... [threshold level] [1]

This report indicates that an even higher level than the AIUM 1°C limit might be acceptable, but that it is still very important to take into account the duration of the exposure as well as the makeup of the exposed tissue.

The most important cases being studied involve tissues that would greatly affect the ability of the patient to survive. In this case, the study of heating of the skull is studied. While most equipment in use today can generate elevations in tissue above 1°C, the potential for larger heating at bone interfaces is possible. The temperature increase at a bone-tissue junction is larger due to the larger absorption coefficient of bone [8]. This temperature increase could be larger than the accepted 1°C limit recommended by the AIUM Bioeffects Committee [3]. In order to simulate the exposure of humans to ultrasound, rats were used in the experiment. For the simulation of fetal tissue, mice were used.

In the study, the researchers determined the absorption coefficient of the skull and used these data to model the temperature increase generated by equipment used on humans. At the time of the report, the largest power being used in human exposures was approximately 500 mW [15]. The theoretical temperature increase model indicates the potential for a 5°C rise for a diameter of not less than approximately 1.0 cm [8]. This increase is substantially higher that the AIUM limit.

While the case of tissue heating in adult humans can easily be detected through a pain indication by the patient, the fetus has no method of communication. In the case of the mice experiments, equilibrium temperature increases in the adult mice were 5.6 ± 0.3 °C.

This value is again substantially higher than the prescribed limit. The conclusion of the paper simply points out:

> It is emphasized that this is an unlikely combination of events but the possibility of its occurrence, in principle, emphasizes the need for physicians to be aware of the output levels used by diagnostic ultrasound equipment and to be able to make rational judgments of the probability of excess heating as they use these devices [8].

In this case, the possibility of the temperature increase exceeding 1°C is clearly possible. If all of the evidence being presented is accurate, there is a serious possibility of physiological bioeffects that are occuring and have not or can not be detected.

Another interesting study on the absorption of ultrasound and the corresponding temperature increase in fetal bone was published by Drewniak et al. [9]. In this study, the temperature increase was measured for fetal femur bones at various stages in the gestation process. As the fetus develops, the soft tissue begins to harden to create the skeleton bone structure. Because bone has a higher absorption coefficient, the temperature effect at a later period in gestation is much more significant. As seen by the following graph (Figure 2.8), the amount of time required to the reach the AIUM 1°C limit decreases rather substantially between the second and third months of gestation. The graph indicates that after only a few seconds of exposure, the 1°C safety limit is reached. Again, the potential for bioeffects exists if the proper precautions are not taken.

2.5 Published Numerical Models

Over the past decade, methods for determining the theoretical temperature increase in tissue exposed to ultrasound have been developed [3,6,10,12,16,17,18,19,20]. These models have been used to model the temperature increase for several important applications and cases. They generally fall into two categories. The first category is that of tissue heating during fetal imaging. In this case, the goal is to estimate the maximum temperature

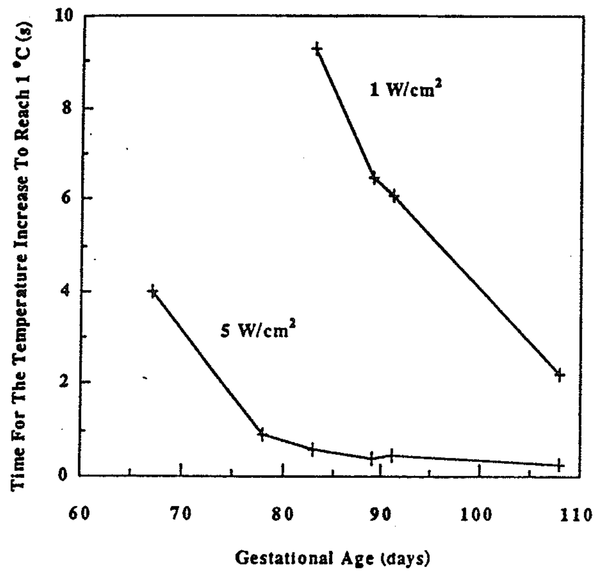

Figure 2.8: Exposure durations required for the temperature increase to exceed the 1°C AIUM safety limit versus gestational age [9].

increase that fetal tissue will be exposed to in order to set safety guidelines for diagnostic ultrasound equipment. In the second application of modeling, the estimation of temperature increases generated by ultrasound for cancer treatment via hyperthermia is studied. In this application the goal of the temperature increase is to generate a very small region of intense heating in order to destroy cancerous cells. By studying the heating pattern, the optimal treatment can be determined.

CHAPTER 3

GENERAL SOLUTION

This chapter presents a mathematical procedure that can be used to calculate the intensity field distribution and the temperature increase in tissues exposed to ultrasound. This method is termed the *general solution* as it can be applied to any transducer geometry as well as to any mathematically modeled tissue medium. The general solution used herein is a more exact and more computationally intensive method for estimating temporal averaged intensity fields and axial temperature rise.

3.1. Transducer Modeling

The calculation of the intensity field distribution is dependent on both the transducer parameters as well as on the mathematical description of the tissue being insonified. The transducer is modeled as a set of acoustic point sources evenly distributed on the transducer's surface. There are two prevalent diagnostic ultrasound transducer geometries in use today. The first is the circular aperture focussed transducer, which can be modeled as a spherical segment with a radius equal to the radius of curvature (ROC) value, as shown in Figure 3.1. The transducer and the medium are mapped in Cartesian coordinates with the origin located at the centroid of the spherical segment. The z-axis (range) is perpendicular to the aperture with z = 0 defined on the transducer surface. The mathematical representation of this surface is defined as

$$x_t^2 + y_t^2 + (z_t - \mathrm{ROC})^2 = \mathrm{ROC}^2, \quad |x_t^2 + y_t^2| \le \frac{D^2}{4} \tag{3.1}$$

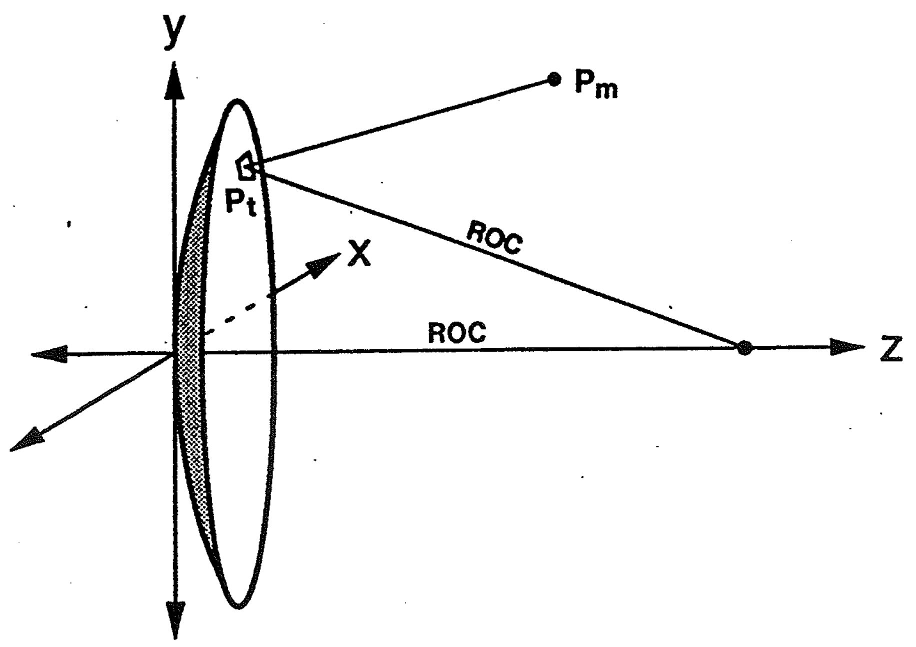

Figure 3.1: Geometrical model of a focussed circular aperture. The coordinates on the surface of the source, P_t, are (x_t,y_t,z_t) and the coordinates in the medium, P_m, are (x_m,y_m,z_m). ROC denotes the radius of curvature and the beam axis is the z-axis.

where (x_t,y_t,z_t) are the point source coordinates in Cartesian coordinates on the transducer surface, D is the diameter of the transducer, and ROC is the radius of curvature. Similarly, the rectangular focussed transducer aperture can be modeled as a cylindrical segment (Figure 3.2) with the origin located at the centroid of the aperture. This geometry modeled the focusing only in one dimension. The aperture is focussed in the image plane (x-z plane) and unfocussed in the elevated plane (y-z plane). This surface is mathematically described as

$$x_t^2 + (z_t - ROC)^2 = ROC^2, \quad |x_t| \leq \frac{x_{width}}{2}, \ |y_t| \leq \frac{y_{width}}{2} \tag{3.2}$$

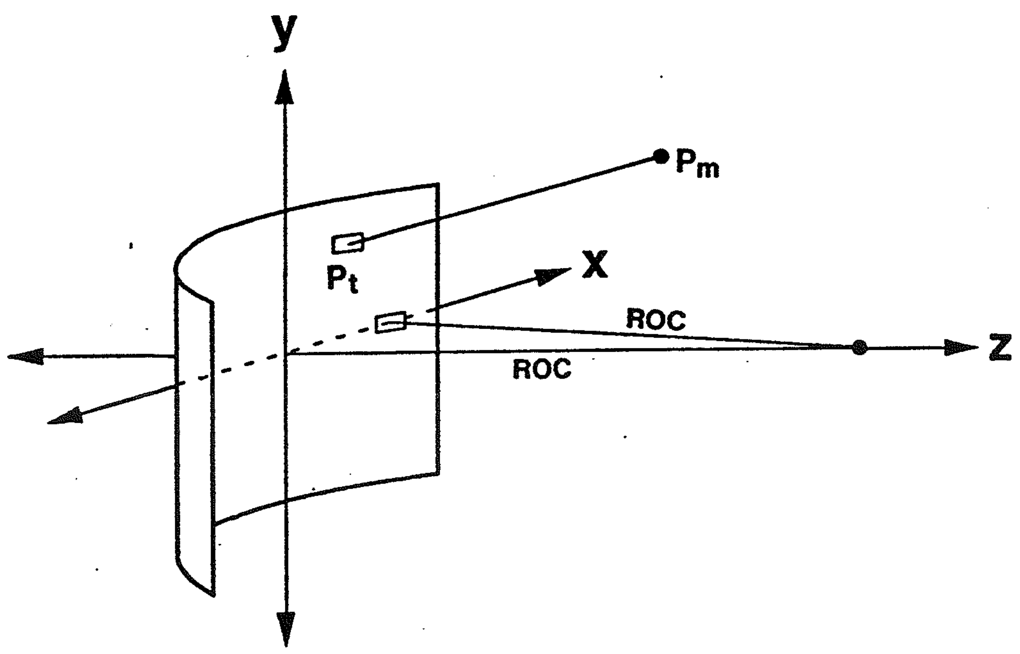

Figure 3.2: Geometrical model of a focussed rectangular aperture. The coordinates on the surface of the source, P_t, are (x_t,y_t,z_t) and the coordinates in the medium, P_m, are (x_m,y_m,z_m). ROC denotes the radius of curvature and the beam axis is the z-axis.

where x_{width} is the width of the transducer in the imaging plane and y_{width} is the width of the transducer in the elevational direction. The aspect ratio of a rectangular aperture is defined as

$$\text{aspect ratio} = \frac{x_{width}}{y_{width}} \tag{3.3}$$

3.2 Intensity Field Distribution

The relative pressure at any point, $P_m(x_m,y_m,z_m)$ in the medium can be determined by calculating the pressure contribution from each point source on the transducer's surface, that is,

$$p_{rel}(x_m,y_m,z_m) = \sum \frac{\exp\left[-(A+jk)|\mathbf{P}_m-\mathbf{P}_t|\right]}{|\mathbf{P}_m-\mathbf{P}_t|} \tag{3.4}$$

where

$$|\mathbf{P}_m-\mathbf{P}_t| = \sqrt{(x_m-x_t)^2 + (y_m-y_t)^2 + (z_m-z_t)^2} \tag{3.5}$$

where (x_t,y_t,z_t) is the transducer surface coordinate, $\mathbf{P}_m$ is a vector from the origin to P_m, $\mathbf{P}_t$ is a vector from the origin to P_t, A is the attenuation coefficient, k is the wave number $(= 2\pi/\lambda)$, λ is the acoustic wavelength (= c/f), c is the propagation speed and f is the ultrasonic frequency.

After the contributions from all of the transducer surface point sources are calculated, the complex relative pressure is converted into a relative intensity value using

$$I_{rel}(x_m,y_m,z_m) = p_{rel}(x_m,y_m,z_m)\, p_{rel}^*(x_m,y_m,z_m) \tag{3.6}$$

where $p_{rel}^*(x_m,y_m,z_m)$ denotes the complex conjugate of $p_{rel}(x_m,y_m,z_m)$. Thus, the relative intensity at any point in the medium can be obtained by providing the coordinates of the point in the medium and the geometric model of the transducer (Section 3.1).

These relative intensity values are then scaled to reflect the source power W_0 (at $z = 0$). The power at an axial distance z from the face of the transducer is given as

$$W(z) = W_0 \exp(-2Az) \tag{3.7}$$

The scaling of the relative intensity data is achieved by calculating the power for a fixed plane, perpendicular to the beam axis, which is the summation of the $I_{rel}(x_m,y_m,z_m)$ values over a plane perpendicular to the range (z) axis at range z and compared to the known power, W(z) (Equation (3.7)). The scaling factor is given as

$$G(z) = \frac{W(z)}{\sum_{xy} I_{rel}(x_m,y_m,z)\Delta x\Delta y} \tag{3.8}$$

where Δx is the lateral spacing in the x-direction and Δy is the lateral spacing in the y-direction. Thus, the actual intensity value at range z in the medium for the source power W_0 is given by

$$I(x_m,y_m,z) = G(z)\, I_{rel}(x_m,y_m,z) \tag{3.9}$$

3.3. Beam Width and Spatial Average Temporal Average Intensity

For the intensity distribution in each plane perpendicular to the beam axis, the maximum intensity value is determined. These intensity data are then scanned from the edge of the field toward the beam axis. The term "beam width" does not have an exact definition from application to application. The 10 dB beam width is determined from the first point encountered from the edge of the field that is 1/10 of the maximum intensity value of that plane of data. Thus, the beam width is defined as twice the distance from this point to the beam axis.

The spatial average temporal average intensity, or $I_{SATA}(z)$, is computed as

$$I_{SATA}(z) = \frac{W_{CS}(z)}{A_{CS}(z)} \tag{3.10}$$

where $W_{CS}(z)$ is the ultrasonic power within a given cross-section and $A_{CS}(z)$ is the area of that cross-section at each range z. For the circular aperture, the cross-sectional area is defined as a circle with diameter equal to the calculated 10 dB beam width.

For the rectangular aperture, two different cross-sectional areas are calculated. The first cross-sectional area is a rectangle with edge lengths equal to the 10 dB beam width in the x- and y-directions. The second cross-sectional area is that of an ellipse with lengths of the major and minor axes defined as the 10 dB beam width in the x- and y-directions, respectively.

3.4. Axial Temperature Increase

3.4.1 Heated disc method

In the AIUM Bioeffects Report [3], the heated disc method for calculating the axial temperature increase due to an ultrasonic beam was presented. The heated disc method models the ultrasound beam as a set of contiguous thin discs, each centered on the beam axis, with diameter equal to a beam width. The intensity of the ultrasound beam and, therefore, the intensity of each disc are modeled as being uniform over each disc and zero outside of the disc. The spatial average temporal average intensity of each disc is calculated using Equation (3.10) where the power at an axial distance z, W(z), is defined by Equation (3.7) and the cross-sectional area is defined as

$$A_{CS}(z) = \frac{\pi w^2(z)}{4} \tag{3.11}$$

where w(z) is the beam width at the axial distance z.

Each disc is then modeled as a heat source, or a heated disc. The total axial temperature increase is the superposition of the temperature increases generated by each heated disc. The steady-state temperature increase due to a single disc (Figure 3.3) is given by

$$\Delta T_{DISC,AXIAL}(z,z_0) = \frac{(2\alpha I_{SATA}(z_0))L\varepsilon}{2K}\left[\exp(-d/L) - \exp(-r^*/L)\right] \tag{3.12}$$

where $d = |z-z_0|$, $r^*=\sqrt{a^2+d^2}$, a is the radius of the disc, z_0 represents the axial location of the heated disc, α is the absorption coefficient, L is the perfusion length, ε is the thickness of the disc, and K is the thermal conductivity coefficient. Note that the temperature increase occurs on both sides of the heated disc. Thus, given the beam width, source power and medium characteristics (L, α, and K), the axial temperature increase can be calculated using

$$\Delta T_{AXIAL}(z) = \sum_{z_0=\varepsilon}^{\infty} \Delta T_{DISC,AXIAL}(z,z_0) \tag{3.13}$$

The summation represents the temperature contribution from every heated disc for a point on the beam axis, a distance z from the transducer.

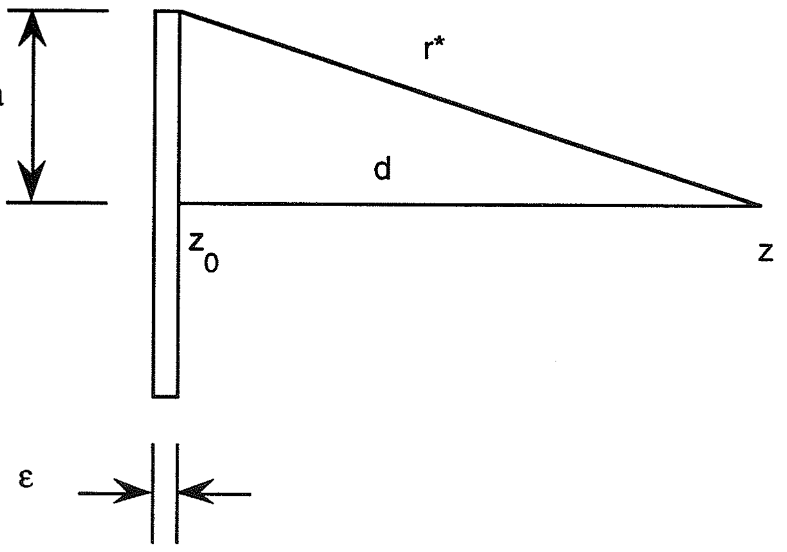

Figure 3.3: Geometry for calculating the temperature increase due to a heated disc [3].

3.4.2 General method

From the calculated intensity field distribution (Equation (3.9)), the axial temperature increase along the z-axis can be calculated for a general intensity field distribution. A point source solution to the bio-heat transfer equation was developed by Nyborg [12] (Section 2.3) and provided a mathematical description of the temperature increase due to a heated point source. The steady-state temperature increase at an observation point, $P_{obs}(x_{obs},y_{obs},z_{obs})$, due to a heated point source located at the point, $P_{hs}(x_{hs},y_{hs},z_{hs})$, is given by

$$\Delta T(P_{obs},P_{hs}) = \frac{2C}{r}\exp(-r/L) \tag{3.14}$$

$$C = \frac{q_v \Delta v}{8\pi K} = \frac{(2\alpha I)\Delta v}{8\pi K} \tag{3.15}$$

$$r = \sqrt{(x_{obs} - x_{hs})^2 + (y_{obs} - y_{hs})^2 + (z_{obs} - z_{hs})^2} \tag{3.16}$$

where ΔT is the temperature increase above the ambient level, r is the distance from the point source to the observation point, L is the perfusion length, Δv is the volume of the point source, K is the thermal conductivity coefficient, α is the absorption coefficient, and I is the ultrasonic intensity of the point source. (I is determined from Equation (3.9).)

The temperature along the z-axis is calculated by using the principle of superposition and summing the thermal contribution from every point source in the medium to the points along the z-axis. For a point on the z-axis, P_{obs} $(0,0,z_{obs})$, the steady-state temperature increase is

$$\Delta T_{AX}(z_{obs}) = \sum_{xyz} \Delta T(P_{hs}(x,y,z),P_{obs}(0,0,z_{obs})) \tag{3.17}$$

CHAPTER 4

APPLICATION OF THE GENERAL SOLUTION: CIRCULAR VERSUS RECTANGULAR APERTURES

4.1 Circular Aperture Case

Initially, the circular aperture focussed transducer case is computed to verify the general solution method. This is accomplished by comparing the results for the circular aperture focussed transducer with results using the heated disc method with the same transducer and medium parameters.

4.1.1 Results of the general solution

To compare the general solution approach to the heated disc approach [3], the same source geometry and homogeneous tissue properties are assumed (see Tables 4.1 and 4.2).

Table 4.1: Circular aperture focussed transducer properties.

f	frequency	3 MHz
W_0	source power	100 mW
D	source diameter	2 cm
ROC	radius of curvature	10 cm

Table 4.2: Homogeneous tissue properties.

c	propagation speed	1540 m/s
$A=\alpha$	attenuation = absorption	0.15 Np/cm
ρ	tissue density	0.001 kg/cm^3
L	perfusion length	1.18 cm
K	thermal conductivity	0.006 W/cm^2

For the circular aperture (Figure 3.1), the f number (ROC/diameter) is 5. Figure 4.1 shows a three-dimensional representation of the intensity field for this transducer, calculated with the general solution. Also, from the general solution, Figure 4.2 (solid line) shows the 10 dB beam width, Figure 4.3 (solid line) the spatial average temporal average intensity and Figure 4.4 (solid line) the temperature increase, each as a function of axial range.

4.1.2 Comparison with heated disc solution

A direct comparison of the general solution to the published heated disc model [3] for the identical set of source and medium parameters is shown in Figures 4.2, 4.3, and 4.4, where the dashed lines are the heated disk solutions [1]. In Figure 4.4, the temperature increase profiles for both models for the circular aperture are very similar in their maximum temperature increase and the axial distance at which the maxima occur. These two maximum temperature increases and their locations are essentially the same for the two models (see Table 4.3).

Table 4.3: Comparison of location and value of maximum temperature increase for the general solution and the heated disc method [3].

	ΔT_{max}	ΔT_{max} range
Heated disc model	0.492°C	1.05 cm
General solution	0.480°C	1.07 cm

However, near the focal region ($z \approx 6$ cm), the magnitude of the axial temperature increase is different for the two models. This difference is explained by examining the beam widths (Figure 4.2) for the two cases. The two beam widths are strikingly similar except for the "less steep" change of the general solution 10 dB beam width around the

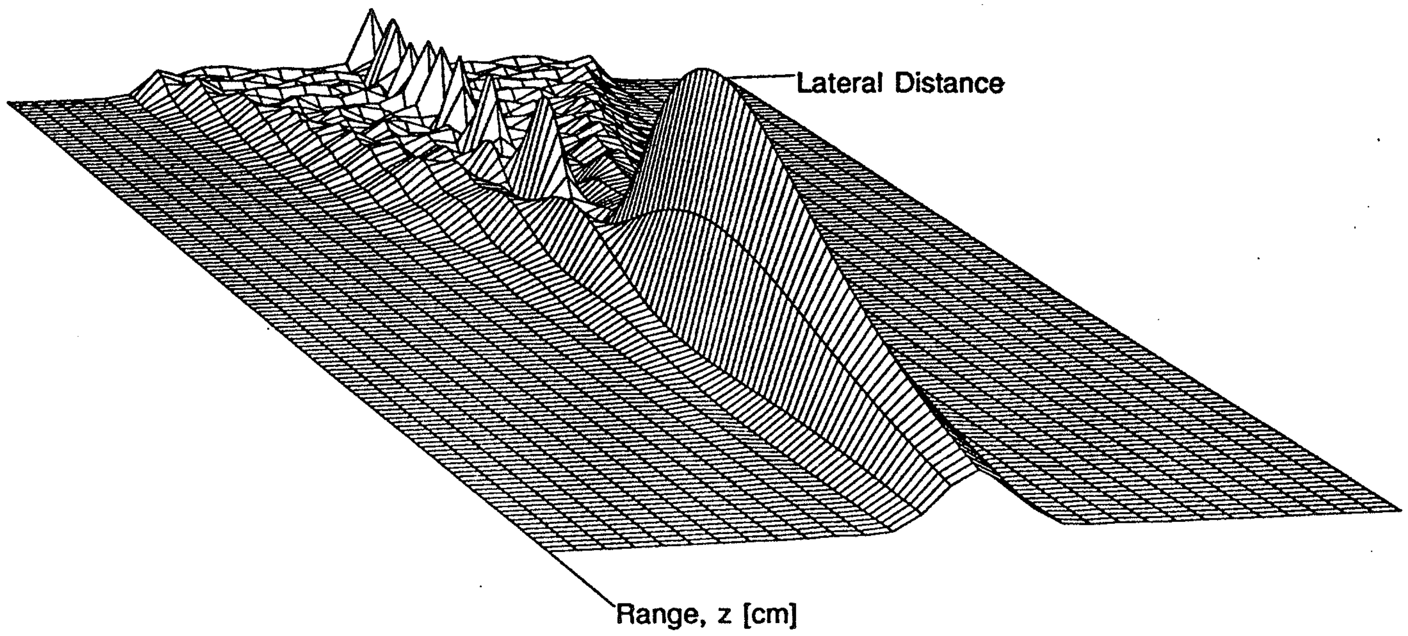

Figure 4.1. Three-dimensional plot of the intensity field distribution for the transducer and medium of Tables 4.1 and 4.2.

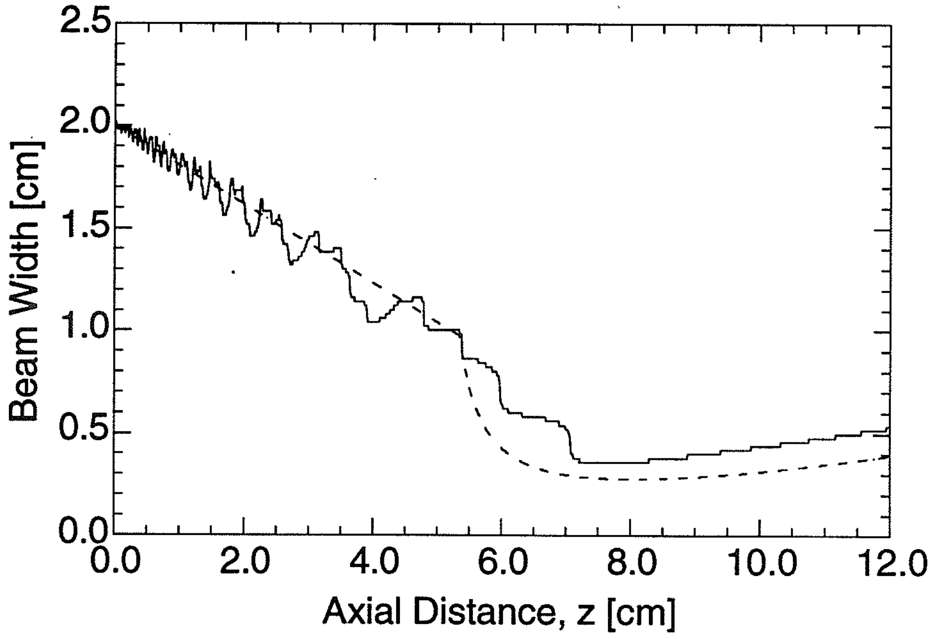

Figure 4.2: The 10 dB beam width calculated from the general solution (solid line) and the beam width used in [3] (dashed line).

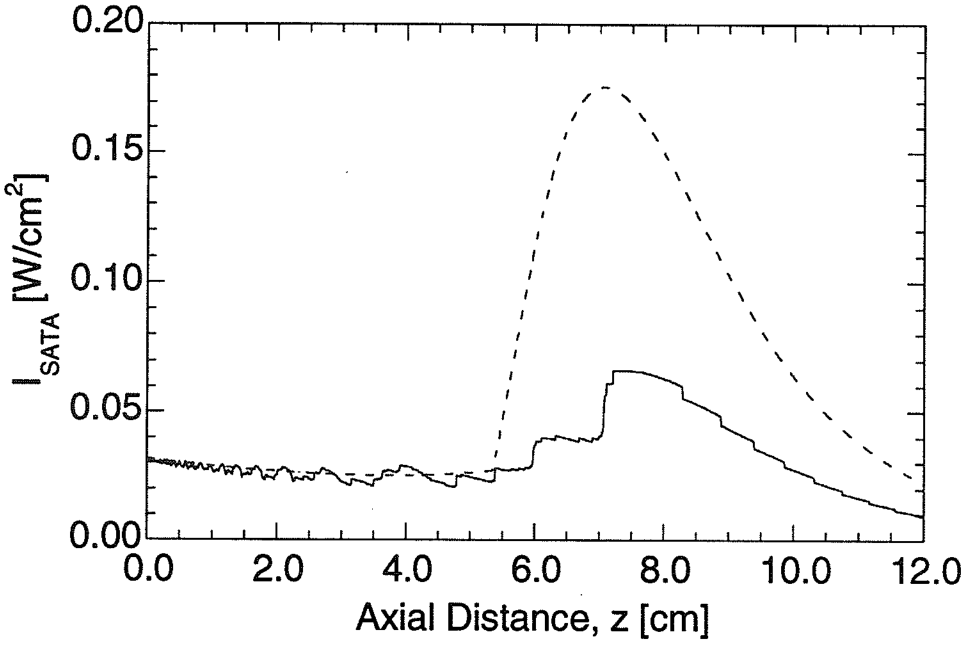

Figure 4.3: The spatial average temporal average intensity calculated from the general solution (solid line) and the spatial average temporal average intensity used in [3] (dashed line).

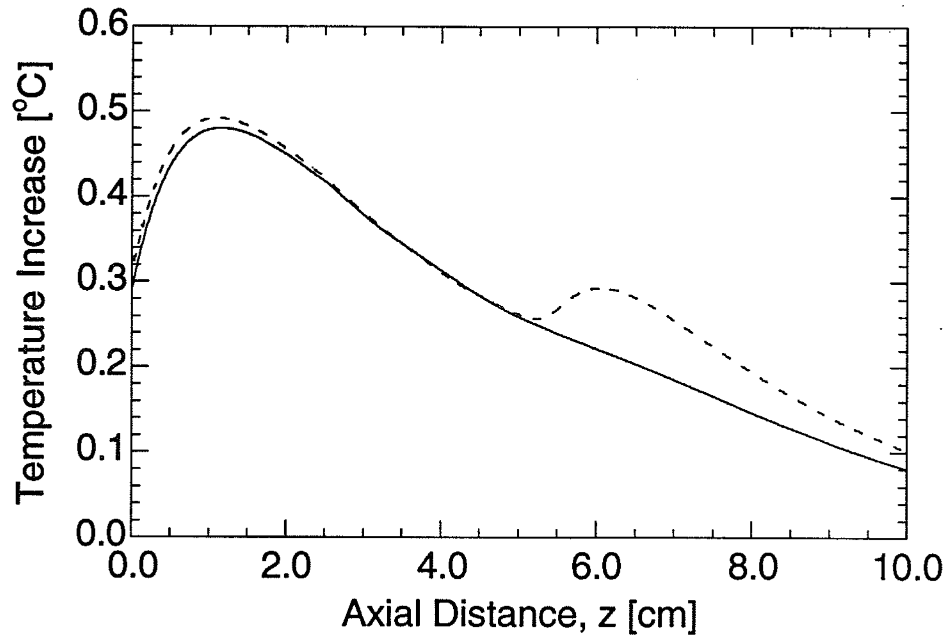

Figure 4.4: The temperature increase calculated from the general solution (solid line) and the temperature increase used in [3] (dashed line).

focal region. This beam width difference corresponds to a significant difference in the axial intensity profiles (Figure 4.3) for the two models. The large difference in the intensity values in the focal region indicates that the I_{SATA} maximum near the focal region for the heated disc model [3] results from the beam width modeling and not from the ultrasonic field.

To further demonstrate the hypothesis that the differences in the temperature increase profiles for the circular aperture (Figure 4.4) are due to the beam width differences, the calculated general solution 10 dB beam width is used with the heated disc model. Figure 4.5 shows the axial temperature increase profile calculated *with the heated disc model* for two beam widths, *viz.*, that used by [3] (dashed line) and that determined from the computed general solution of the ultrasonic field (solid line). The apparent "roughness" of the latter temperature rise profile (using the heated disc model with the

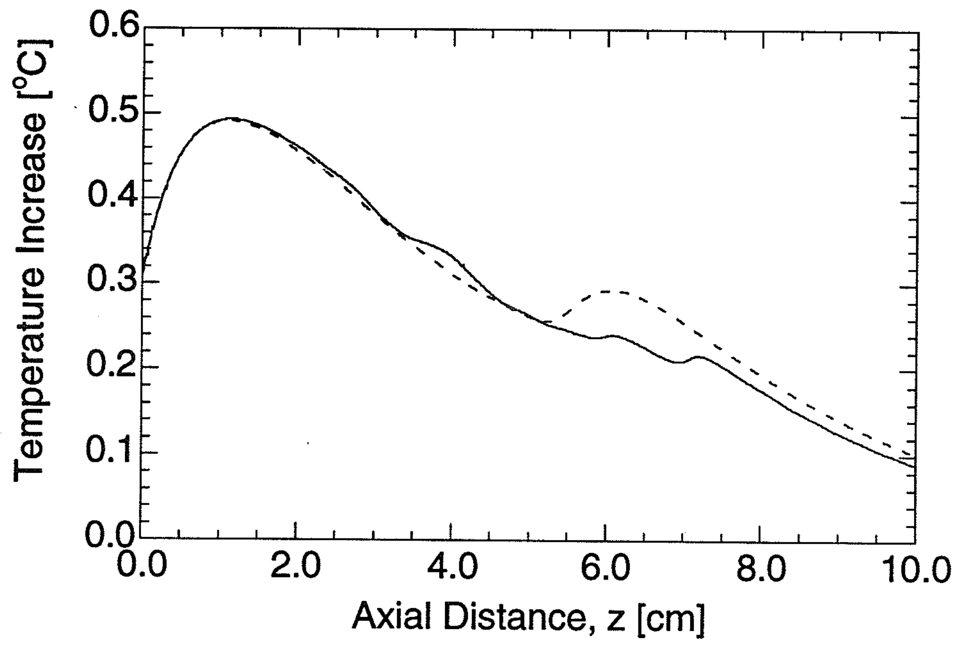

Figure 4.5: The temperature increase calculated with the heated disc model for two beam widths, *viz.*, the beam width used by [3] (dashed line) and the beam width determined from the computed general solution of the ultrasonic field (solid line).

10 dB computed beam width) is due to the apparent "roughness" of the 10 dB computed beam width shown in Figure 4.2 (solid line).

The difference in the temperature increase is explained by the definition of the beam width used in [3]. The width of the ultrasound beam was modeled based on a paper by Kossoff [21]. In this paper, the beam width in the focal region (>6.0 cm for this case) is defined as the distance between the first minima of the lateral ultrasound beam. This beam width is given in [21] as

$$w(z) = \frac{0.96}{K\sin\left(\frac{\pi T}{2Kz}\right)} \tag{4.1}$$

where

$$K = \frac{ROC}{ROC - z} \tag{4.2}$$

and T, the transition distance for an equivalent flat transducer is

$$T = \frac{D^2}{4\lambda} \tag{4.3}$$

As Figure 4.6 shows, the use of this definition of beam width for the heated disc method is inaccurate. The heated disc method assumes that ultrasonic power of the beam is contained entirely within the beam width. In addition, it is assumed that the intensity distribution over the disc is of uniform value. Figure 4.6 shows the lateral plot (z = 7.0 cm) of the intensity over a given beam width (0.30 cm from Equation (4.1)). The solid curve is that of the lateral intensity as calculated using the general method of Section 3.2. The solid rectangle represents the intensity as determined in the heated disc model as defined in

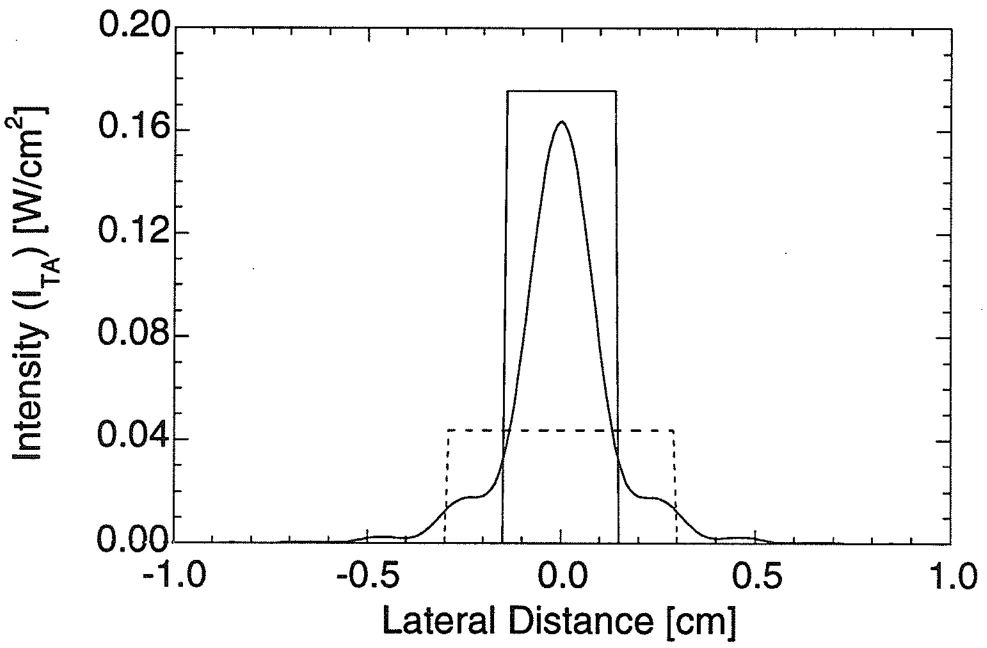

Figure 4.6: Comparison of the temporal average intensity used in [3] for the heated disc method and the temporal average intensity calculated by the general method of Section 3.2. The dashed rectangular box represents the temporal average intensity using the corrected beam width value (Equation (4.3)). (z = 7.0 cm)

Section 3.4.1 using Equation (4.1) for the beam width. Clearly, the intensity distribution is not uniform, nor is it contained entirely within the beam width defined by Kossoff [21] (Equation (4.1)). This leads to an overestimate of the I_{SATA} value used in the heated disc calculation as shown in Figure 4.3.

In reviewing the work presented by Kossoff [21] , it was noted that the equation for the beam width of a focussed transducer was incorrectly printed. Equation (9) of [21] should include the diameter, D, as a term in the numerator, that is,

$$w(z) = \frac{0.96D}{K\sin\left(\frac{\pi T}{2Kz}\right)} \tag{4.3}$$

The dashed rectangle of Figure 4.6 corresponds to the I_{SATA} using the corrected beam

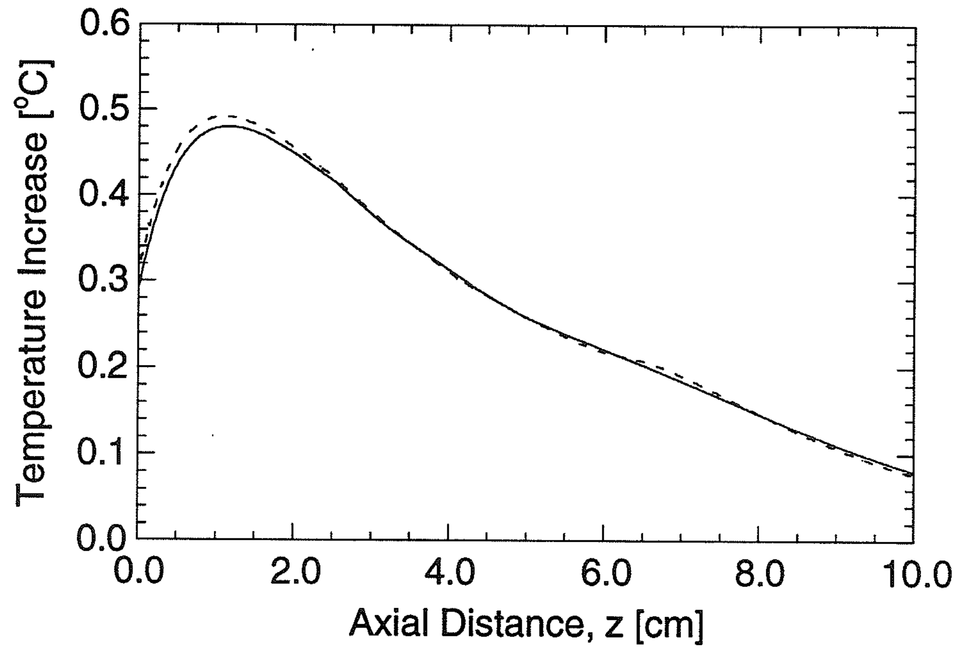

Figure 4.7: The temperature increase calculated from the general solution (solid line) and the temperature increase using the heated disc method [3] with the corrected beam width (dashed line).

width. The axial temperature increase using Equation (4.3) as the beam width in the focal region and the heated disc method is shown in Figure 4.7.

In summary, the maximum temperature increases for a circular aperture focussed transducer geometry are the same for both the heated disc and the general solution temperature rise models for the specified parameters. It has been shown that the discrepancy between the axial temperature increase calculated by the general solution and by the AIUM heated disc method [3] is due to the use of the incorrect beam width as defined in Kossoff [21].

4.2 Rectangular Aperture Case

The second class of results that is generated using the general solution are for rectangular aperture focussed transducers (Figure 3.2). The aperture parameters for the four cases studied are given in Table 4.4. The transducers are operating at a frequency of 3 MHz, and the same medium characteristics as those in Table 4.2 are used to model the medium.

Table 4.4: Aperture parameters for the rectangular apertures studied.

Aspect Ratio	x_{width}(cm)	y_{width}(cm)
1	1.7725	1.7725
π	3.14	1.00
2π	4.44	0.707
4π	6.28	0.50

The area of each aperture is π cm^2, the same as that for the circular aperture discussed in Section 4.1. Each transducer has a radius of curvature of 10 cm in the imaging plane. Thus, the f numbers for these four cases are 6.64, 3.18, 2.27 and 1.59 for the aspect ratios of 1, π, 2π, and 4π, respectively. Figures 4.8 and 4.9 show the 10 dB beam width in the imaging plane (x-z plane) and the elevational plane (y-z plane), respectively, for each of the four cases. Figures 4.10 and 4.11 show the spatial average temporal average intensity using a rectangular and elliptical cross-sectional area, respectively. Figure 4.12 shows the axial temperature increase for these four cases. The corresponding maximum temperature increases and their axial locations are given in Table 4.5.

Table 4.5 Temperature profile data for the rectangular aperture cases.

Aspect Ratio	ΔT_{max} (°C)	Axial Distance (cm)
1	0.447	1.0375
π	0.394	1.1150
2π	0.321	1.1625
4π	0.242	1.1250

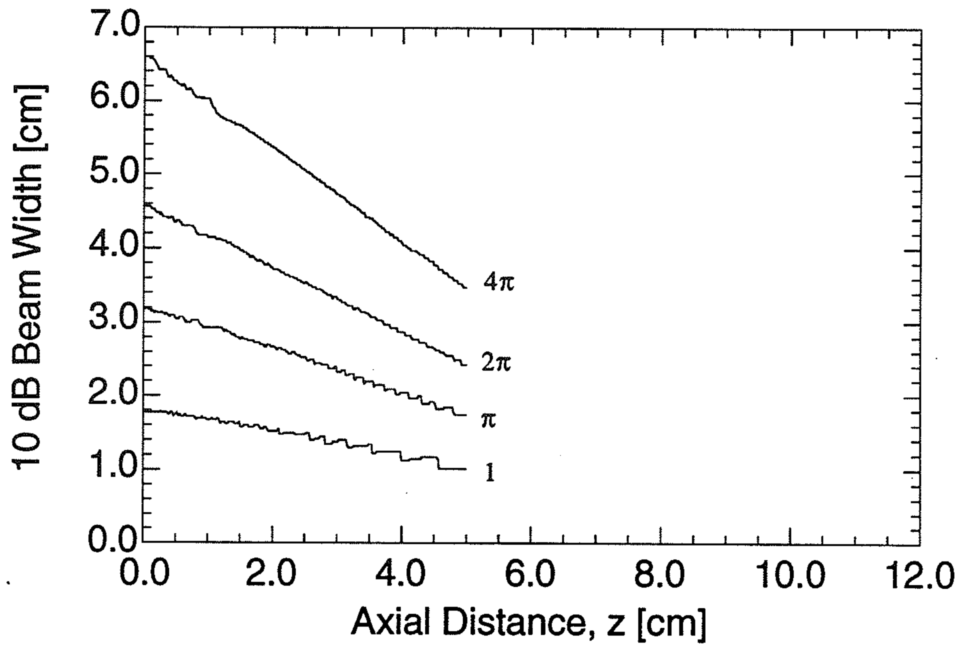

Figure 4.8: The 10 dB beam width in the imaging plane (x-z plane) calculated from the general solution for the rectangular aperture cases studied.

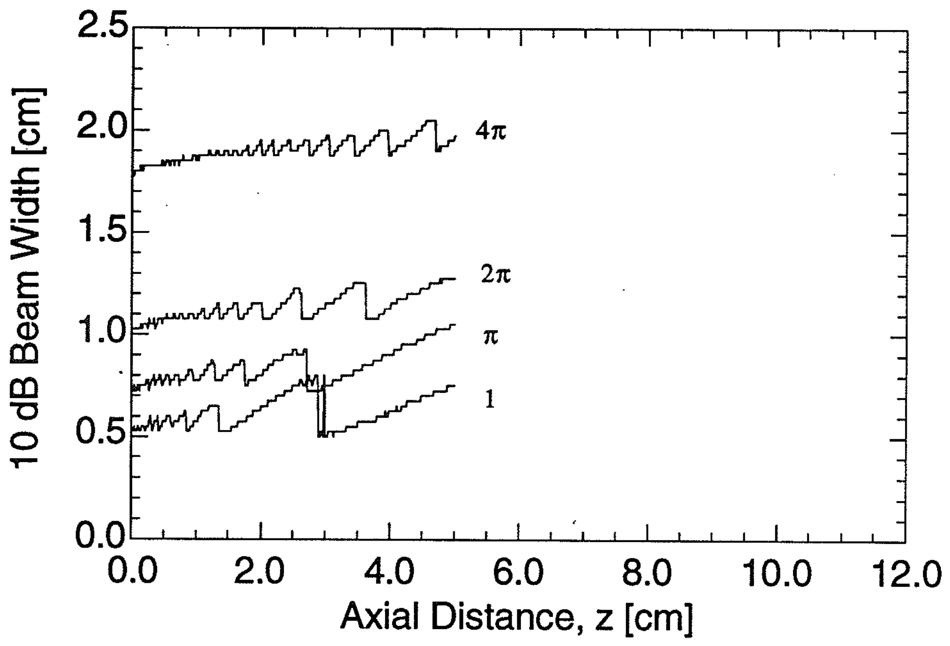

Figure 4.9: The 10 dB beam width in the elevational plane (y-z plane) calculated from the general solution for the rectangular aperture cases studied.

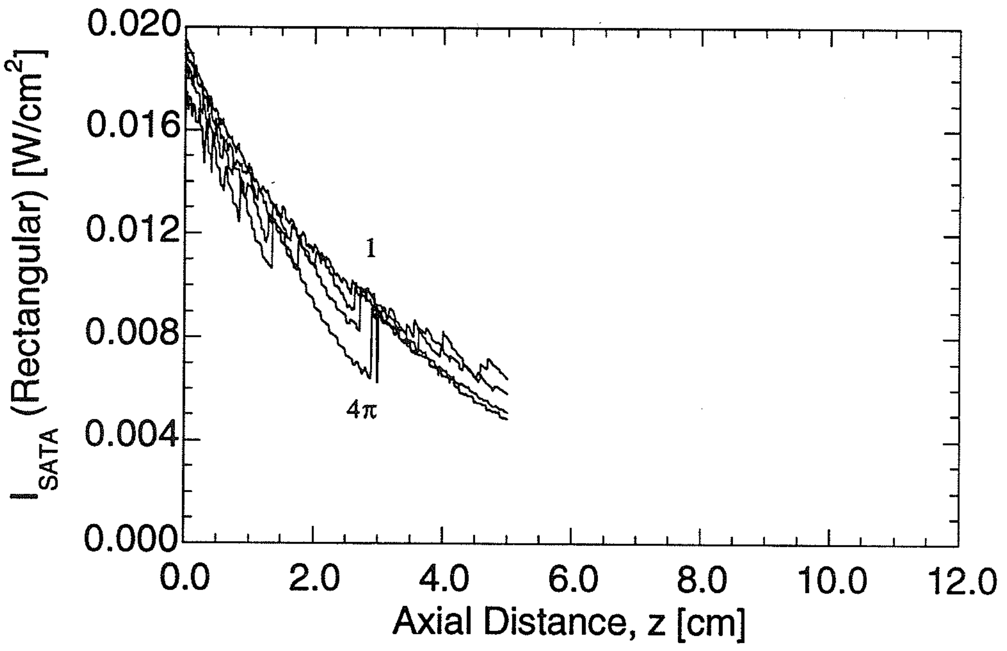

Figure 4.10: The spatial average temporal average intensity calculated from the general solution for the rectangular aperture cases studied. In this case the cross-sectional area is a rectangle with edge lengths equal to the 10 dB beam widths in the x- and y-directions.

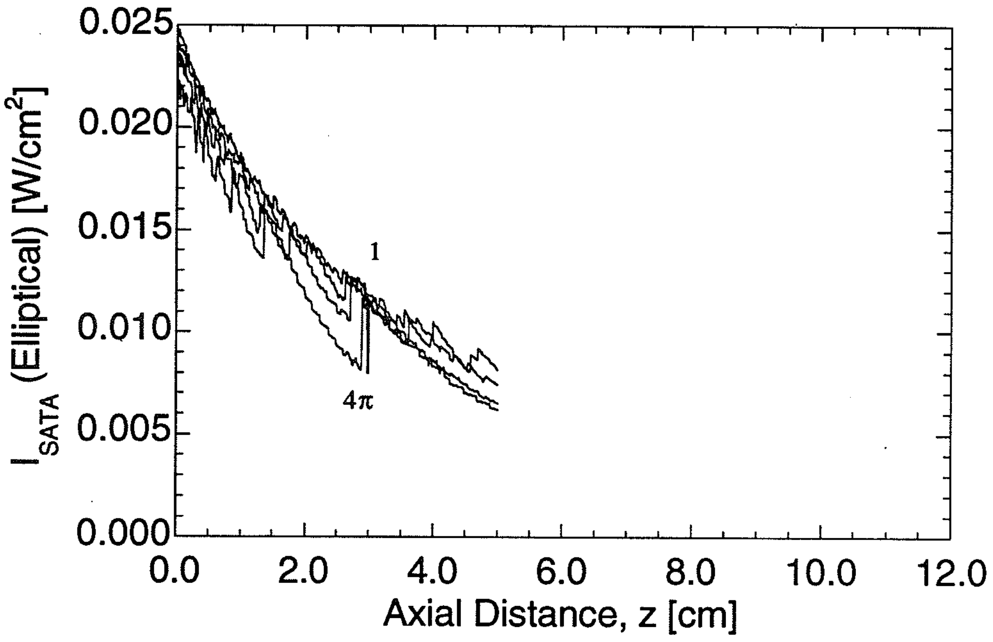

Figure 4.11: The spatial average temporal average intensity calculated from the general solution for the rectangular aperture cases studied. In this case the cross-sectional area is an ellipse with major and minor axis lengths equal to the 10 dB beam widths in the x- and y-directions, respectively.

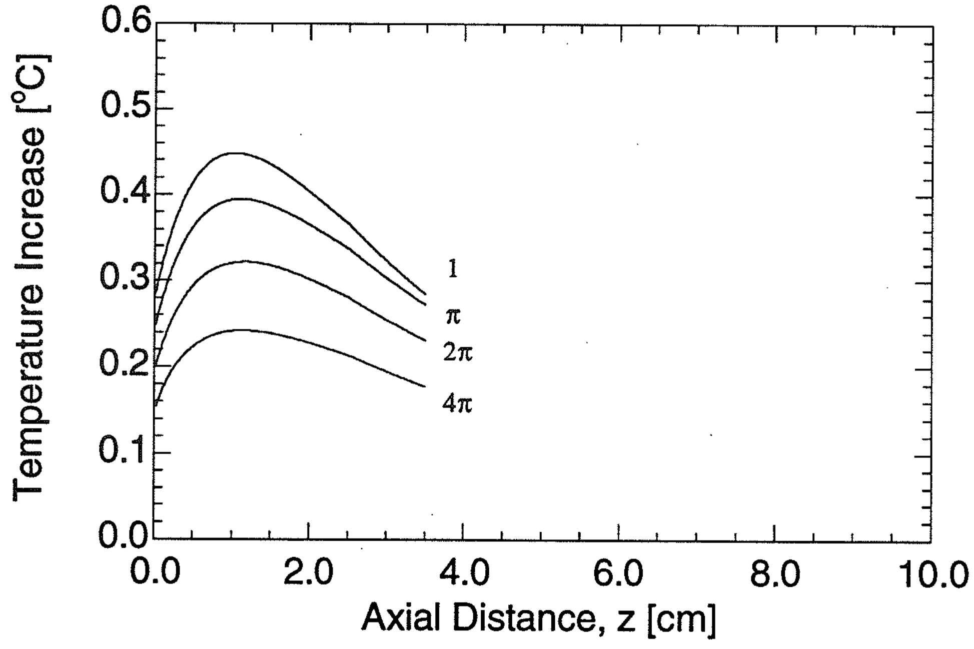

Figure 4.12: The temperature increase calculated from the general solution for the rectangular aperture cases studied.

4.3 Comparison of Circular Aperture Results with Rectangular Aperture Results

In determining the dimensions of the rectangular apertures, the surface areas of both the circular and rectangular transducers are the same for comparison purposes. In the case of the circular aperture, the surface area is π cm^2. For the rectangular apertures, the cross-sectional area is equal to the circular aperture surface area of π cm^2 for each aspect ratio.

The first comparison is between the circular aperture case and the rectangular aperture case with an aspect ratio of 1. The temperature profiles exhibit the same characteristics, namely, the singular maximum near 1.0 cm and the same general shape (Figure 4.13). The maximum temperature increase for the square aperture is slightly lower than for the circular aperture by 0.037°C or 8.2%. This small difference in the maximum temperature rise can be attributed to the difference in the distribution of the point sources in that the distance from the four square corners are subjected to greater attenuation effects and to the geometry differences wherein the square source is focussed only in the imaging plane. This is evident when examining the intensity (I_{SATA}) for the square aperture case as compared to that for the circular aperture (Figure 4.14). In Figure 4.14, the axial I_{SATA} is given for the square aperture for two different cross-sectional areas that were used to calculate the spatial average temporal average intensity, I_{SATA}, for the square aperture. The difference in the intensity and thus the temperature increase can also be explained by the fact that for the circular aperture focussing is occurring in two dimensions while for the rectangular aperture there is only focussing in one dimension.

The additional observation is how the aspect ratio relates to the maximum temperature increase. As the aspect ratio increases, the distance from the corners of the aperture is further increased. Thus, due to the attenuation effect, the resulting maximum axial temperature rise decreases as the aspect ratio increases. This trend is shown by the temperature profiles in Figure 4.12.

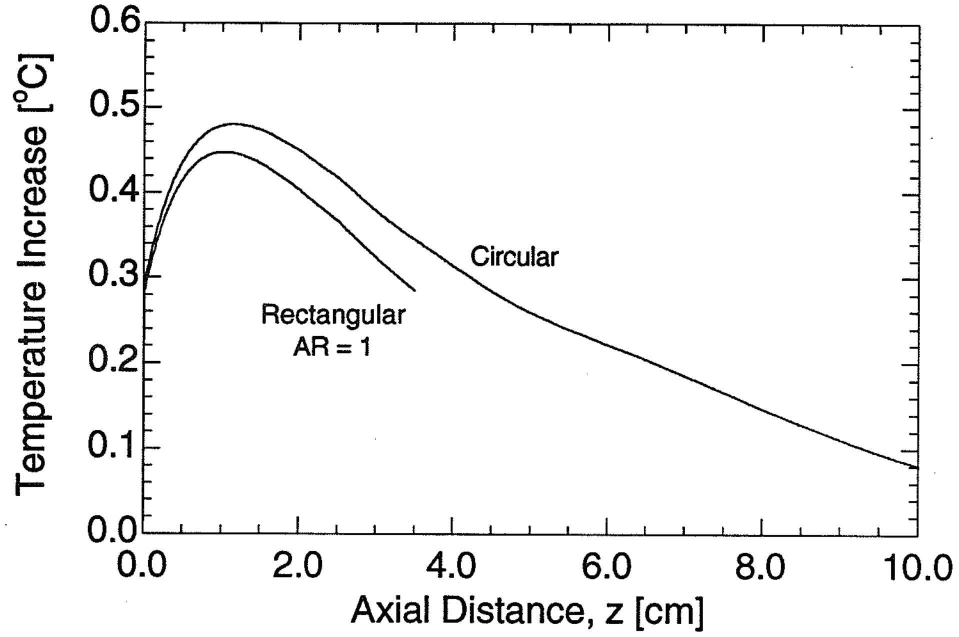

Figure 4.13: Comparison of the axial temperature increase between the circular aperture of Section 4.1 and the rectangular aperture with an aspect ratio of 1.

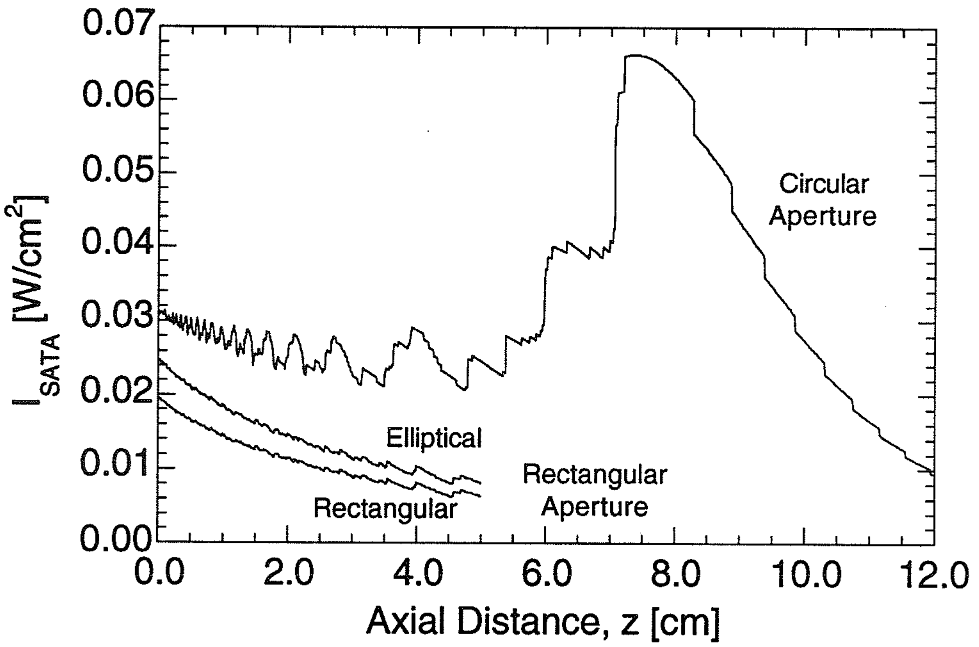

Figure 4.14: Comparison of the spatial average temporal average intensity for the circular aperture case of Section 4.1 and the square aperture case (both rectangular and elliptical cross-section I_{SATA} values are presented).

CHAPTER 5

APPLICATION TO LAYERED MEDIA

The theory and results of the general solution presented in Chapters 3 and 4 are restricted to the case of a homogeneous medium. In this chapter, the medium is modeled as a series of layered tissues. This model is a more representative model of the tissues encountered during the use of clinical diagnostic ultrasound.

5.1 Multilayer Geometry

The geometry of the multilayered tissue case is shown in Figure 5.1.

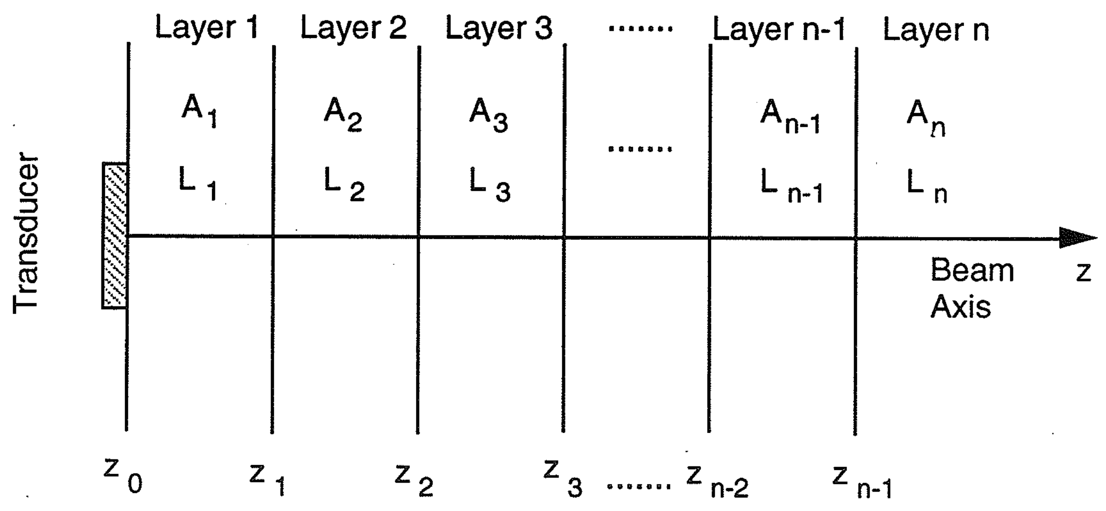

Figure 5.1: Geometry for multilayered medium. A_i is the attenuation coefficient of layer i; z_i is the axial distance of the boundary between layer i and layer i+1.

In this case, each layer is modeled with uniform but not necessarily the same thicknesses and with the boundaries perpendicular to the ultrasound beam axis. The boundaries are planar. Each layer of tissue has a unique attenuation (=absorption) coefficient denoted by

A_i, and a unique perfusion length, L_i, where i is the layer number. As indicated in Figure 5.1, layers are numbered consecutively, with the layer in contact with the transducer labeled as Layer 1. For this simulation, all other tissue characteristics (K, ρ, and c) are defined the same for each layer, i.e., $c_1=c_2=c_3...c_{n-1}=c_n$. In addition, it is assumed that all of the energy is transmitted through each layer, i.e., there is no reflection of energy at the tissue boundaries.

5.2 Intensity Field Distribution

Due to the different attenuation values for each of the layers of this tissue model, the calculation of the relative pressure (Equation (3.4)) must modified. The equation representing the relative pressure for the layered media case ($n \geq 2$) is

$$p_{rel}(x_m,y_m,z_m)=\sum\left[\frac{\exp(-jk\,|\mathbf{P}_m-\mathbf{P}_t|)}{|\mathbf{P}_m-\mathbf{P}_t|}\left(\prod_{i=1}^{n-1}\exp(-A_i\,|\mathbf{P}_i-\mathbf{P}_{i-1}|)\right)\exp(-A_n|\mathbf{P}_m-\mathbf{P}_{n-1}|)\right] \tag{5.1}$$

where n is defined as the number of the layer that contains P_m, A_i is defined as in Section 5.1, and P_i is defined as the point of intersection of $\mathbf{P}_m$-$\mathbf{P}_t$ and the boundary between layers i and i-1 (for i=0, P_0 is defined as P_t). The product term accounts for the attenuation in all of the layers preceding the layer containing P_m. The last term represents the attenuation of the signal by the layer containing the point, P_m.

The relative intensity at the point, $P_m(x_m,y_m,z_m)$, is calculated using Equation (3.6) as for the homogeneous case. However, in order to scale the relative intensity to reflect the source power, W_0, the fact that the attenuation coefficient is not a constant throughout the media must be considered. Therefore, Equation (3.7) is rewritten as

$$W(z) = W_0\left[\prod_{i=1}^{n-1}\exp(-A_i(z_{i+1}-z_i))\right]\exp(-A_{i+1}(z-z_i)) \tag{5.2}$$

where i and n are defined as for Equation (5.1) and A_i and z_i are defined as in Section 5.1. Then the equations for G(z), Equation (3.8), and $I(x_m,y_m,z_m)$, Equation (3.9), are applied to the actual intensity value at $P_m(x_m,y_m,z_m)$.

5.3 Temperature Increase

When calculating the temperature increase using the general solution for the layered tissue model, it must be noted that the rate of heat generation by a point source is dependent on the absorption coefficient of the tissue that contains the point source, that is,

$$q_v = 2\alpha_n I(x_m,y_m,z_m) \tag{5.3}$$

Thus when calculating the temperature increase as described in Section 3.4.2, Equation (3.15) must be modified accordingly. In addition, the possibility of a different perfusion length for each medium must be accounted for.

5.4 Fetal Imaging

5.4.1 Tissue model

The layered tissue model of Section 5.1 is applicable to fetal imaging [6]. In ultrasonic imaging of the fetus, typical imaging consists of the ultrasound beam propagating through the anterior abdominal wall and the fluid filled bladder. Figure 5.2 is Figure 5.1 redrawn to define the actual layered case in question. In Figure 5.2, A_{aw}, A_b, and A_c are the attenuation values of the abdominal wall, bladder and conceptus, respectively. The thicknesses of the abdominal wall and the bladder are given as d_{aw} and d_b, respectively.

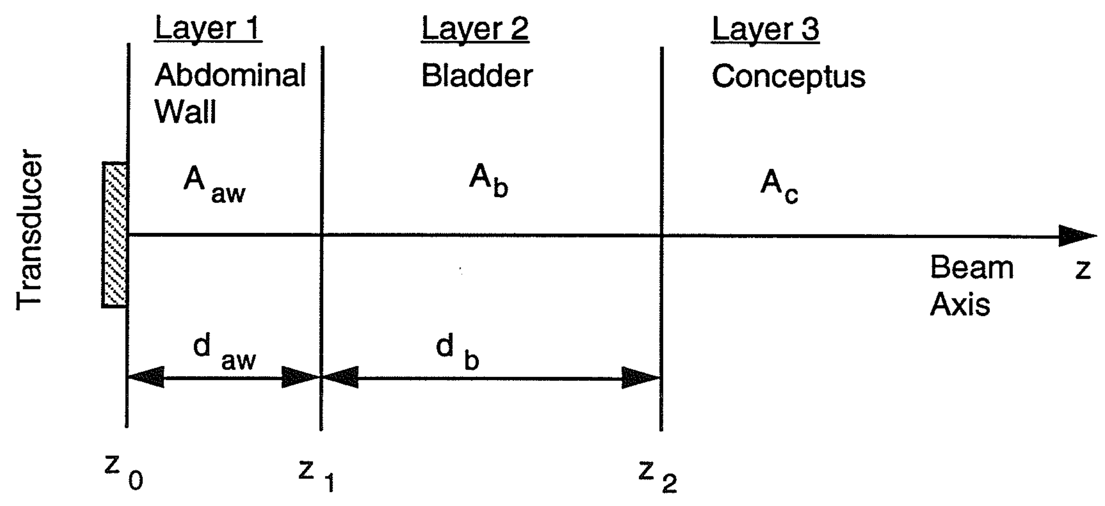

Figure 5.2: Layered tissue model representing tissues normally encountered during diagnostic fetal imaging.

5.4.2 Cases studied

In order to evaluate the effect of the thickness of the maternal abdominal wall on the temperature increase in the tissue, four cases are evaluated. The first case is that of the homogeneous tissue model of Chapters 3 and 4. This model is used as a baseline result. This is due to the fact that the FDA regulates ultrasound equipment on the homogeneous tissue model. The homogeneous intensity field is evaluated to determine the power that will yield a spatial peak temporal average intensity (I_{SPTA}) of 720 mW/cm^2, the maximum derated value allowed by FDA. Using this power level, the layered tissue model is then evaluated. For the homogeneous case studied, all of the circular aperture transducer and tissue values of Section 4.1 were used with the exception of the attenuation of the tissue (FDA's derating factor of 0.3 dB/MHz-cm) and the source power, W_0. The data presented for the homogeneous and layered media data are for a source power, W_0, of 225 mW, based on I_{SPTA} of 720 mW/cm^2. Table 5.1 lists the cases studied for the layered tissue model. The resulting intensity field data and axial temperature increases generated for each of these cases were analyzed. Table 5.2 presents the I_{SPTA} as well as the maximum temperature increases in the abdominal wall and the fetal tissue.

Table 5.1. Tissue parameters used in the layered tissue model.

		Case 1	Case 2	Case 3
d_{aw}	abdominal wall thickness (cm)	0.1	1.0	2.0
d_b	bladder thickness (cm)	5.0	5.0	5.0
A_{aw}	abd. wall attenuation (dB/cm-MHz)	0.5	0.5	0.5
A_b	bladder attenuation (dB/cm-MHz)	0.0	0.0	0.0
A_c	conceptus attenuation (dB/cm-MHz)	0.3	0.3	0.3

Table 5.2. I_{SPTA} and temperature increase data for the cases presented in Table 5.1.

		Case 1	Case 2	Case 3
I_{SPTA}	SPTA intensity (mW/cm^2)	2007	1772	1543
$\Delta T_{max;aw}$	abd wall max temp increase (°C)	0.10	0.83	1.10
$\Delta T_{max;c}$	conceptus max temp increase (°C)	1.52	1.23	0.94

Figures presenting plots for the beam width, I_{SATA} and the temperature increase profiles are presented in Appendix D. For each graph, the results of the homogeneous tissue model are provided for comparison. These examples demonstrate the potential of the general solution approach to the case of fetal imaging.

5.5 Small Parts and Eye Imaging

In addition to the applicability of the general solution to fetal imaging, it may also be applied to the case of small parts and eye imaging. In this section, the example of imaging of the eye is presented.

5.5.1 Tissue model

Typical eye imaging consists of the ultrasound beam propagating through the lens and the vitreous humor. The tissue model is presented in Figure 5.3.

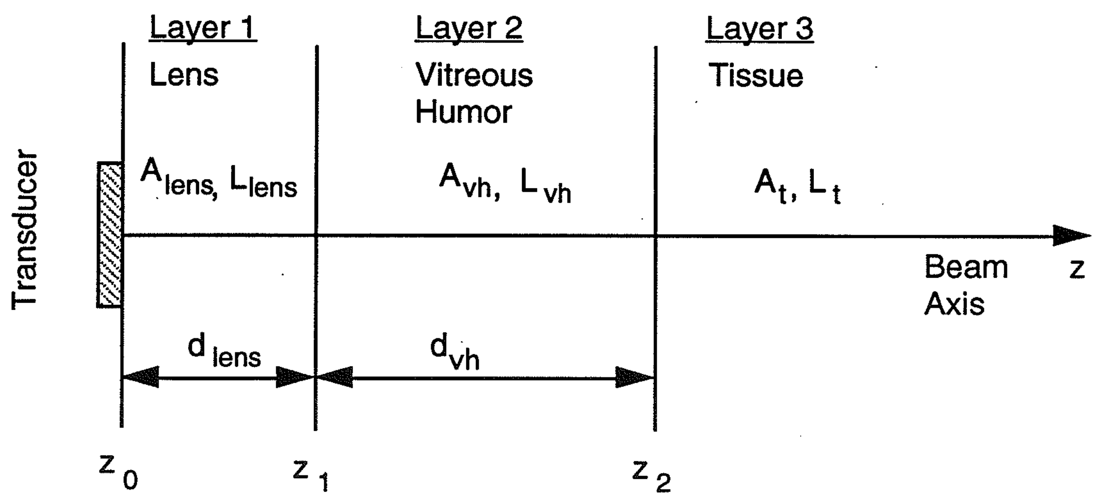

Figure 5.3: Layered tissue model for the eye.

5.5.2 Cases studied

In this study, the tissue characteristics were held constant while the size of the transducer was varied. Values for the attenuation and perfusion lengths of the different structures as well as their thicknesses are given in Table 5.3. It should be noted that in this case, different perfusion lengths were used for the lens and the tissue. This is due to the fact that the lens is a special case and perfusion is quite low.

Table 5.4 lists the specific transducer parameters used in this study. The physical difference between the transducers is their diameters. The source powers are determined for an I_{SPTA} of 720 mW/cm^2 in a homogeneous tissue with a derating factor of 0.3 dB/cm-MHz as is the fetal imaging case.

Table 5.5 presents the I_{SPTA} for each of the cases as well as the maximum temperature increase in the lens and the tissue. Figures providing the beam widths for the two transducers as well as the I_{SATA} and temperature increase plots are presented in Appendix E. These cases demonstrate the application of the general solution to a layered medium and the flexibility of the transducer and tissue models that can be used.

Table 5.3. Tissue characteristics of the eye.

d_{lens}	lens thickness	0.4 cm
d_{vh}	vitreous humor thickness	1.8 cm
A_{lens}	lens attenuation	2.0 dB/cm-MHz
A_{vh}	vit. humor attenuation	0.0 dB/cm-MHz
A_t	tissue attenuation	0.7 dB/cm-MHz
L_{lens}	lens perfusion	4.0 cm
L_t	tissue perfusion	1.18 cm

Table 5.4. Physical transducer operating parameters for the eye imaging cases.

		Case 4	Case 5
f	frequency (MHz)	7.5	7.5
D	source diameter (cm)	1.0	0.5
W_0	source power (mW)	353	66.7

Table 5.5. I_{SPTA} and temperature increase data for the transducer cases of Table 5.4.

		Case 4	Case 5
I_{SPTA}	SPTA intensity (mW/cm^2)	640	613
$\Delta T_{max;lens}$	lens temp increase (°C)	9.95	3.10
$\Delta T_{max;t}$	tissue max temp increase (°C)	3.09	0.68

CHAPTER 6

CONCLUSIONS

A new method for calculating the intensity field distribution and the associated temperature increase in the tissue has been presented. The main feature of this method is its flexibility with the type of transducer as well as with the tissue models. Previously published methods have been restricted to circular transducers due to the complex computations required. The general solution does not require any ultrasound beam symmetry to calculate the temperature increase in tissue exposed to ultrasound. In addition, the method is not restricted to calculating the axial temperature increase. The temperature increase at any point in the medium may be determined.

The general solution has been compared to the accepted heated disc method and produced the same results. In addition, the general solution has extended the ability to calculate the temperature increase to the case of the rectangular aperture transducer. As discussed in Chapter 4, the relationship between the aspect ratio of a rectangular aperture transducer and the maximum temperature increase has been developed. As the aspect ratio of the rectangular transducer increases, the maximum temperature increase decreases. Also, for the square aperture transducer, a slightly lower temperature increase than for the circular aperture case was shown.

Finally, the general solution has been applied to the case of a layered tissue model for both fetal imaging and eye imaging. As more data become available on the tissue layers involved in fetal imaging, more accurate estimations of the temperature increase of fetal tissue will be possible.

The potential of the general solution has only begun to be explored. Continuing work will focus on applying the general solution to a variety of different transducer characteristics, specifically: frequency, diameter, and f-number (focussing) for the circular aperture case and frequency, f-number, as well as two dimensional focussing, i.e., focussing in both the imaging and elevational planes for the rectangular aperture. In addition, the theoretical values will be compared to actual experimental measurements of circular aperture transducers. Also, as more data become available on the characteristics of the tissues involved in fetal imaging, temperature increases will be calculated for actual experimental data.

In summary, the general solution has been demonstrated successfully but will still have need to undergo a substantial test with a variety of transducer parameters as well as changes in the tissue models.

REFERENCES

[1] M.W. Miller and M.C. Ziskin, "Biological consequences of hyperthermia," *Ultrasound Med. Biol.*, vol. 15, no. 8, pp. 707-722, 1989.

[2] V. Abraham, M.C. Ziskin, and S. Heyner, "Temperature elevation in the rat fetus due to ultrasound exposure," *Ultrasound Med. Biol.*, vol. 15, pp. 443-449, 1989.

[3] American Institute of Ultrasound in Medicine (AIUM), Bioeffects Committee, "Biological considerations for the safety of diagnostic ultrasound," *J. Ultrasound Med.*, vol. 7, no. 10 (Supplement), pp. S8-S17, 1988.

[4] L.E. Kinsler, A.R. Frey, A.B. Coppens, and J.V. Sanders. Fundamentals of Acoustics 3rd ed. New York: John Wiley & Sons. 1982.

[5] NCRP, "Biological effects of ultrasound: mechanisms and clinical implications," National Council on Radiation Protection and Measurements (NCRP) Report No. 74 (NCRP Publications, Bethesda, MD).

[6] W.L. Nyborg, "Heat generation by ultrasound in a relaxing medium," *J. Acoust. Soc. Am.*, vol. 70, no. 2, pp. 310-312, 1981.

[7] K.E. Thomensieus and P.A. Lewin, "Ultrasound bioeffects 1991: an update," *Ultrasound Quarterly*, vol. 9, no. 2, pp. 111-137, 1991.

[8] E.L. Carstensen, S.Z. Child, S. Norton, and W.L. Nyborg, "Ultrasonic heating of the skull," *J. Acoust. Soc. Am.*, vol. 87, no. 3, pp. 1310-1317, 1990.

[9] J.L Drewniak, K.I. Carnes, and F. Dunn, "*In vitro* ultrasonic heating of fetal bone," *J. Acoust. Soc. Am.*, vol. 86, pp. 1254-1258, 1989.

[10] NCRP, 1992. In preparation.

[11] H.H. Pennes, "Analysis of tissue and arterial blood temperatures in the resting human forearm," *J. Appl. Physiol.*, vol. 1, pp. 93-122, 1948.

[12] W.L. Nyborg, "Solutions of the bio-heat transfer equation," *Phys. Med. Biol.*, vol. 33, no. 7, pp. 785-792, 1988.

[13] N.B. Hornback, "Hyperthermia and cancer: human clinical trial experience," vols. I and II, Florida: CRC Press, 1984.

[14] A.J. Cheung and A. Neyzari, "Deep local hyperthermia for cancer therapy: external electromagnetic and ultrasound techniques," *Cancer Res.* (Supplement), vol. 44, pp. 4736-4744S, 1984.

[15] F.A. Duck, H.C. Starritt, and S.P. Anderson, "A survey of the acoustic output of ultrasonic Doppler equipment," *Clin. Phys. Physiol.*, vol. 8, pp. 39-49, 1987.

[16] A.K. Chan, R.A. Sigelmann, A.W. Guy, and J.F. Lehmann, "Calculation by the method of finite differences of the temperature distribution in layered tissues," *IEEE Trans. Biomed. Engr.*, vol. BME-20, no. 2, pp. 86-90, 1973.

[17] F.L. Lizzi and M. Ostromogilsky, "Analytical modelling of ultrasonically induced tissue heating," *Ultrasound Med. Biol.*, vol. 13, no. 10, pp. 607-618, 1987.

[18] W.L. Nyborg and R.B. Steele, "Temperature elevation in a beam of ultrasound," *Ultrasound Med. Biol.*, vol. 9, pp. 611-620, 1983.

[19] W.L. Nyborg, "NCRP-AIUM models for temperature calculations," *Ultrasound Med. Biol.*, vol. 15, Supplement No. 1, pp. 37-40, 1989.

[20] W.L. Nyborg and W.D. O'Brien, "An alternative simple formula for temperature estimates," *J. Ultrasound Med.*, vol. 8, pp. 653-654, 1989.

[21] G. Kossoff, "Analysis of focusing action of spherically curved transducers," *Ultrasound Med. Biol.*, vol. 5, pp. 359-365, 1979.

APPENDIX A

COMPUTER PROGRAMS FOR A CIRCULAR APERTURE FOCUSSED TRANSDUCER AND A HOMOGENEOUS MEDIUM

A.1 Intensity Field Distribution

```
/*   field.c

This program computes the intensity field distribution for a circular
aperture focussed transducer.

*/

#include <math.h>      /* include the math library  */

#define pi 3.141592654

/* constant definitions used to describe the medium  */

float c              = 1540.0;     /* speed of ultrasound wave (m/sec)     */
float attenuation    = 0.1382;     /* attenuation of wave (np/cm)          */
float density        = 0.001;      /* density (rho)  (kg/cm^3)             */
float L              = 1.18;       /* perfusion length (cm)                */
float K              = 0.006;      /* thermal conductivity coeff (W/cm^2) */

/* constant definiton used to describe the transducer */

float diameter       = 2.0;        /* diameter of transducer (cm)     */
float ROC            = 10.0;       /* radius of curvature (cm)        */
float freq           = 4000000.0;  /* frequency (Hz)                  */
float power          = 0.100;      /* power output (W)                */

/* constant defintions used to define medium space  */

float space_step     = 0.01;       /* step size to map space (cm)  */
float axis_size      = 15.0;       /* maximum axial distance       */
float t_step         = 6.0;      /* step size to map transduer.  lambda/t_step */
float area           = 0.0;

/* variable declarations  */

float lambda,                    /* wavelength (cm) */
      transducer_step,           /* step size to map transducer  */
      wave_num,                  /* wave number k (rad/cm)       */
      radius,                    /* radius of transducer         */
      inv_impedance;             /* inverse value of impedance   */
      radius_steps;              /* number of transducer steps in radius */

float intensity ;  /* calculated intensity values */


/*********************************************
   function initialization
```

```
This function initializes several global variables.

********************************************/
void initialization ()
{
int z;

     lambda = (c/freq)*100;
     transducer_step = lambda/t_step;
     wave_num = 2*pi/lambda;
     radius = diameter/2;
     area = transducer_step*transducer_step;
     inv_impedance = 1/(density * c);
     radius_steps = (int) (radius/transducer_step);
}

/********************************************
   function data_file_header

This function will output the values of the constants used in
the calculation of the intensity.

********************************************/
void data_file_header (axis_start, axis_stop)
float axis_start, axis_stop;
{
   printf ("%f %f %f %f %f %f %f %f %f %f %f %f %f\n", c, attenuation, density,
L, K, diameter, ROC, freq, power, t_step, space_step,axis_start, axis_stop);

}

/********************************************
   function calc_intensity

This function calculates the intensity at a point
in space by using superposition of the points on
the face of the transducer.  The transducer is mapped
in to several grid points.

********************************************/
void calc_intensity (z)
float z;
{
   float x_map, y_map, z_map;  /* indices used to map the transducer       */
   int   x_step, y_step;       /* integral values for mapping steps         */
   float r1,r2;                /* distance from transducer to point in space */
   float x=0.0;                /* lateral distance                          */
   float a,b;                  /* coefficients of complex term a+jb         */
   float pressure_real,
         pressure_imag;        /* coefficients for complex pressure         */
   float loss1,loss2;          /* loss coefficent for lossy material        */
   int   y_max;                /* the maximum y_map value to maintain a
                                  spherical geometry                        */
   float z_map_2, y_map_2;

   printf ("%d %f\n", (int) ((1.0)/space_step +1.0), z);
   for (x=0.0; x<= 1.0; x+=space_step)
   {
      a=0.0; b=0.0;    /* initialize coefficients */
      for (x_step = 0; x_step <radius_steps; x_step++)
      {
         x_map = (x_step + 0.5) * transducer_step;
         y_max = (int) (radius_steps*sqrt(radius*radius - x_map*x_map));

         for (y_step = 0; y_step <= y_max; y_step++)

         {
            y_map = (y_step + 0.5) * transducer_step;

            y_map_2 = y_map*y_map;
            z_map = ROC - sqrt((ROC*ROC) -(x_map*x_map) - y_map_2);
            z_map_2 = (z_map-z)*(z_map-z);

            r1= sqrt ( (x-x_map)*(x-x_map) + y_map_2 + z_map_2);
            r2= sqrt ( (x+x_map)*(x+x_map) + y_map_2 + z_map_2);

            loss1 = exp (-attenuation*r1);
            loss2 = exp (-attenuation*r2);

            a+= loss1*cos(r1*wave_num)/r1 + loss2*cos(r2*wave_num)/r2;
            b+= loss1*sin(r1*wave_num)/r1 + loss2*sin(r2*wave_num)/r2;
         } /* end for y_map  */
      } /* end for x_map */
      pressure_real = 2.0*area*a;  /* 2.0 * becuase it mapped a hemisphere */
      pressure_imag = 2.0*area*b;
      intensity = (pressure_real*pressure_real +
pressure_imag*pressure_imag)*0.5*inv_impedance;
      printf ("%e ", intensity);
   }  /* end for x */
   printf ("\n");
} /* end calc_intensity */

main ()
{
int axis_start_step, axis_stop_step, z_step;
float axis_start, axis_stop,
      z;

   axis_start = space_step;
   axis_stop = 12.0;

   axis_start_step = (int) (axis_start/space_step +0.5);
   axis_stop_step = (int) (axis_stop/space_step +0.5);


   initialization ();
   data_file_header (axis_start, axis_stop);

   for (z_step = axis_start_step; z_step<= axis_stop_step; z_step+= 1)
   {
      z = z_step *space_step;
      calc_intensity (z);
   }
}
```

A.2 Beam Width and Intensity Calculations

```
/* bwinten.c

This program analyzes the field data and generates a table of relevant field
information.  Calculated values include 10 dB beam width, I(SATA), I(TA),
and the power at an axial distance, z.
*/

#include <math.h>
/* #defines for the field size definiton. */

#define true 1
#define false 0
#define pi 3.141592654
#define x_array 101
#define y_array 101
#define x_max 100
#define y_max 100

/* Parameters used to quantify the field and the analysis parameters. */

float attenuation;          /* attenuation of wave (np/cm)          */
float L;                    /* perfusion length (cm)                */
float K;                    /* thermal conductivity coeff (W/cm^2) */
float space_step;           /* step size to map space (cm)   */
float x_lateral=1.5;        /* lateral distance in x direction to scan */
float transducer_power = 0.2769;   /* transducer source power output */
float axis_start, axis_stop;


/*************************************
   function read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

*************************************/

void read_header_data ()
{
   scanf ("%*f %f %*f %f %f %*f %*f %*f %*f %*f %f %f %f", &attenuation, &L, &K,
&space_step, &axis_start, &axis_stop);


}


/***************************************
 function read_array (array)

This function reads data from stdin and loads it into an array.

***************************************/

void read_array (array)
float array [x_array][y_array];
{
```

```
int x_index, y_index;

   for (x_index=0; x_index <=x_max; x_index++)
      for (y_index=0; y_index<= y_max; y_index ++)
         array[x_index][y_index]= 0.0;

   scanf ("%*f %*f");   /* this line skips the data line containing location
information */
   for (x_index =0; x_index<= x_max ; x_index ++)
     {
         scanf ("%f", &array[x_index][0]);
        /* printf ("%d %f \n", x_index, array[x_index][0]); */
     }
}
/***************************************
  function output_array (array)

This function outputs the contents of array.

***************************************/

void output_array (array)
float array[x_array][y_array];
{
int x_index, y_index;

   for (x_index =0; x_index<=x_max; x_index++)
   {
      for (y_index =0; y_index<= y_max; y_index++)
          printf ("%f ", array[x_index][y_index]);
      printf ("\n");
   }
   printf ("\n");
}

/***************************************
  function interpolate_array (array)

This function interpolates the one-dimensional array
into a two-dimensional array using the circular symmetry
of the problem studied.

***************************************/
void interpolate_array (array)
float array[x_array][y_array];
{
int x_index, y_index, integer;
float distance, fraction;

   for (x_index = 0; x_index <= x_max; x_index++)
      for (y_index = 1; y_index <= y_max; y_index ++)
      {
         distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
         if ((int) (distance+0.5) < x_max)
           {
             integer = (int) distance;
             fraction = distance - (float) integer;
```

```
              array[x_index][y_index] = array[integer][0] +
fraction*(array[integer +1][0] - array[integer][0]);
            }  /* end if */
      } /* end for y_index */
} /* end interpolate_array */


/***************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

***************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale;
int x_index, y_index;

   for ( y_index =1; y_index <= y_max; y_index++)
      for (x_index =1; x_index <= x_max; x_index ++)
         sum += 4.0*array[x_index][y_index];

   for (x_index = 1; x_index <= x_max; x_index ++)
      sum += 2.0* array[x_index][0];
   for (y_index =1 ; y_index <= y_max; y_index ++)
      sum += 2.0* array[0][y_index];
   sum += array[0][0];


   power = sum * space_step * space_step;
   scale = (transducer_power*exp(-2*attenuation*z))/power;

   printf ("%f %f %f %f ", z, transducer_power*exp(-2*attenuation*z), scale,
scale*array[0][0]);

   return (scale);
}

/************************************
void beamwidth (intensity, z)

This function determines the 10 dB beamwidths.

**************************************/
void beamwidth (intensity, z, scale)

float intensity [x_array][y_array];
float z, scale;
{
int i;
float tendb_value;
int tendb;
int x_index, y_index;
float distance, sum, area, spatial, maxint;
int tendb_index;

tendb_value = 0;
tendb = false;

maxint=0;
   for (i=0; i<x_max; i++)
      if (scale*intensity[i][0] > maxint ) maxint = scale*intensity[i][0];

   for (i=0; i< x_max; i++)
      if ((scale*intensity[i][0] > 0.1*maxint))
         {tendb_value = i*space_step; tendb = true;}


tendb_index = (int) ((tendb_value/space_step) +0.5);

   sum = scale*intensity[0][0];

   for (x_index =1; x_index <= tendb_index; x_index++)
      sum += 2.0*scale*intensity[x_index][0];

   for (x_index = 0; x_index <= x_max; x_index++)
      for (y_index = 1; y_index <= y_max; y_index ++)
      {
         distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
         if ((distance) < tendb_index)
               sum += 4.0*scale*intensity[x_index][y_index];
      }
   area = pi * tendb_value * tendb_value;
   spatial = sum/(area)/(10000);

   printf ("%f %f %f\n", 2*tendb_value,spatial,maxint);
}

main ()
{
int x_index, y_index;
float scale, z;
float intensity [x_array][y_array];

   read_header_data ();

   for (x_index=0; x_index <=x_max; x_index++)
      for (y_index=0; y_index<= y_max; y_index ++)
         intensity[x_index][y_index]= 0.0;

   for (z= 0.01; z<= 12.0; z+= 0.01)
   {

      read_array (intensity);
      interpolate_array (intensity);
      scale = power_scale (intensity, z);
      beamwidth (intensity, z, scale);
   }

}
```

A.3. Temperature Increase

```
/* analyze_circle.c

This program reads in intensity data from stdin and calculates the temperature
contribution from each data point.  The resulting output is the axial intensity
and temperature distribution for the given data file.

*/

#include <math.h>

/* #define statements that define the field size and temperature mask.   */

#define pi 3.141592654
#define x_array 101
#define y_array 101
#define mask_array_lateral 101
#define mask_array_axial 251
#define x_max 100
#define y_max 100
#define mask_max 200

/* Variable definitions used for field parameters as well as transducer
   definitions.  */

float attenuation;          /* attenuation of wave (np/cm)          */
float L;                    /* perfusion length (cm)                */
float K;                    /* thermal conductivity coeff (W/cm^2) */
float space_step;           /* step size to map space (cm)   */
float x_lateral=1.5;        /* lateral distance in x direction to scan */
float transducer_power = 0.2769;
int mask_size_lateral, mask_size_axial;
float axis_start, axis_stop;

float mask[mask_array_lateral][mask_array_lateral][mask_array_axial];
float temp[1451];


/*************************************
   funtion read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

*************************************/

void read_header_data ()
{
   scanf ("%*f %f %*f %f %f %*f %*f %*f %*f %*f %f %f %f", &attenuation, &L, &K,
&space_step, &axis_start, &axis_stop);
   printf ("%f %f\n", axis_start, axis_stop);

}

/********************************************
```

```
   function mask_create

 This function creates the mask used for
 superposition of the temperature increases
 due to each point in the field.

 *********************************************/
 void mask_create ()
 {
 int i, j, k;
 float r, coefficient;

      coefficient = (attenuation*(space_step*space_step*space_step))/(2*pi*K);

      for (i = 0; i <= mask_size_lateral; i++)
         for (j = 0; j <= mask_size_lateral; j++)
            for (k = 0; k <= mask_size_axial; k++)
            if (i != 0 && j != 0 && k != 0)
            {
               r=sqrt((i*space_step)*(i*space_step) +
 (j*space_step)*(j*space_step) + (k*space_step)*(k*space_step));
               mask[i][j][k] = coefficient*exp(-r/L)/r;
            } /* end if */
 } /* end mask_create */


 /***************************************
  fucntion read_array (array)

 This function reads data from stdin and loads it into
 an array.

 ***************************************/

 void read_array (array)
 float array [x_array][y_array];
 {
 int x_index, y_index;

    for (x_index=0; x_index <=x_max; x_index++)
       for (y_index=0; y_index<= y_max; y_index ++)
          array[x_index][y_index]= 0.0;

    scanf ("%*f %*f");   /* this line skips the data line containing location
 information */
    for (x_index =0; x_index<= x_max ; x_index ++)
      {
          scanf ("%f", &array[x_index][0]);
         /* printf ("%d %f \n", x_index, array[x_index][0]); */
      }
 }
 /***************************************
   function output_array (array)

 This function outputs the contents of array.

 ***************************************/
```

```
void output_array (array)
float array[x_array][y_array];
{
int x_index, y_index;

   for (x_index =0; x_index<=x_max; x_index++)
   {
      for (y_index =0; y_index<= y_max; y_index++)
         printf ("%f ", array[x_index][y_index]);
      printf ("\n");
   }
   printf ("\n");
}

/****************************************
  function interpolate_array (array)

This function interpolates the one-dimensional array
into a two-dimensional array using the circular symmetry
of the problem studied.

****************************************/
void interpolate_array (array)
float array[x_array][y_array];
{
int x_index, y_index, integer;
float distance, fraction;

   for (x_index = 0; x_index <= x_max; x_index++)
      for (y_index = 1; y_index <= y_max; y_index ++)
      {
         distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
         if ((int) (distance+0.5) < x_max)
           {
             integer = (int) distance;
             fraction = distance - (float) integer;
             array[x_index][y_index] = array[integer][0] +
fraction*(array[integer +1][0] - array[integer][0]);
           }  /* end if */
      } /* end for y_index */
} /* end interpolate_array */


/****************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

****************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale;
```

```
 int x_index, y_index;

    for ( y_index =1; y_index <= y_max; y_index++)
       for (x_index =1; x_index <= x_max; x_index ++)
          sum += 4.0*array[x_index][y_index];

    for (x_index = 1; x_index <= x_max; x_index ++)
       sum += 2.0* array[x_index][0];
    for (y_index =1 ; y_index <= y_max; y_index ++)
       sum += 2.0* array[0][y_index];
    sum += array[0][0];

    power = sum * space_step * space_step;
    scale = (transducer_power*exp(-2*attenuation*z))/power;
    printf ("%f %f %f %f %f\n", z, sum, scale, array[0][0],scale*array[0][0]);
    return (scale);
 }

 /*************************************
   function temperature

 This function uses the mask and the values
 of intensity to superimpose the effect of points
 in space on the temperature increase along the
 axis.

 *************************************/
 void temperature (intensity, scale, z)
 float intensity[x_array][y_array];
 float scale;
 float z;
 {
 int x_index, y_index, z_index, k;
 float scale2;

    z_index = (int) (z/space_step +0.5);
    for (y_index =0; y_index <= mask_size_lateral; y_index ++)
       for (x_index =0 ;x_index <=mask_size_lateral; x_index ++)
       {
          scale2 = 2.0;
          if ((x_index !=0) && (y_index !=0) ) scale2 = 4.0;
          if ((x_index) ==0 && (y_index == 0) ) scale2 = 1.0;

          for (k = -mask_size_axial; k<= mask_size_axial; k++)
          if ((z_index + k) > 0)
             temp[z_index+k] +=
scale*scale2*intensity[x_index][y_index]*mask[x_index][y_index][abs(k)];
       }

} /* end temperature */


main ()
{
int x_index, y_index;
float scale, z;
float intensity [x_array][y_array];
```

```
   read_header_data ();
   mask_size_lateral= 100;
   mask_size_axial=250;
   printf ("mask size lateral= %d  axial =%d\n", mask_size_lateral,
mask_size_axial);
   mask_create ();
   printf ("mask created\n");

   for (x_index=0; x_index <=x_max; x_index++)
      for (y_index=0; y_index<= y_max; y_index ++)
         intensity[x_index][y_index]= 0.0;

   for (z= 0.01; z<= 12.0; z+= 0.01)
   {

      read_array (intensity);
      interpolate_array (intensity);
      scale = power_scale (intensity, z);
      temperature (intensity, scale, z);
   }

   for (z= 0.01; z <= 12.0; z+= 0.01)
      printf ("%f %f\n", z, temp[(int) (z/0.01 +0.5)]);
}
```

APPENDIX B

COMPUTER PROGRAMS FOR A RECTANGULAR APERTURE FOCUSSED TRANSDUCER AND A HOMOGENEOUS MEDIUM

B.1 Intensity Field Distribution

```
/*   field.c

This program generates the intensity field due to a rectangular
aperture transducer.

*/

#include <math.h>     /* include the math library  */

#define pi 3.141592654

/* constant definitions used to describe the medium  */

float c              = 1540.0;     /* speed of ultrasound wave (m/sec)     */
float attenuation    = 0.15;       /* attenuation of wave (np/cm)          */
float density        = 0.001;      /* density (rho)  (kg/cm^3)             */
float L              = 1.18;       /* perfusion length (cm)                */
float K              = 0.006;      /* thermal conductivity coeff (W/cm^2)  */

/* constant definiton used to describe the transducer */

float x_width        = 3.14;       /* width in x-direction (focused)    */
float y_width        = 1.00;       /* width in y-direction (unfocussed)*/
float ROC            = 10.0;       /* radius of curvature (cm)          */
float freq           = 3000000.0;  /* frequency (Hz)                    */
float power          = 0.100;      /* power output (W)                  */

/* constant defintions used to define medium space  */
float x_lateral      = 2.5;        /* lateral distance in x direction to scan    */
float y_lateral      = 1.5;        /* lateral distance in y-direction to scan    */
float space_step     = 0.025;      /* step size to map space (cm)                */
float axis_size      = 15.0;       /* maximum axial distance                     */
float t_step         = 6.0;        /* step size to map transduer.  lambda/t_step */
float area           = 0.0;

/* variable declarations  */

float lambda,                   /* wavelength (cm)                */
      transducer_step,          /* step size to map transducer  */
      wave_num,                 /* wave number k (rad/cm)         */
      inv_impedance;            /* inverse value of impedance     */
      x_map_steps,              /* number of transducer steps in x & y dir.  */
      y_map_steps;
      x_steps, y_steps;         /* number of lateral steps in x&y dir. */
```

```
float intensity ;  /* calculated intensity values */


/********************************************
   function initialization

This function initializes several global variables.

*********************************************/
void initialization ()
{
int z;

      lambda = (c/freq)*100;
      transducer_step = lambda/t_step;
      wave_num = 2*pi/lambda;
      area = transducer_step*transducer_step;
      inv_impedance = 1/(density * c);
      x_map_steps = (int) (x_width/(2.0*transducer_step));
      y_map_steps = (int) (y_width/(2.0*transducer_step));
      x_steps = (int) (x_lateral/space_step + 0.5);
      y_steps = (int) (y_lateral/space_step + 0.5);
}

/********************************************
   function data_file_header

This function will output the values of the constants used in
the calculation of the intensity.

*********************************************/
void data_file_header (axis_start, axis_stop)
float axis_start, axis_stop;
{
   printf ("%f %f %f %f %f %f %f %f %f %f %f %f %f %f %f %f\n", c, attenuation,
density, L, K, x_width, y_width, ROC, freq, power, t_step, space_step, x_lateral,
y_lateral, axis_start, axis_stop);

}
/********************************************
   function calc_intensity

This function calculates the intensity at a point
in space by using superposition of the points on
the face of the transducer.  The transducer is mapped
into several grid points.

*********************************************/
void calc_intensity (z)
float z;
{
   float x_map, y_map, z_map;         /* Indices used to map the transducer */
   float x_map_2r, x_map_2l, y_map_2t, y_map_2b, z_map_2;
   int   x_map_step, y_map_step;      /* integral values for mapping steps */
   int   x_step, y_step;
   float r1, r2, r3, r4;        /* distance from transducer to point in space */
   float x=0.0,y=0.0;           /* lateral distance */
   float a,b;                   /* coefficients of complex term a+jb  */
```

```
    float pressure_real,
          pressure_imag;                /* coefficients for complex pressure */
    float loss1, loss2, loss3, loss4;  /* loss coeffecent for lossy material */

    printf ("%d %f\n", (int) (x_lateral/space_step +1.5), z);
    for (x_step =0; x_step<= x_steps; x_step++)
    {
      x = x_step* space_step;
      for (y_step=0; y_step<=y_steps ; y_step++)
      {
         y= y_step *space_step;
         a=0.0; b=0.0;    /* initialize coefficients */
         for (y_map_step = 0; y_map_step <=y_map_steps; y_map_step++)
         {
           y_map = (y_map_step + 0.5) * transducer_step;
           for (x_map_step = 0; x_map_step <=x_map_steps; x_map_step++)
           {
              x_map = (x_map_step + 0.5) * transducer_step;
              z_map = ROC - sqrt((ROC*ROC) -(x_map*x_map));

              z_map_2 = (z-z_map)*(z-z_map);
              x_map_2r = (x-x_map)*(x-x_map);
              x_map_2l = (x+x_map)*(x+x_map);
              y_map_2t = (y-y_map)*(y-y_map);
              y_map_2b = (y+y_map)*(y+y_map);

              r1 = sqrt ( x_map_2r + y_map_2t + z_map_2);
              r2 = sqrt ( x_map_2l + y_map_2t + z_map_2);
              r3 = sqrt ( x_map_2l + y_map_2b + z_map_2);
              r4 = sqrt ( x_map_2r + y_map_2b + z_map_2);

              loss1 = exp (-attenuation*r1);
              loss2 = exp (-attenuation*r2);
              loss3 = exp (-attenuation*r3);
              loss4 = exp (-attenuation*r4);

              a+= loss1*cos(r1*wave_num)/r1 +loss2*cos(r2*wave_num)/r2 +
loss3*cos(r3*wave_num)/r3 + loss4*cos(r4*wave_num)/r4;
              b+= loss1*sin(r1*wave_num)/r1 +loss2*sin(r2*wave_num)/r2 +
loss3*sin(r3*wave_num)/r3 + loss4*sin(r4*wave_num)/r4;
           } /* end for x_map_step */
         } /* end for y_map_step */
      pressure_real = area*a;
      pressure_imag = area*b;
      intensity = (pressure_real*pressure_real +
pressure_imag*pressure_imag)*0.5*inv_impedance;
      printf ("%f ", intensity);
     } /* end for y_step */
     printf ("\n");
   }  /* end for x_step */
   printf ("\n");
} /* end calc_intensity */

main ()
{
float axis_start, axis_stop,
      z;
   axis_start = space_step;
   axis_stop = 6.0;

   initialization ();
   data_file_header (axis_start, axis_stop);

   for (z = axis_start; z<= axis_stop; z+= space_step)
      calc_intensity (z);
}
```

B.2 Beam Width and Intensity Calculations

```
/* analyze_array.c

This program analyzes an intensity field data file generated for a rectangular
aperture transducer.  Output consists of the 10 dB beam width in the x and y
directions, I(SATA) for a rectangular and elliptical cross sectional area
and power information

*/

/* constant declarations */

#include <math.h>
#define true 1
#define false 0
#define pi 3.141592654
#define x_array 161
#define y_array 121
#define x_max 160
#define y_max 120

/* declaration of field parameters and simulation parameters */

float attenuation;      /* attenuation of wave (Np/cm)            */
float L;                /* perfusion length (cm)                  */
float K;                /* thermal conductivity coeff (W/cm^2)    */
float space_step;       /* step size to map space (cm)            */
float x_lateral;        /* lateral distance in x direction to scan */
float y_lateral;        /* lateral distance in y-direction to scan */
float transducer_power = 0.100;
float axis_start, axis_stop;

float temp[1200];
/************************************
   funtion read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

*************************************/

void read_header_data ()
{
   scanf ("%*f %f %*f %f %f %*f %*f %*f %*f %*f %*f %f %f %f %f %f",
&attenuation, &L, &K, &space_step, &x_lateral, &y_lateral, &axis_start,
&axis_stop);
   printf ("%f %f\n", axis_start, axis_stop);

}

/**************************************
 fucntion read_array (array)

This function reads data from stdin and loads it into
an array.  The information is loaded using every other row and column of
array.

**************************************/

void read_array (array)
float array [x_array][y_array];
{
int x_index, y_index;

   scanf ("%*f %*f");   /* this line skips the data line containing loaction
information */
   for (x_index =0; x_index<= x_max ; x_index ++)
      for (y_index =0; y_index <= y_max; y_index ++)
         scanf ("%f", &array[x_index][y_index]);
}

/***************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

***************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale;
int x_index, y_index;

   for ( y_index =1; y_index <= y_max; y_index++)
      for (x_index =1; x_index <= x_max; x_index ++)
         sum += 4.0*array[x_index][y_index];


   for (x_index = 1; x_index <= x_max; x_index ++)
      sum += 2.0* array[x_index][0];
   for (y_index =1 ; y_index <= y_max; y_index ++)
      sum += 2.0* array[0][y_index];
   sum += array[0][0];

   power = sum * space_step * space_step;
   scale = (transducer_power*exp(-2*attenuation*z))/power;
   printf ("%f %f %f ", z, transducer_power*exp(-2*attenuation*z),
scale*array[0][0]);
   return (scale);
}

/***********************************
void beamwidth (intensity, z)

This function determines the 10 dB beamwidth in the x and y directions.
It also computes I(SATA).

**************************************/
void bwinten (intensity, z, scale)

float intensity [x_array][y_array];
float z, scale;
{
int i;

float tendb_value_x, tendb_value_y;
int tendb_x, tendb_y;

int x_index, y_index;
float distance, sum, area_rect, area_ellipse, rectangle, ellipse;
float maxint_x, maxint_y;
int tendb_index_x, tendb_index_y;

tendb_value_x = 0;
tendb_value_y = 0;
tendb_x = false;
tendb_y = false;

maxint_x=0;
   for (i=0; i<x_max; i++)
      if (scale*intensity[i][0] > maxint_x ) maxint_x = scale*intensity[i][0];

maxint_y =0;
   for (i=0; i<y_max; i++)
      if (scale*intensity[0][i] > maxint_y ) maxint_y = scale*intensity[0][i];

   for (i=0; i< x_max; i++)
      if ((scale*intensity[i][0] > 0.1*maxint_x))
         {tendb_value_x = i*space_step; tendb_x = true;}

   for (i=0; i<y_max; i++)
      if ((scale*intensity[0][i] > 0.1*maxint_y))
         {tendb_value_y = i*space_step; tendb_y = true;}

tendb_index_x = (int) ((tendb_value_x/space_step) +0.5);
tendb_index_y = (int) ((tendb_value_y/space_step) +0.5);


   sum = scale*intensity[0][0];

   for (x_index = 1; x_index < tendb_index_x; x_index++)
      sum += 2.0*scale*intensity[x_index][0];

   for (y_index =1; y_index < tendb_index_y; y_index++)
```

```
      sum += 2.0*scale*intensity[0][y_index];

   for (x_index = 1; x_index < tendb_index_x; x_index ++)
      for (y_index = 1; y_index < tendb_index_y; y_index ++)
         sum += 4.0*scale*intensity[x_index][y_index];

   area_rect = 4.0*tendb_value_x * tendb_value_y;
   area_ellipse = pi*tendb_value_x*tendb_value_y;
   rectangle = sum/(area_rect)/(10000);
   ellipse = sum/(area_ellipse)/(10000);

   printf ("%f %f %f %f\n",2.0*tendb_value_x,
2.0*tendb_value_y,rectangle,ellipse);
}


main ()
{
float scale, z;
float intensity [x_array][y_array];

   read_header_data ();

   for (z= 0.025; z<= 5.0; z+= 0.0125)
   {

      read_array (intensity);
      scale = power_scale (intensity, z);
      bwinten (intensity, z, scale);
   }

}
```

B.3 Temperature Increase

```
/* analyze_array.c

This program reads in intenisty data from stdin and calculates the temperature
contribution from each data point.  The resulting output is the axial intensity
and temperature distribution for the given data file.


*/

#include <math.h>

/* constant declarations */

#define pi 3.141592654
#define x_array 161
#define y_array 121
#define mask_array_lateral_x 161
#define mask_array_lateral_y 161
#define mask_array_axial 201
#define x_max 160
#define y_max 120
#define mask_max_lateral_x 160
```

```
#define mask_max_lateral_y 120
#define mask_max_axial 200

/* declaration of field variables and simulation parameters */

float attenuation;      /* attenuation of wave (np/cm)          */
float L;                /* perfusion length (cm)                */
float K;                /* thermal conductivity coeff (W/cm^2) */
float space_step;       /* step size to map space (cm)   */
float x_lateral;        /* lateral distance in x direction to scan */
float y_lateral;        /* lateral distance in y-direction to scan */
float transducer_power = 0.100;
int mask_size_lateral_x, mask_size_lateral_y, mask_size_axial;;
float axis_start, axis_stop;

float mask[mask_array_lateral_x][mask_array_lateral_y][mask_array_axial];
float temp[1200];

/************************************
   funtion read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

************************************/

void read_header_data ()
{
   scanf ("%*f %f %*f %f %f %*f %*f %*f %*f %*f %*f %f %f %f %f %f",
&attenuation, &L, &K, &space_step, &x_lateral, &y_lateral, &axis_start,
&axis_stop);
   printf ("%f %f\n", axis_start, axis_stop);

}

/**********************************************
  function mask_create

This function creates the mask used for
superposition of the value of the change
in temperature to be used later.

**********************************************/
void mask_create ()
{
int i, j, k;
float r, coefficient;

      coefficient = (attenuation*(space_step*space_step*space_step))/(2*pi*K);

      for (i = 0; i <= mask_size_lateral_x; ++i)
         for (j = 0; j <= mask_size_lateral_y; ++j)
            for (k = 0; k <= mask_size_axial; ++k)
            if (i != 0 && j != 0 && k != 0)
            {
               r=sqrt((i*space_step)*(i*space_step) +
(j*space_step)*(j*space_step) + (k*space_step)*(k*space_step));
               mask[i][j][k] = coefficient*exp(-r/L)/r;
            } /* end if */
} /* end mask_create */

/*************************************
 fucntion read_array (array)

This function reads data from stdin and loads it into
an array.  The information is loaded using every other row and column
of array.

**************************************/

void read_array (array)
float array [x_array][y_array];
{
int x_index, y_index;

   scanf ("%*f %*f");   /* this line skips the data line containing loaction
information */
   for (x_index =0; x_index<= x_max ; x_index ++)
      for (y_index =0; y_index <= y_max; y_index ++)
         scanf ("%f", &array[x_index][y_index]);
}

/**************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

***************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale;
int x_index, y_index;

   for ( y_index =1; y_index <= y_max; y_index++)
      for (x_index =1; x_index <= x_max; x_index ++)
         sum += 4.0*array[x_index][y_index];


   for (x_index = 1; x_index <= x_max; x_index ++)
      sum += 2.0* array[x_index][0];
   for (y_index =1 ; y_index <= y_max; y_index ++)
      sum += 2.0* array[0][y_index];
   sum += array[0][0];

   power = sum * space_step * space_step;
   scale = (transducer_power*exp(-2*attenuation*z))/power;
   printf ("%f %f %f %f %f\n", z, sum, scale, array[0][0],scale*array[0][0]);
   return (scale);
}

/*************************************
```

```
  function temperature

This function uses the mask and the values
of intensity to superimpose the effect of points
in space on the temperature increase along the
axis.

**************************************/
void temperature (intensity, scale, z)
float intensity[x_array][y_array];
float scale;
float z;
{
int x_index, y_index, z_index, k;
float scale2;

   z_index = (int) (z/space_step +0.5);
   for (y_index =0; y_index <= mask_size_lateral_y; y_index ++)
      for (x_index =0 ;x_index <=mask_size_lateral_x; x_index ++)
      {
         scale2 = 2.0;
         if ((x_index !=0) && (y_index !=0) ) scale2 = 4.0;
         if ((x_index) ==0 && (y_index == 0) ) scale2 = 1.0;

         for (k = -mask_size_axial; k<= mask_size_axial; k++)
         if ((z_index + k) > 0)
            temp[z_index+k] +=
scale*scale2*intensity[x_index][y_index]*mask[x_index][y_index][abs(k)];
      }

} /* end temperature */


main ()
{
float scale, z;
float intensity [x_array][y_array];

   read_header_data ();
   mask_size_lateral_x= 160;
   mask_size_lateral_y= 120;
   mask_size_axial = 200;
   printf ("mask size lateral x = %d mask size lateral y = %d  axial = %d\n",
mask_size_lateral_x, mask_size_lateral_y, mask_size_axial);
   mask_create ();

   for (z= 0.0125; z<= 6.0; z+= 0.0125)
   {

      read_array (intensity);
      scale = power_scale (intensity, z);
      temperature (intensity, scale, z);
   }

   for (z= 0.0125; z <= 6.0; z+= 0.0125)
      printf ("%f %f\n", z, temp[(int) (z/0.0125 +0.5)]);
}
```

APPENDIX C

COMPUTER PROGRAMS FOR A CIRCULAR APERTURE FOCUSSED TRANSDUCER AND A LAYERED MEDIUM

C.1 Intensity Field Distribution

```
/*   field.c

This program calculates the intensity field for a circular aperture focussed
transducer for a layered medium model.
*/

#include <math.h>     /* include the math library  */

#define pi 3.141592654
#define d1 daw
#define d2 (daw+db)

/* constant definitions used to describe the medium  */

float c             = 1540.0;    /* speed of ultrasound wave (m/sec)          */

float A1            = 0.2303;    /* Aaw, attenuation of abdominal wall (Np/cm)*/

float A2            = 0.0;       /* Ab, attenaution of bladder (Np/cm)        */
float A3            = 0.1382;    /* Ac, attenuation of conceptus (Np/cm)      */

float daw           = 1.0;       /* thickness of abdominal wall (cm)          */
float db            = 5.0;       /* thickness of bladder (cm)                 */

float density       = 0.001;     /* density (rho)  (kg/cm^3)                  */
float L             = 1.18;      /* perfusion length (cm)                     */
float K             = 0.006;     /* thermal conductivity coeff (W/cm^2)       */

/* constant definiton used to describe the transducer */

float diameter      = 2.0;       /* diameter of transducer (cm)               */
float ROC           = 10.0;      /* radius of curvature (cm)                  */
float freq          = 4000000.0; /* frequency (Hz)                            */
float power         = 0.100;     /* power output (W)                          */

/* constant defintions used to define medium space  */

float space_step    = 0.01;     /* step size to map space (cm)                */
float axis_size     = 15.0;     /* maximum axial distance                     */
float t_step        = 6.0;      /* step size to map transduer.  lambda/t_step */
float area          = 0.0;

/* variable declarations  */

float lambda,                /* wavelength (cm)               */
```

```
      transducer_step,       /* step size to map transducer   */
      wave_num,              /* wave number k (rad/cm)        */
      radius,                /* radius of transducer          */
      inv_impedance;         /* inverse value of impedance    */

int   radius_steps;          /* number of transducer steps in radius */

float intensity ;            /* calculated intensity values   */


/********************************************
   function initialization

This function initializes several global variables.

********************************************/
void initialization ()
{
int z;

     lambda = (c/freq)*100;
     transducer_step = lambda/t_step;
     wave_num = 2*pi/lambda;
     radius = diameter/2;
     area = transducer_step*transducer_step;
     inv_impedance = 1/(density * c);
     radius_steps = (int) (radius/transducer_step);
}

/********************************************
   function data_file_header

This function will output the values of the constants used in
the calculation of the intensity.

********************************************/
void data_file_header (axis_start, axis_stop)
float axis_start, axis_stop;
{
   printf ("%f %f %f %f %f %f %f %f %f %f %f %f\n", c, density, L, K, diameter,
ROC, freq, power, t_step, space_step,axis_start, axis_stop);

}
/********************************************
   function calc_intensity_1

This function calculates the intensity at a point
in space by using superposition of the points on
the face of the transducer.  The transducer is mapped
into several grid points.

********************************************/
void calc_intensity_layer1 (z)
float z;
{
   float x_map, y_map, z_map;  /* indices used to map the transducer        */
   int   x_step, y_step;       /* integral values for mapping steps          */
   float r1,r2;                /* distance from transducer to point in space */
```

```
    float x=0.0;                /* lateral distance                       */
    float a,b;                  /* coefficients of complex term a+jb      */
    float pressure_real,
          pressure_imag;        /* coefficients for complex pressure      */
    float loss1,loss2;          /* loss coefficent for lossy material     */
    int   y_max;                /* the maximum y_map value to maintain a
                                   spherical geometry                     */
    float z_map_2, y_map_2;

    printf ("%d %f\n", (int) ((1.5)/space_step +1.0), z);


    for (x=0.0; x<= 1.5; x+=space_step)
    {
      a=0.0; b=0.0;     /* initialize coefficients */
      for (x_step = 0; x_step <=radius_steps; x_step++)
      {
        x_map = (x_step + 0.5) * transducer_step;
        y_max = (int) (radius_steps*sqrt(radius*radius - x_map*x_map));

        for (y_step = 0; y_step <= y_max; y_step++)

        {
          y_map = (y_step + 0.5) * transducer_step;

          y_map_2 = y_map*y_map;
          z_map = ROC - sqrt((ROC*ROC) -(x_map*x_map) - y_map_2);
          z_map_2 = (z_map-z)*(z_map-z);

          r1= sqrt ( (x-x_map)*(x-x_map) + y_map_2 + z_map_2);
          r2= sqrt ( (x+x_map)*(x+x_map) + y_map_2 + z_map_2);

          loss1 = exp (-A1*r1);
          loss2 = exp (-A1*r2);

          a+= loss1*cos(r1*wave_num)/r1 + loss2*cos(r2*wave_num)/r2;
          b+= loss1*sin(r1*wave_num)/r1 + loss2*sin(r2*wave_num)/r2;
        } /* end for y_map  */
      } /* end for x_map */
      pressure_real = 2.0*area*a;  /* 2.0 * becuase it mapped a hemisphere */
      pressure_imag = 2.0*area*b;
      intensity = (pressure_real*pressure_real +
pressure_imag*pressure_imag)*0.5*inv_impedance;
      printf ("%e ", intensity);
   }  /* end for x */
   printf ("\n");
} /* end calc_intensity */


/**********************************************
   function calc_intensity_2

This function calculates the intensity at a point
in space by using superposition of the points on
the face of the transducer.  The transducer is mapped
in to several grid points.

**********************************************/
```

```
void calc_intensity_layer2 (z)
float z;
{
   float x_src, y_src, z_src;  /* indices used to map the transducer         */
   int   x_step, y_step;       /* integral values for mapping steps          */
   float r1,r2;                /* distance from transducer to point in space */
   float x=0.0;                /* lateral distance                           */
   float a,b;                  /* coefficients of complex term a+jb          */
   float pressure_real,
         pressure_imag;        /* coefficients for complex pressure          */
   float loss;                 /* loss coefficent for lossy material         */
   int   y_max;                /* the maximum y_map value to maintain a
                                  spherical geometry                         */
   float z_src_2, y_src_2;
   float delta_x, delta_x_2,
         delta_y, delta_y_2,
         delta_z, delta_z_2;   /* vector PmPt */

   float d_src_obs, d_src_1, d_1_obs;  /* distances from src-layers-obs point */

   printf ("%d %f\n", (int) ((1.5)/space_step +1.0), z);


   for (x=0.0; x<= 1.5; x+=space_step)
   {
      a=0.0; b=0.0;    /* initialize coefficients */
      for (x_step = 0; x_step <=radius_steps; x_step++)
      {
         x_src = (x_step + 0.5) * transducer_step;
         y_max = (int) (radius_steps*sqrt(radius*radius - x_src*x_src));

         for (y_step = 0; y_step <= y_max; y_step++)

         {
            y_src = (y_step + 0.5) * transducer_step;

            z_src = ROC - sqrt((ROC*ROC) -(x_src*x_src) - y_src*y_src);

            delta_x = x - x_src;
            delta_x_2 = delta_x*delta_x;
            delta_y = y_src;
            delta_y_2 = delta_y*delta_y;
            delta_z = z-z_src;
            delta_z_2 = delta_z*delta_z;

            d_src_obs = sqrt (delta_x_2 + delta_y_2 + delta_z_2);
            d_src_1 = d_src_obs*(d1-z_src)/(delta_z);
            d_1_obs = d_src_obs*(z-d1)/(delta_z);

            loss = exp (-A1*d_src_1 -A2*d_1_obs);
            a+= loss*cos(d_src_obs*wave_num)/d_src_obs;
            b+= loss*sin(d_src_obs*wave_num)/d_src_obs;

            delta_x = x + x_src; /* x - (-x_src) */
            delta_x_2 = delta_x*delta_x;

            d_src_obs = sqrt (delta_x_2 + delta_y_2 + delta_z_2);
            d_src_1 = d_src_obs*(d1-z_src)/(delta_z);
            d_1_obs = d_src_obs*(z-d1)/(delta_z);

            loss = exp (-A1*d_src_1 -A2*d_1_obs);

            a+= loss*cos(d_src_obs*wave_num)/d_src_obs;
            b+= loss*sin(d_src_obs*wave_num)/d_src_obs;
         } /* end for y_map  */
      } /* end for x_map */
      pressure_real = 2.0*area*a;  /* 2.0 * becuase it mapped a hemisphere */
      pressure_imag = 2.0*area*b;
      intensity = (pressure_real*pressure_real +
pressure_imag*pressure_imag)*0.5*inv_impedance;
      printf ("%e ", intensity);
   }  /* end for x */
   printf ("\n");
} /* end calc_intensity */

/*********************************************
   function calc_intensity_3

This function calculates the intensity at a point
in space by using superposition of the points on
the face of the transducer.  The transducer is mapped
in to several grid points.

*********************************************/
void calc_intensity_layer3 (z)
float z;
{
   float x_src, y_src, z_src;  /* indices used to map the transducer          */
   int   x_step, y_step;       /* integral values for mapping steps           */
   float r1,r2;                /* distance from transducer to point in space */
   float x=0.0;                /* lateral distance                            */
   float a,b;                  /* coefficients of complex term a+jb           */
   float pressure_real,
         pressure_imag;        /* coefficients for complex pressure           */
   float loss;                 /* loss coefficent for lossy material          */
   int   y_max;                /* the maximum y_map value to maintain a
                                  spherical geometry                          */
   float z_src_2, y_src_2;
   float delta_x, delta_x_2,
         delta_y, delta_y_2,
         delta_z, delta_z_2;

   float d_src_obs, d_src_1, d_1_2, d_2_obs;


   printf ("%d %f\n", (int) ((1.5)/space_step +1.0), z);


   for (x=0.0; x<= 1.5; x+=space_step)
   {
      a=0.0; b=0.0;    /* initialize coefficients */
      for (x_step = 0; x_step <=radius_steps; x_step++)
      {
         x_src = (x_step + 0.5) * transducer_step;
         y_max = (int) (radius_steps*sqrt(radius*radius - x_src*x_src));
```

```
          for (y_step = 0; y_step <= y_max; y_step++)

          {
             y_src = (y_step + 0.5) * transducer_step;

             z_src = ROC - sqrt((ROC*ROC) -(x_src*x_src) - y_src*y_src);

             delta_x = x - x_src;
             delta_x_2 = delta_x*delta_x;
             delta_y = y_src;
             delta_y_2 = delta_y*delta_y;
             delta_z = z-z_src;
             delta_z_2 = delta_z*delta_z;

             d_src_obs = sqrt (delta_x_2 + delta_y_2 + delta_z_2);
             d_src_1 = d_src_obs*(d1-z_src)/(delta_z);
             d_1_2 = d_src_obs*(d2-d1)/(delta_z);
             d_2_obs = d_src_obs*(z-d2)/(delta_z);

             loss = exp (-A1*d_src_1 -A2*d_1_2 -A3*d_2_obs);
             a+= loss*cos(d_src_obs*wave_num)/d_src_obs;
             b+= loss*sin(d_src_obs*wave_num)/d_src_obs;

             delta_x = x + x_src; /* x - (-x_src) */
             delta_x_2 = delta_x*delta_x;

             d_src_obs = sqrt (delta_x_2 + delta_y_2 + delta_z_2);
             d_src_1 = d_src_obs*(d1-z_src)/(delta_z);
             d_1_2 = d_src_obs*(d2-d1)/(delta_z);
             d_2_obs = d_src_obs*(z-d2)/(delta_z);

             loss = exp (-A1*d_src_1 -A2*d_1_2 -A3*d_2_obs);

             a+= loss*cos(d_src_obs*wave_num)/d_src_obs;
             b+= loss*sin(d_src_obs*wave_num)/d_src_obs;
          } /* end for y_map  */
       } /* end for x_map */
       pressure_real = 2.0*area*a;  /* 2.0 * becuase it mapped a hemisphere */
       pressure_imag = 2.0*area*b;
       intensity = (pressure_real*pressure_real +
pressure_imag*pressure_imag)*0.5*inv_impedance;
       printf ("%e ", intensity);
    }  /* end for x */
    printf ("\n");
} /* end calc_intensity */



main ()
{
int z_step, axis_start_step, axis_stop_step;
float axis_start, axis_stop,
      z;

   axis_start = space_step;
   axis_stop = 12.0;

   initialization ();
```

```
    data_file_header (axis_start, axis_stop);

    axis_start_step = (int) (axis_start/space_step +0.5);
    axis_stop_step = (int) (axis_stop/space_step +0.5);

    for (z_step = axis_start_step; z_step <= axis_stop_step; z_step ++)
    {
       z = z_step*space_step;

       if (z < d1) calc_intensity_layer1 (z);
       else
          if (z < d2) calc_intensity_layer2 (z);
          else
             calc_intensity_layer3 (z);
    }

}
```

C.2 Beam Width and Intensity Calculations

```
/* bwinten.c

This program analyzes the field data to calculate the 10 dB beam width, I(SATA),
I(TA) and power at an axial distance z along the beam axis.

*/

#include <math.h>

/* #defines for the field size */
#define true 1
#define false 0
#define pi 3.141592654
#define x_array 151
#define y_array 151
#define x_max 150
#define y_max 150

/* Layer parameters to be used */
#define d1 daw
#define d2 (daw+db)

float daw=1.0;       /* thickness of the abdominal wall (cm)   */
float db=5.0;        /* thickness of the bladder        (cm)   */

float  A1 = 0.2303;  /* attenuation coefficient of the abd. wall (Np/cm) */
float  A2 = 0.0;     /* attenuation coefficient of the bladder   (Np/cm) */
float  A3 = 0.1382;  /* attenuation coefficient of the conceptus (Np/cm) */


float L;                /* perfusion length (cm)                          */
float K;                /* thermal conductivity coeff (W/cm^2)            */
float space_step;       /* step size to map space (cm)                    */
float x_lateral=1.5;    /* lateral distance in x direction to scan (cm)*/
float transducer_power = 0.2769;
float axis_start, axis_stop;
```

```
/************************************
   funtion read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

************************************/

void read_header_data ()
{
   scanf ("%*f %*f %f %f %*f %*f %*f %*f %*f %f %f %f", &L, &K, &space_step,
&axis_start, &axis_stop);


}

/**************************************
 fucntion read_array (array)

This function reads data from stdin and loads it into an array.

**************************************/

void read_array (array)
float array [x_array][y_array];
{
int x_index, y_index;

   for (x_index=0; x_index <=x_max; x_index++)
      for (y_index=0; y_index<= y_max; y_index ++)
         array[x_index][y_index]= 0.0;

   scanf ("%*f %*f");   /* this line skips the data line containing loaction
information */
   for (x_index =0; x_index<= x_max ; x_index ++)
     {
         scanf ("%f", &array[x_index][0]);
        /* printf ("%d %f \n", x_index, array[x_index][0]); */
     }
}
/**************************************
  function output_array (array)

This function outputs the contents of array.

**************************************/

void output_array (array)
float array[x_array][y_array];
{
int x_index, y_index;

   for (x_index =0; x_index<=x_max; x_index++)
   {
      for (y_index =0; y_index<= y_max; y_index++)
          printf ("%f ", array[x_index][y_index]);
      printf ("\n");
    }
    printf ("\n");
}

/***************************************
  function interpolate_array (array)

This function interpolates the one-dimensional array
into a two-dimensional array using the circular symmetry
of the problem studied.

***************************************/
void interpolate_array (array)
float array[x_array][y_array];
{
int x_index, y_index, integer;
float distance, fraction;

    for (x_index = 0; x_index <= x_max; x_index++)
       for (y_index = 1; y_index <= y_max; y_index ++)
       {
          distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
          if ((int) (distance+0.5) < x_max)
            {
              integer = (int) distance;
              fraction = distance - (float) integer;
              array[x_index][y_index] = array[integer][0] +
fraction*(array[integer +1][0] - array[integer][0]);
            }  /* end if */
      } /* end for y_index */
} /* end interpolate_array */


/***************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

***************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale, attenuation;
int x_index, y_index;

    for ( y_index =1; y_index <= y_max; y_index++)
       for (x_index =1; x_index <= x_max; x_index ++)
          sum += 4.0*array[x_index][y_index];

    for (x_index = 1; x_index <= x_max; x_index ++)
       sum += 2.0* array[x_index][0];
    for (y_index =1 ; y_index <= y_max; y_index ++)
       sum += 2.0* array[0][y_index];
```

```
   sum += array[0][0];


   power = sum * space_step * space_step;

   if (z < d1) attenuation = A1*z;
      else if (z<d2) attenuation = (A1*d1 + A2*(z-d1));
              else attenuation = (A1*d1 + A2*(d2-d1) + A3*(z-d2));

   scale = (transducer_power*exp(-2*attenuation))/power;

   printf ("%f %f %f %f ", z, transducer_power*exp(-2*attenuation), scale,
scale*array[0][0]);

   return (scale);
}

/************************************
void beamwidth (intensity, z)

This function determines the 10 dB beamwidth.

**************************************/
void beamwidth (intensity, z, scale)

float intensity [x_array][y_array];
float z, scale;
{
int i;
float tendb_value;
int tendb;

int x_index, y_index;
float distance, sum, area, spatial, maxint;
int tendb_index;

tendb_value = 0;
tendb = false;

maxint=0;
   for (i=0; i<x_max; i++)
      if (scale*intensity[i][0] > maxint ) maxint = scale*intensity[i][0];

   for (i=0; i< x_max; i++)
      if ((scale*intensity[i][0] > 0.1*maxint))
         {tendb_value = i*space_step; tendb = true;}


tendb_index = (int) ((tendb_value/space_step) +0.5);

   sum = scale*intensity[0][0];

   for (x_index =1; x_index <= tendb_index; x_index++)
      sum += 2.0*scale*intensity[x_index][0];

   for (x_index = 0; x_index <= x_max; x_index++)
      for (y_index = 1; y_index <= y_max; y_index ++)
      {
            distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
            if ((distance) < tendb_index)
                 sum += 4.0*scale*intensity[x_index][y_index];
         }
     area = pi * tendb_value * tendb_value;
     spatial = sum/(area)/(10000);

     printf ("%f %f %f\n", 2*tendb_value,spatial,maxint);
}

main ()
{
int x_index, y_index;
float scale, z;
float intensity [x_array][y_array];

    read_header_data ();

    for (x_index=0; x_index <=x_max; x_index++)
       for (y_index=0; y_index<= y_max; y_index ++)
          intensity[x_index][y_index]= 0.0;

    for (z= 0.01; z<= 12.0; z+= 0.01)
    {

       read_array (intensity);
       interpolate_array (intensity);
       scale = power_scale (intensity, z);
       beamwidth (intensity, z, scale);
    }

}
```

C.3 Temperature Increase

```
/* analyze_layered.c

This program calculates the axial temperature increase due to a layered medium.
The temperature mask is recalculated for each layer attenuation coefficient.

*/

#include <math.h>

/* layer thicknesses  */

#define d1 daw
#define d2 (daw+db)


/* Constants used to define the field and temperature mask size.  */

#define pi 3.141592654
#define x_array 151
#define y_array 151
#define mask_array_lateral 151
#define mask_array_axial 251
```

```
#define x_max 150
#define y_max 150
#define mask_max 250

/* layer parameters  */
float A1      = 0.2303;    /* Aaw, attenuation of abdominal wall (Np/cm)*/

float A2      = 0.0;       /* Ab, attenaution of bladder (Np/cm)        */
float A3      = 0.1382;    /* Ac, attenuation of conceptus (Np/cm)      */

float daw     = 1.0;       /* thickness of abdominal wall (cm)          */
float db      = 5.0;       /* thickness of bladder (cm)                 */


float L;                 /* perfusion length (cm)                */
float K;                 /* thermal conductivity coeff (W/cm^2) */
float space_step;        /* step size to map space (cm)  */
float x_lateral=1.5;         /* lateral distance in x direction to scan */
float transducer_power = 0.2769;
int mask_size_lateral, mask_size_axial;
float axis_start, axis_stop;

float mask[mask_array_lateral][mask_array_lateral][mask_array_axial];
float temp[1451];

/*************************************
   funtion read_header_data ()

This function reads the first line of the data file and acquires the
necessary data information to interpolate the array.

**************************************/

void read_header_data ()
{
   scanf ("%*f %*f %f %f %*f %*f %*f %*f %*f %f %f %f", &L, &K, &space_step,
&axis_start, &axis_stop);
   printf ("%f %f\n", axis_start, axis_stop);


}

/**********************************************
  function mask_create

This function creates the temperature mask used to calculate the
axial temperature increase from each point in the field.

**********************************************/
void mask_create (attenuation)
float attenuation;
{
int i, j, k;
float r, coefficient;

     coefficient = (attenuation*(space_step*space_step*space_step))/(2*pi*K);

     for (i = 0; i <= mask_size_lateral; i++)
```

```
         for (j = 0; j <= mask_size_lateral; j++)
            for (k = 0; k <= mask_size_axial; k++)
            if (i != 0 && j != 0 && k != 0)
            {
               r=sqrt((i*space_step)*(i*space_step) +
(j*space_step)*(j*space_step) + (k*space_step)*(k*space_step));
               mask[i][j][k] = coefficient*exp(-r/L)/r;
            } /* end if */
} /* end mask_create */


/***************************************
 fucntion read_array (array)

This function reads data from stdin and loads it into an array.

***************************************/

void read_array (array)
float array [x_array][y_array];
{
int x_index, y_index;

   for (x_index=0; x_index <=x_max; x_index++)
      for (y_index=0; y_index<= y_max; y_index ++)
         array[x_index][y_index]= 0.0;

   scanf ("%*f %*f");   /* this line skips the data line containing loaction
information */
   for (x_index =0; x_index<= x_max ; x_index ++)
     {
         scanf ("%f", &array[x_index][0]);
        /* printf ("%d %f \n", x_index, array[x_index][0]); */
     }
}

/***************************************
  function output_array (array)

This function outputs the contents of array.

***************************************/

void output_array (array)
float array[x_array][y_array];
{
int x_index, y_index;

   for (x_index =0; x_index<=x_max; x_index++)
   {
      for (y_index =0; y_index<= y_max; y_index++)
         printf ("%f ", array[x_index][y_index]);
      printf ("\n");
   }
   printf ("\n");
}
```

```
/***************************************
  function interpolate_array (array)

This function interpolates the one-dimensional array
into a two-dimensional array using the circular symmetry
of the problem studied.

****************************************/
void interpolate_array (array)
float array[x_array][y_array];
{
int x_index, y_index, integer;
float distance, fraction;

   for (x_index = 0; x_index <= x_max; x_index++)
      for (y_index = 1; y_index <= y_max; y_index ++)
      {
         distance =sqrt ( (float) (x_index*x_index + y_index*y_index));
         if ((int) (distance+0.5) < x_max)
           {
             integer = (int) distance;
             fraction = distance - (float) integer;
             array[x_index][y_index] = array[integer][0] +
fraction*(array[integer +1][0] - array[integer][0]);
           }  /* end if */
      } /* end for y_index */
} /* end interpolate_array */


/***************************************
  function power_scale (array);

This function calculates the power of the lateral plane and
calculates a conversion factor for the desired power output.

****************************************/

float power_scale (array, z)
float array[x_array][y_array];
float z;
{
float sum =0.0;
float power, scale, total_attenuation;
int x_index, y_index;

   for ( y_index =1; y_index <= y_max; y_index++)
      for (x_index =1; x_index <= x_max; x_index ++)
         sum += 4.0*array[x_index][y_index];

   for (x_index = 1; x_index <= x_max; x_index ++)
      sum += 2.0* array[x_index][0];
   for (y_index =1 ; y_index <= y_max; y_index ++)
      sum += 2.0* array[0][y_index];
   sum += array[0][0];

   power = sum * space_step * space_step;

   if (z<d1)  total_attenuation = -2.0*A1*z;
```

```
        else if (z<d2) total_attenuation = -2.0*(A1*d1 + A2*(z-d1));
             else total_attenuation = -2.0*(A1*d1 + A2*d2 + A3*(z-d2));

     scale = (transducer_power*exp(total_attenuation))/power;
     printf ("%f %f %f %f %f\n", z, sum, scale, array[0][0],scale*array[0][0]);
     return (scale);
  }

  /*************************************
    function temperature

  This function uses the mask and the values
  of intensity to superimpose the effect of points
  in space on the temperature increase along the
  beam axis.

  **************************************/
  void temperature (intensity, scale, z)
  float intensity[x_array][y_array];
  float scale;
  float z;
  {
  int x_index, y_index, z_index, k;
  float scale2;

     z_index = (int) (z/space_step +0.5);
     for (y_index =0; y_index <= mask_size_lateral; y_index ++)
        for (x_index =0 ;x_index <=mask_size_lateral; x_index ++)
        {
           scale2 = 2.0;
           if ((x_index !=0) && (y_index !=0) ) scale2 = 4.0;
           if ((x_index) ==0 && (y_index == 0) ) scale2 = 1.0;

           for (k = -mask_size_axial; k<= mask_size_axial; k++)
           if ((z_index + k) > 0)
              temp[z_index+k] +=
  scale*scale2*intensity[x_index][y_index]*mask[x_index][y_index][abs(k)];
        }

  } /* end temperature */


  main ()
  {
  int x_index, y_index,
      axis_start_step, axis_stop_step, z_step, axis_d1_step, axis_d2_step;
  float scale, z;
  float intensity [x_array][y_array];

     read_header_data ();
     mask_size_lateral= 150;
     mask_size_axial=250;
     printf ("mask size lateral= %d  axial =%d\n", mask_size_lateral,
  mask_size_axial);
     mask_create (A1);
     printf ("mask created\n");

     for (x_index=0; x_index <=x_max; x_index++)
```

```
         for (y_index=0; y_index<= y_max; y_index ++)
            intensity[x_index][y_index]= 0.0;

      axis_start_step = (int) (axis_start/space_step +0.5);
      axis_d1_step = (int) (d1/space_step +0.5);
      axis_d2_step = (int) (d2/space_step +0.5);
      axis_stop_step = (int) (axis_stop/space_step +0.5);

      for (z_step = axis_start_step; z_step < axis_d1_step; z_step ++)
      {
         z = z_step*space_step;


         read_array (intensity);
         interpolate_array (intensity);
         scale = power_scale (intensity, z);
         temperature (intensity, scale, z);
      }

   /* skip data between d1 and d2 as attenuation = 0  */

      for (z_step = axis_d1_step; z_step < axis_d2_step; z_step ++)
         read_array(intensity);

   /* recalculate mask for the 3rd layer  */
      mask_create (A3);


      for (z_step = axis_d2_step; z_step <= axis_stop_step; z_step ++)
      {
         z = z_step*space_step;

         read_array (intensity);
         interpolate_array (intensity);
         scale = power_scale (intensity, z);
         temperature (intensity, scale, z);
      }

      for (z_step = axis_start_step; z_step < axis_stop_step; z_step ++)
      {
         z = z_step*space_step;
         printf ("%f %f\n", z, temp[z_step]);
      }
   }
```

APPENDIX D

FETAL TISSUE DATA

The following figures report the results of the application of the general solution to the fetal imaging case of Section 5.4. Figures D.1-D.3 show the results for the homogeneous case. Figures D.4 - D.6 are for Case 1, an abdominal wall of 0.1 cm. Figures D.7 - D.9 are for Case 2, an abdominal wall of 1.0 cm. Finally Figures D.10 - D.12 are for Case 3, an abdominal wall of 2.0 cm.

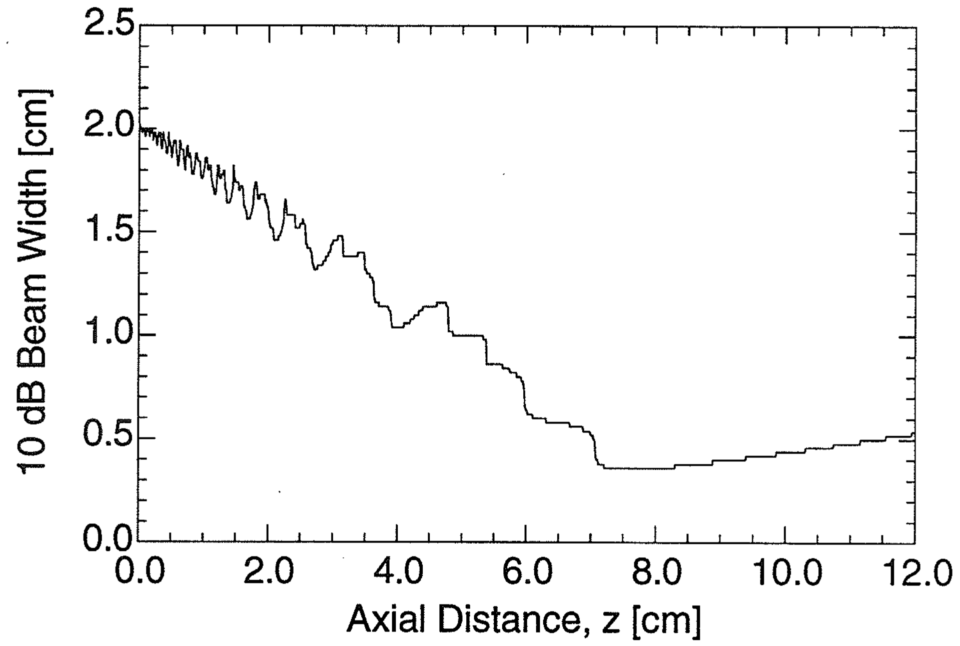

Figure D.1: The 10 dB beam width calculated from the general solution for the homogeneous fetal tissue case (see Tables 4.1 and 4.2. A= 0.3 dB/cm-MHz).

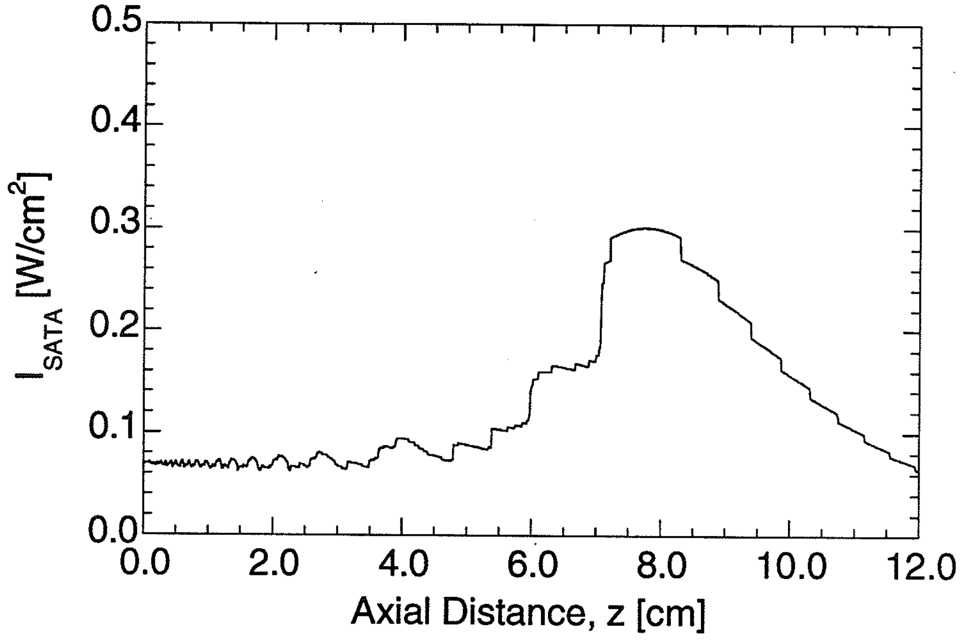

Figure D.2: The spatial average temporal average intensity calculated from the general solution for homogeneous fetal tissue case (see Tables 4.1 and 4.2. A = 0.3 dB/cm-MHz).

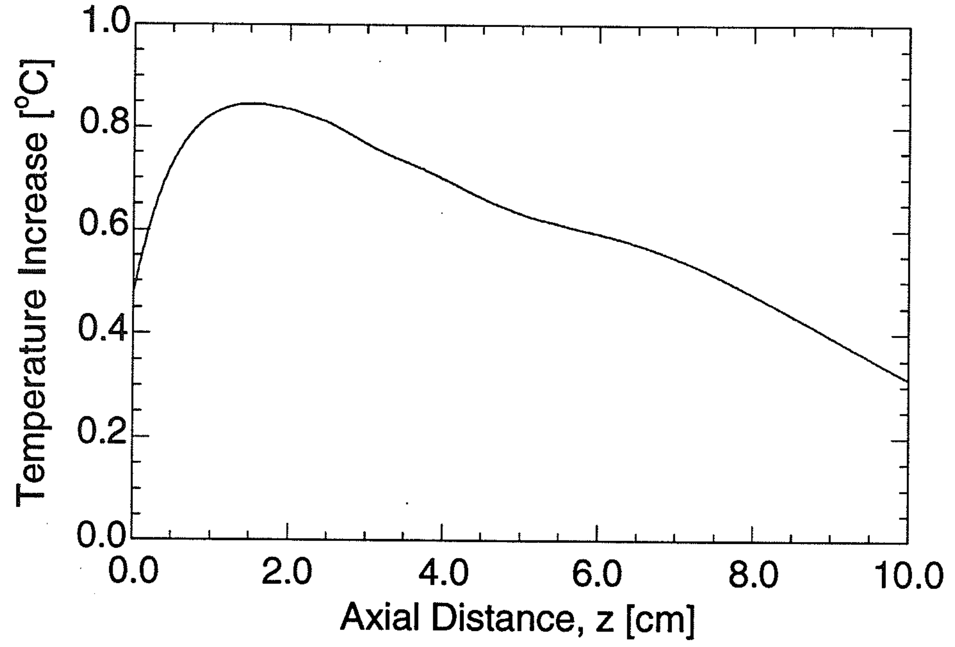

Figure D.3: The axial temperature increase calculated from the general solution for the homogeneous fetal tissue case (see Tables 4.1 and 4.2. A = 0.3 dB/cm-MHz)

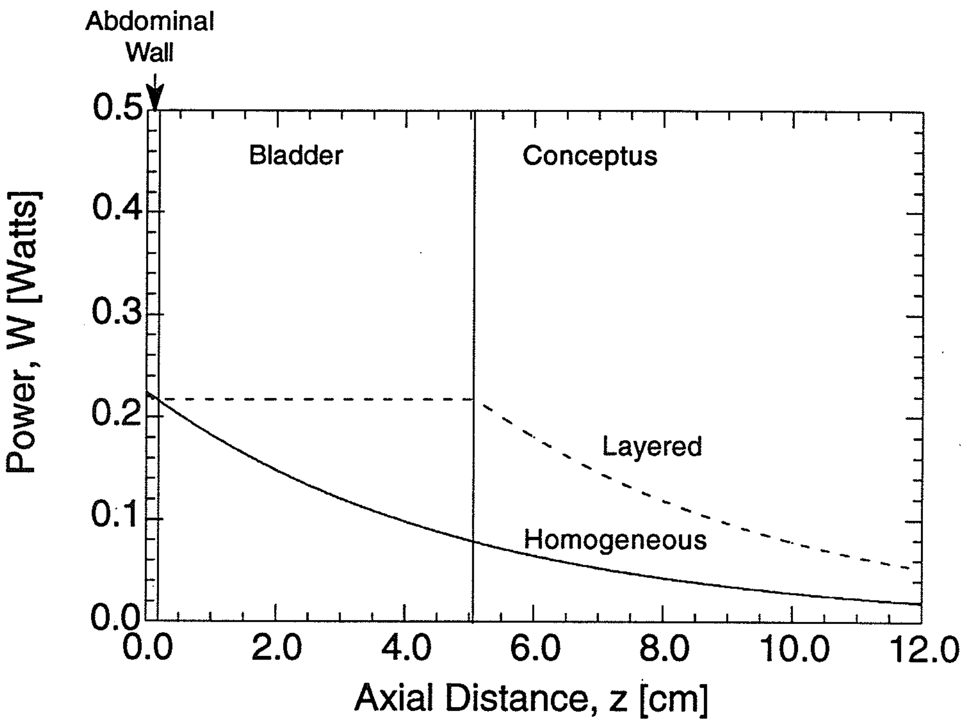

Figure D.4: The axial power distribution for Case 1 (see Tables 4.1, 4.2 and 5.1). Also shown is the axial power distribution for the homogeneous fetal tissue case.

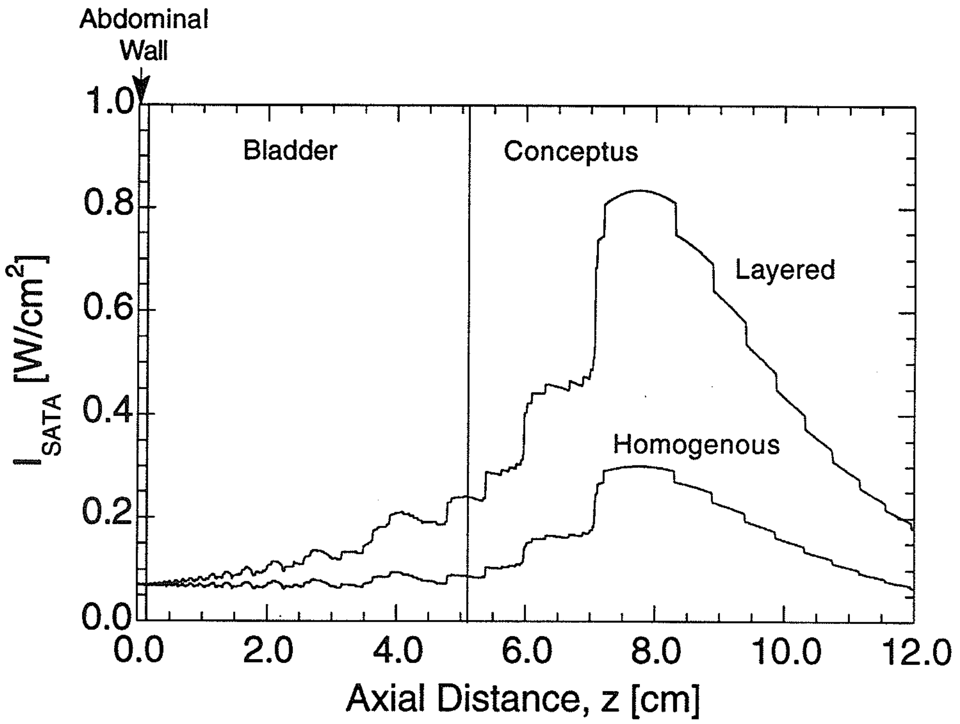

Figure D.5: The spatial average temporal average intensity calculated from the general solution for Case 1 (see Tables 4.1, 4.2, 5.1 and 5.2). Also shown in the spatial average temporal average intensity for the homogeneous fetal tissue case.

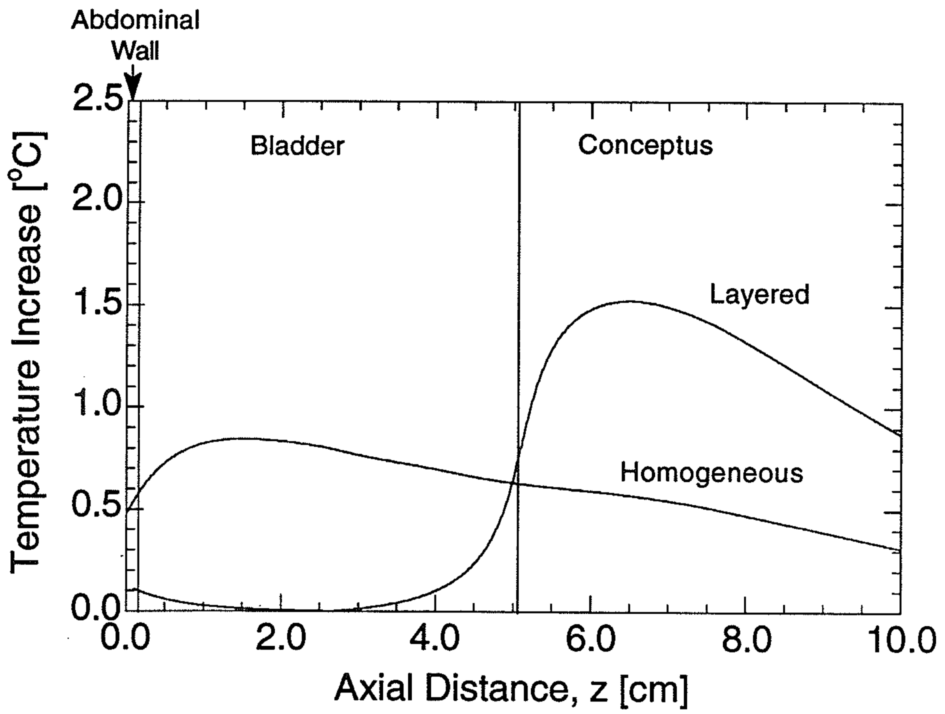

Figure D.6: The axial temperature increase calculated from the general solution for Case 1 conditions (see Tables 4.1, 4.2, 5.1 and 5.2). Also shown is the axial temperature increase for the homogeneous fetal tissue case.

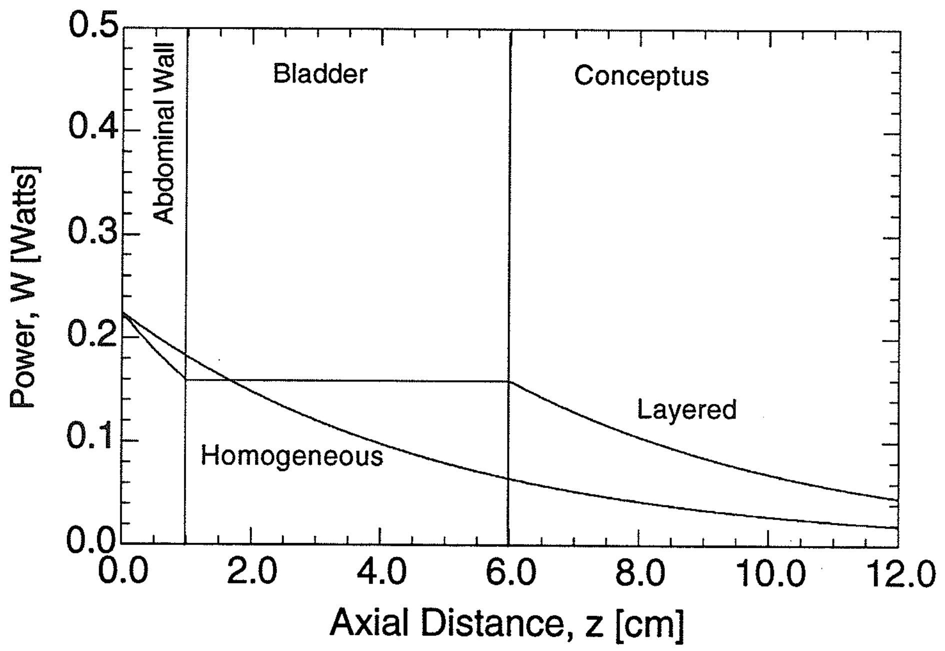

Figure D.7: The axial power distribution for Case 2 conditions (see Table 5.1). Also shown is the axial power distribution for the homogeneous fetal tissue case.

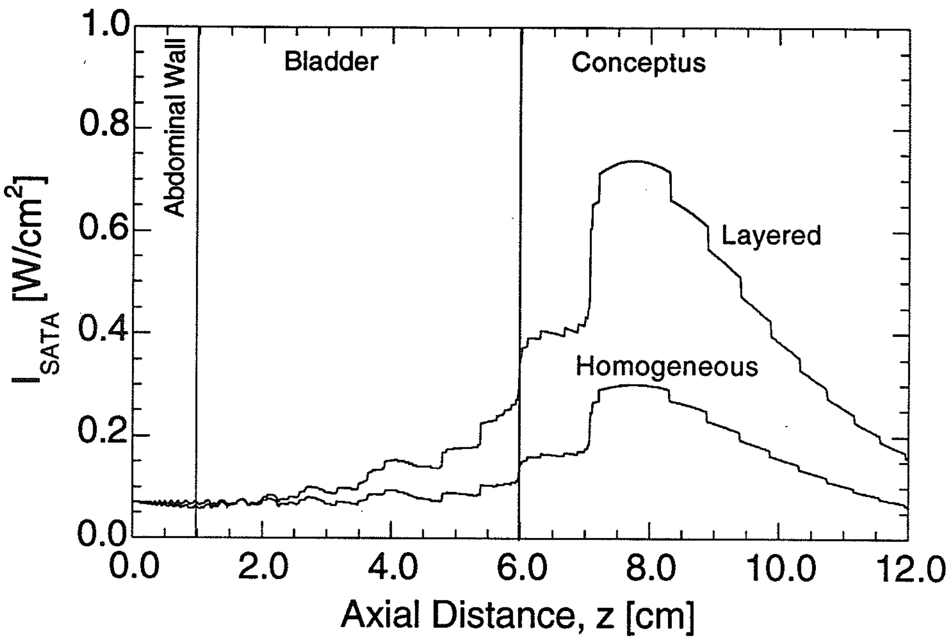

Figure D.8: The spatial average temporal average intensity calculated from the general solution for Case 2 conditions (see Tables 5.1 and 5.2). Also shown is the spatial average temporal average intensity for the homogeneous tissue case.

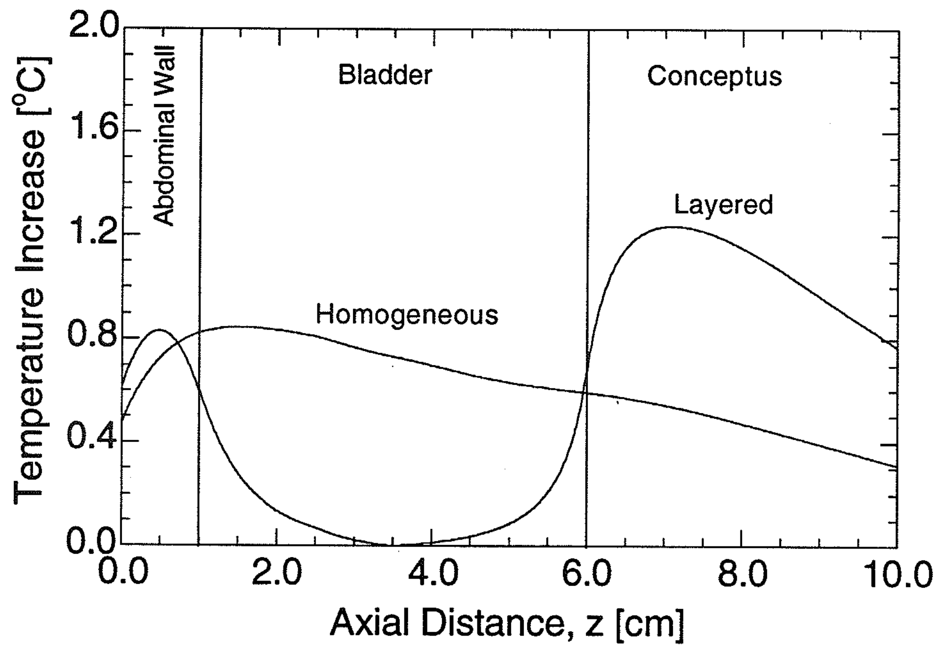

Figure D.9: The temperature increase calculated from the general solution for Case 2 conditions (see Tables 5.1 and 5.2). Also shown is the temperature increase for the homogeneous tissue case.

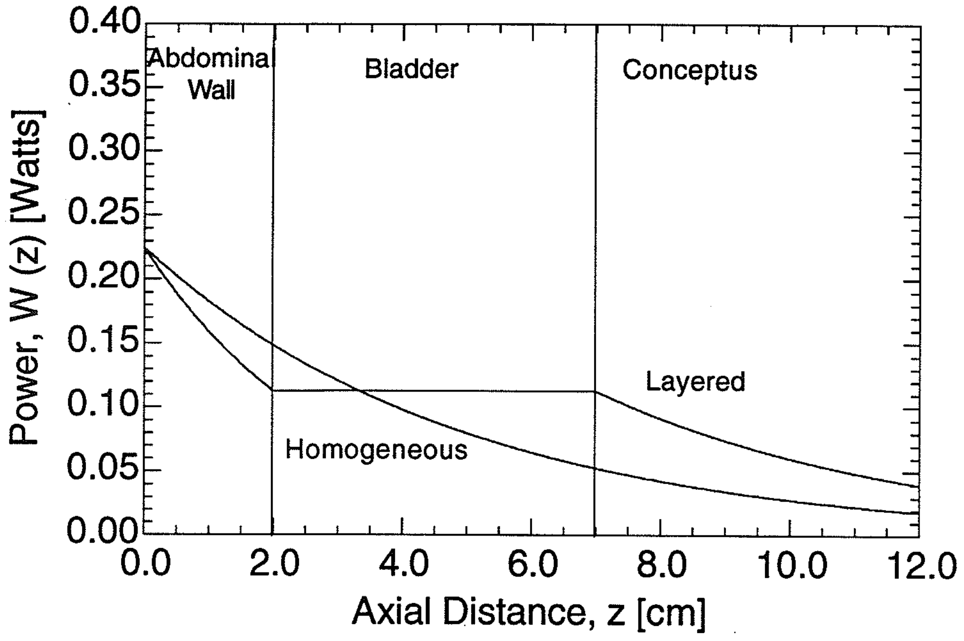

Figure D.10: The axial power distribution for Case 3 conditions (see Tables 4.1, 4.2 and 5.1). Also shown is the axial power distribution for the homogeneous fetal tissue case.

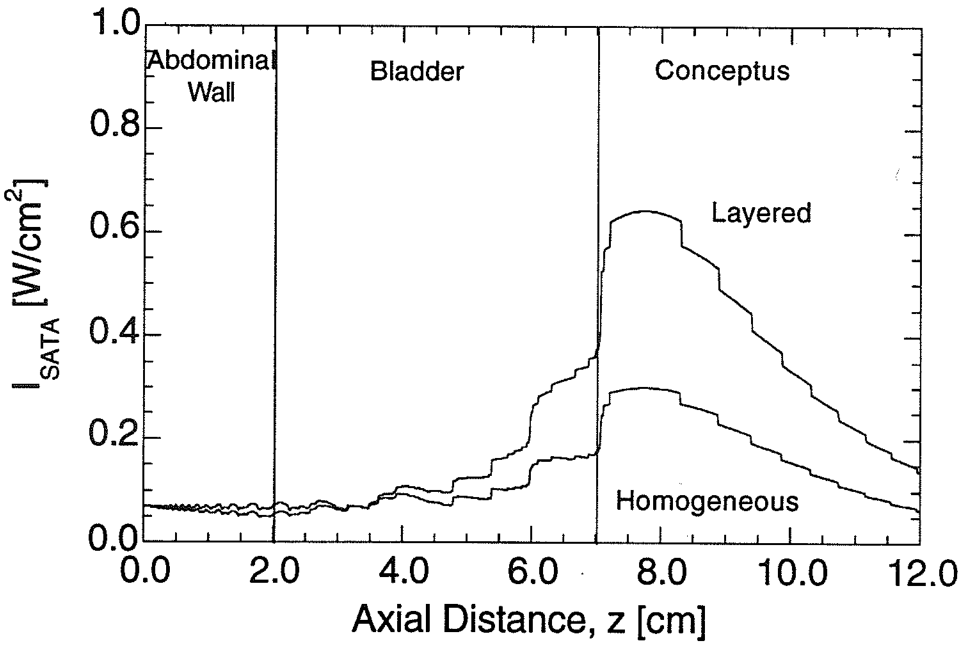

Figure D.11: The spatial average temporal average intensity calculated from the general solution for Case 3 conditions (see Tables 4.1, 4.2, 5.1 and 5.2). Also shown is the spatial average temporal average intensity for the homogeneous fetal tissue case.

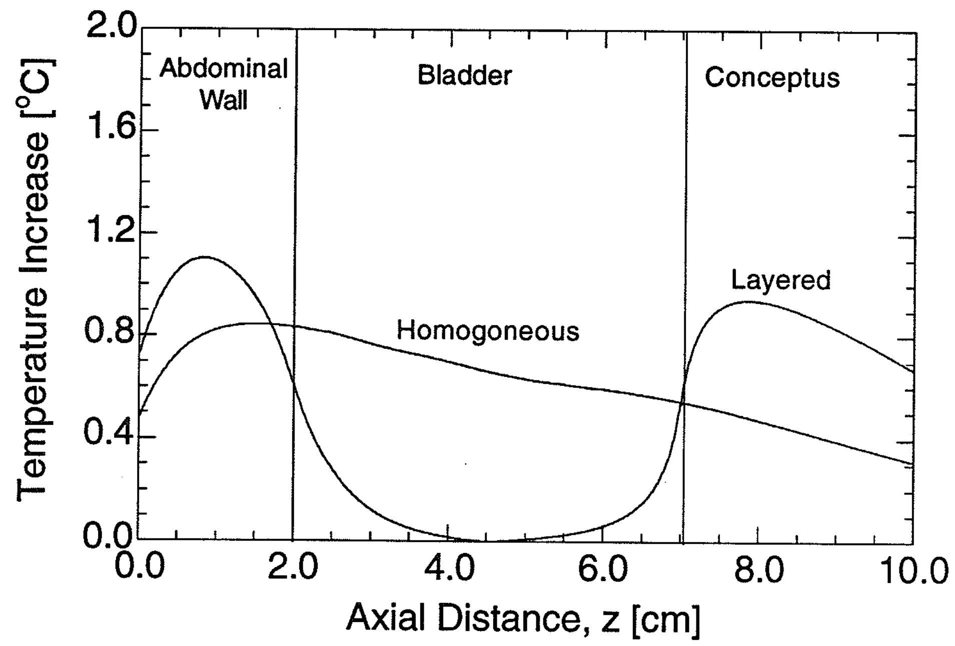

Figure D.12: The temperature increase calculated from the general solution for Case 3 conditions (see Tables 4.1, 4.2, 5.1, and 5.2). Also shown is the temperature increase for the homogeneous fetal tissue case.

APPENDIX E

EYE IMAGING DATA

The following figures report the results of the application of the general solution to the small part and eye imaging case of Section 5.5. Figures E.1-E.3 show the results for the homogeneous case for the transducer of Case 4. Figures E.4 - E.6 present the results for the layered tissue model of Case 4. Figures E.7 - E.9 represent the results for the homogeneous tissue case using the transducer of Case 5. Finally, Figures E.10 - E.12 present the results of the general solution for the layered tissue model using the transducer of Case 5.

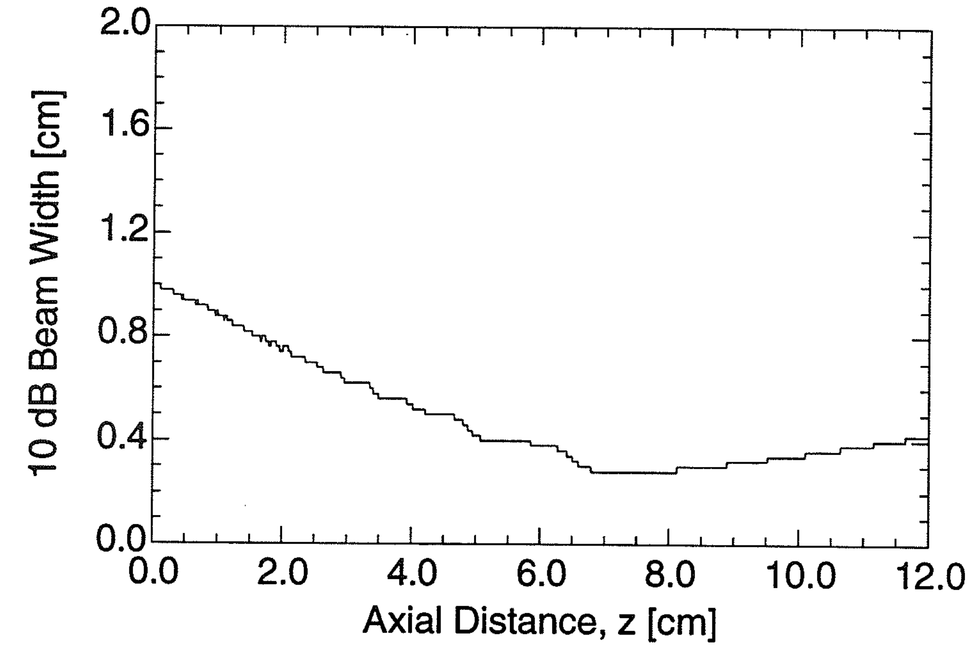

Figure E.1: The 10 dB beam width calculated from the general solution for the homogeneous fetal tissue case and the transducer of Case 4 (see Tables 4.1, 4.2 and 5.4).

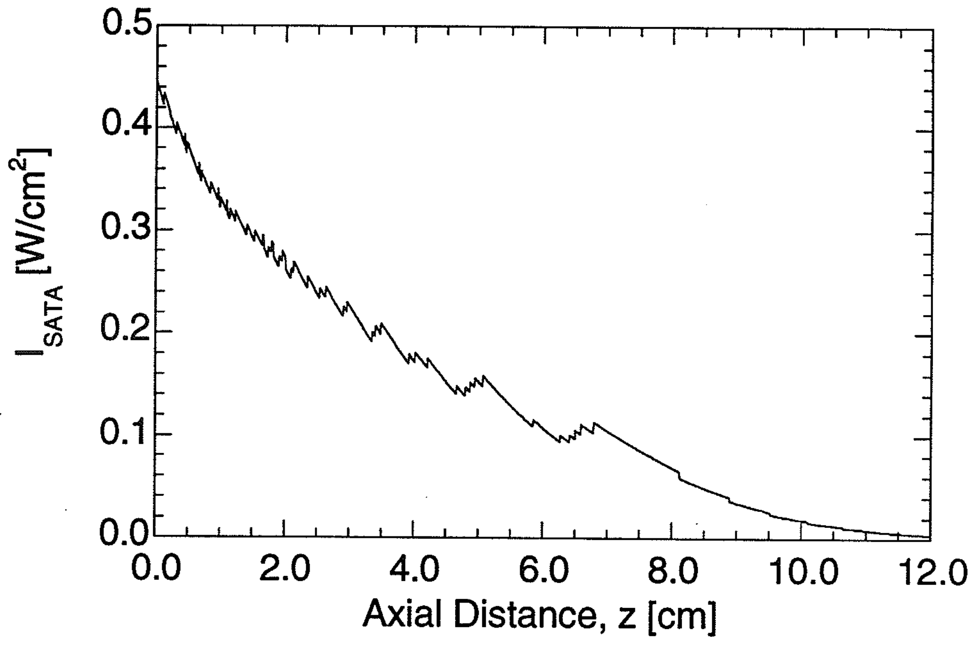

Figure E.2: The spatial average temporal average intensity calculated from the general solution for the homogeneous tissue case and the transducer of Case 4 (see Tables 4.1, 4.2 and 5.4).

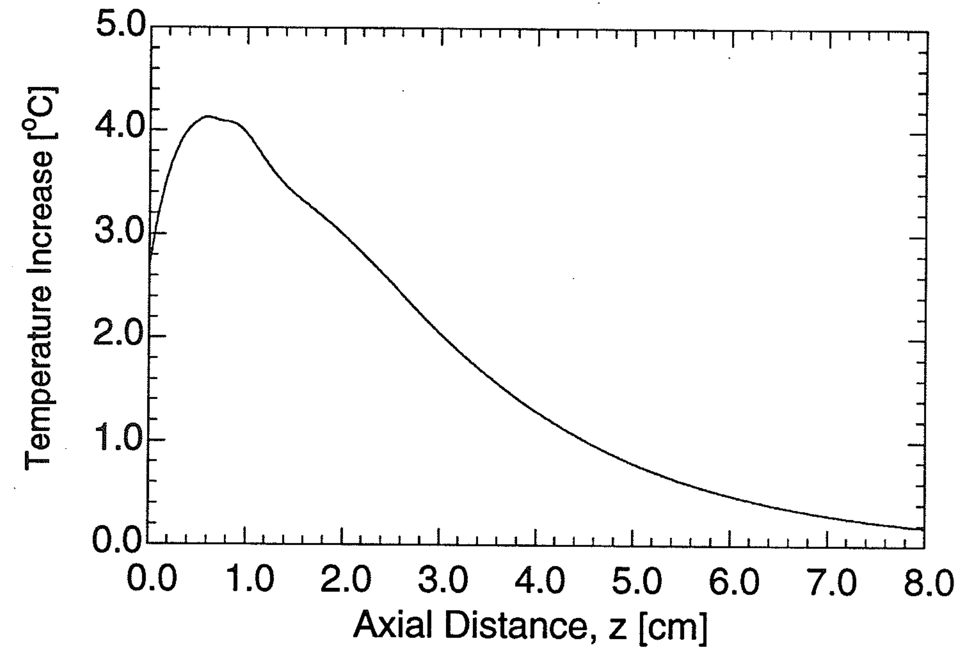

Figure E.3: The temperature increase calculated from the general solution for the homogeneous tissue case and the transducer of Case 4 (see Tables 4.1, 4.2, and 5.4).

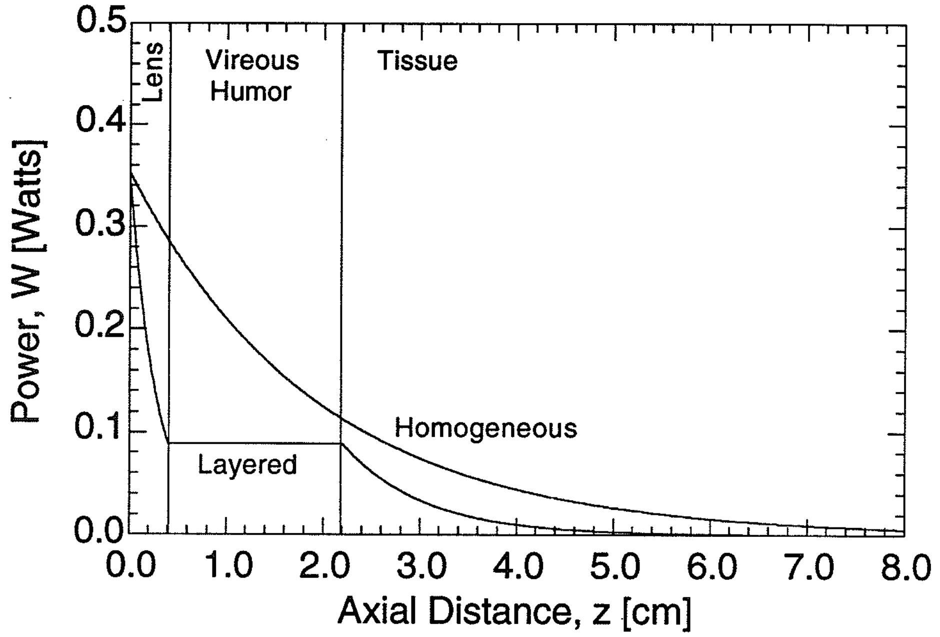

Figure E.4: The axial power distribution for Case 4 (see Tables 4.1, 4.2, 5.3 and 5.4). Also plotted is the axial power distribution for the homogeneous tissue case.

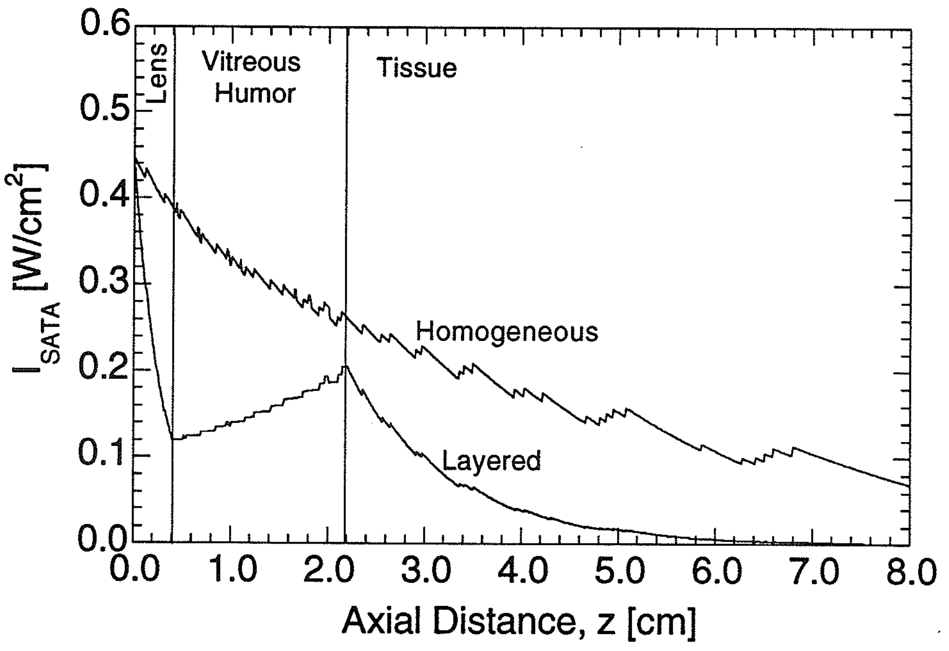

Figure E.5: The spatial average temporal average intensity calculated from the general solution for Case 4 (see Tables 4.1, 4.2, 5.3, 5.4 and 5.5). Also shown is the spatial average temporal average intensity for the homogeneous tissue case.

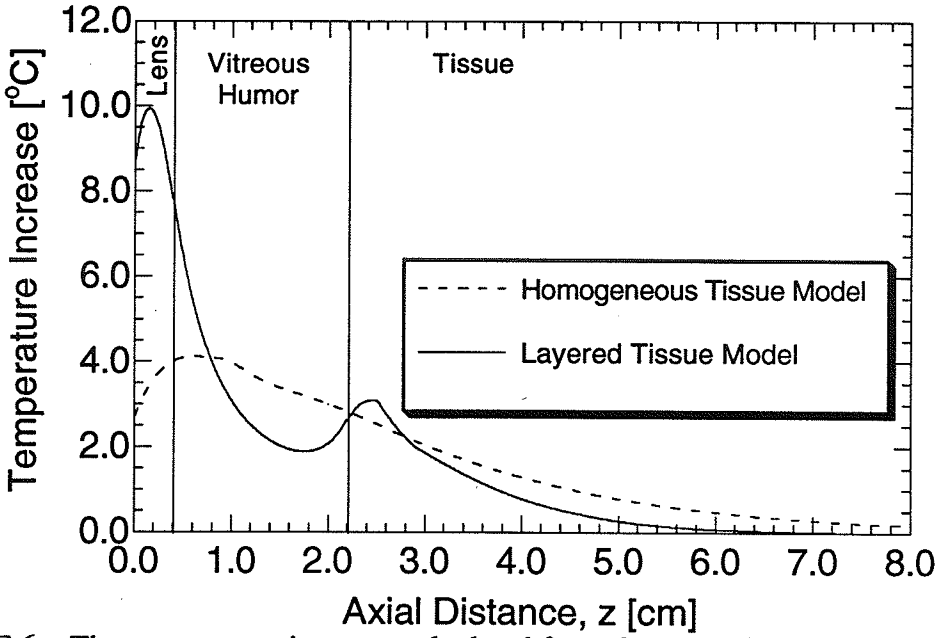

Figure E.6: The temperature increase calculated from the general solution for Case 4 (see Tables 4.1, 4.2, 5.3, 5.4, and 5.5). Also shown is the temperature increase for the homogeneous tissue case.

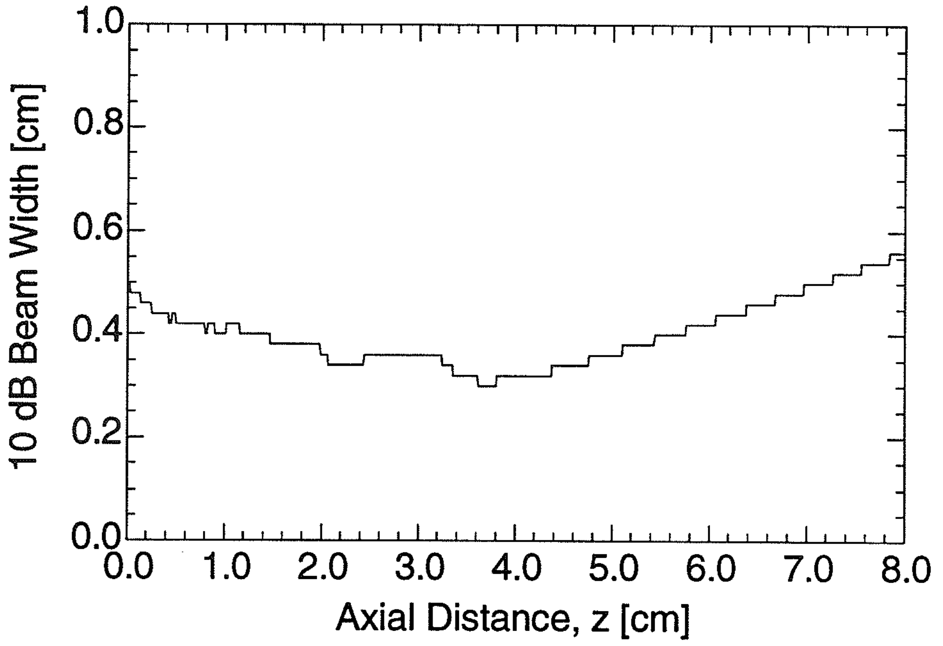

Figure E.7: The 10 dB beam width calculated from the general solution for the homogeneous tissue case and the transducer of Case 5 (see Tables 4.1,4.2 and 5.4).

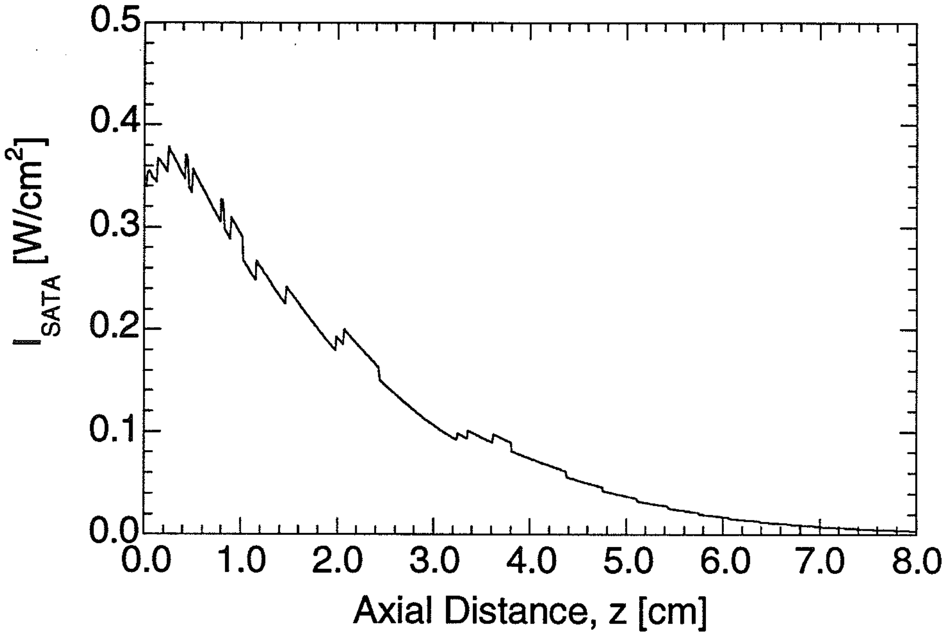

Figure E.8: The spatial average temporal average intensity calculated from the general solution for the homogeneous tissue case and the transducer of Case 5 (see Tables 4.1,4.2, and 5.4).

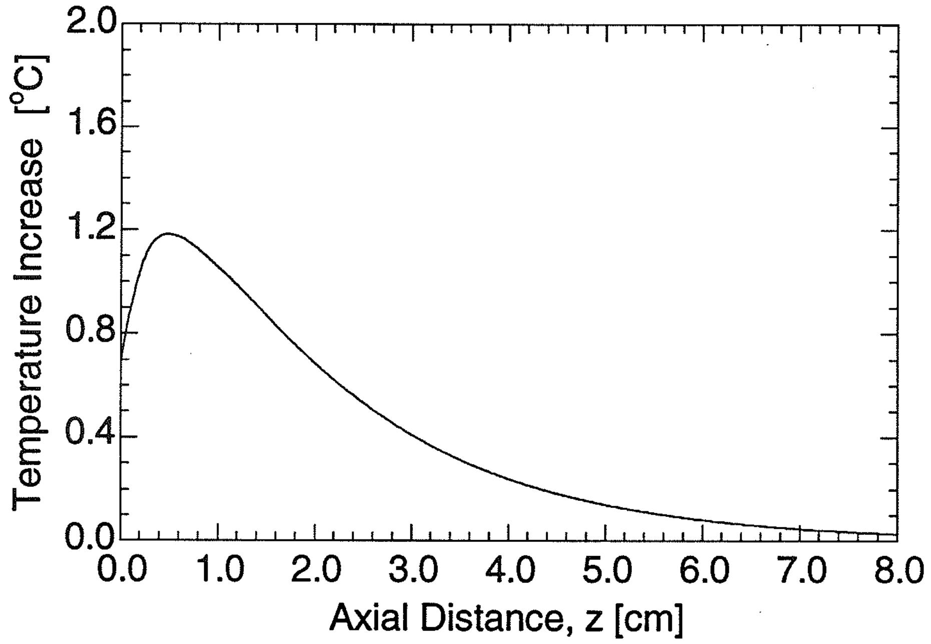

Figure E.9: The temperature increase calculated from the general solution for the homogeneous tissue case and the transducer of Case 5 (see Tables 4.1, 4.2, and 5.4).

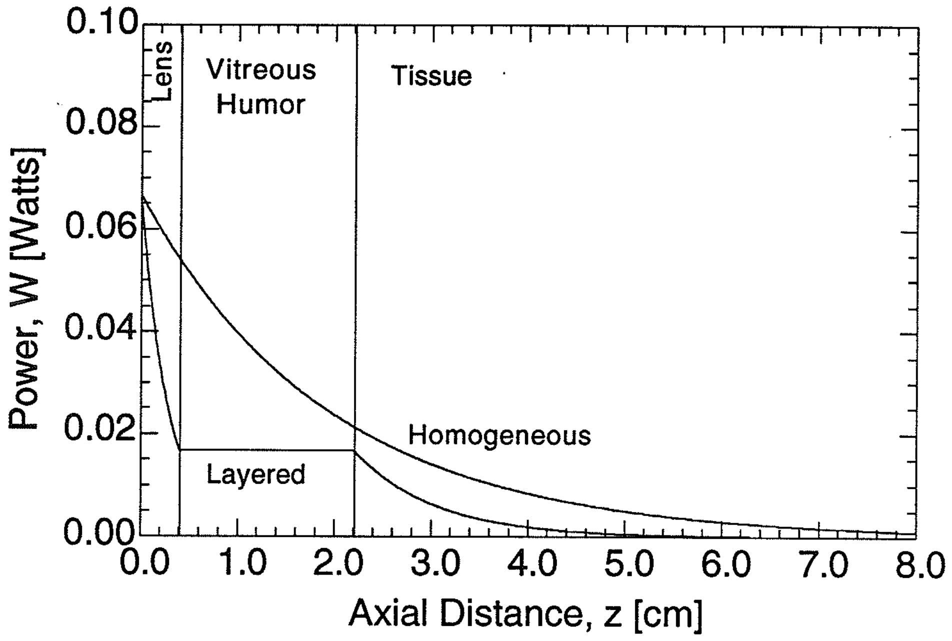

Figure E.10: The axial power distribution for Case 5 (see Tables 4.1, 4.2, and 5.4). Also shown is the axial power distribution for the homogeneous tissue case.

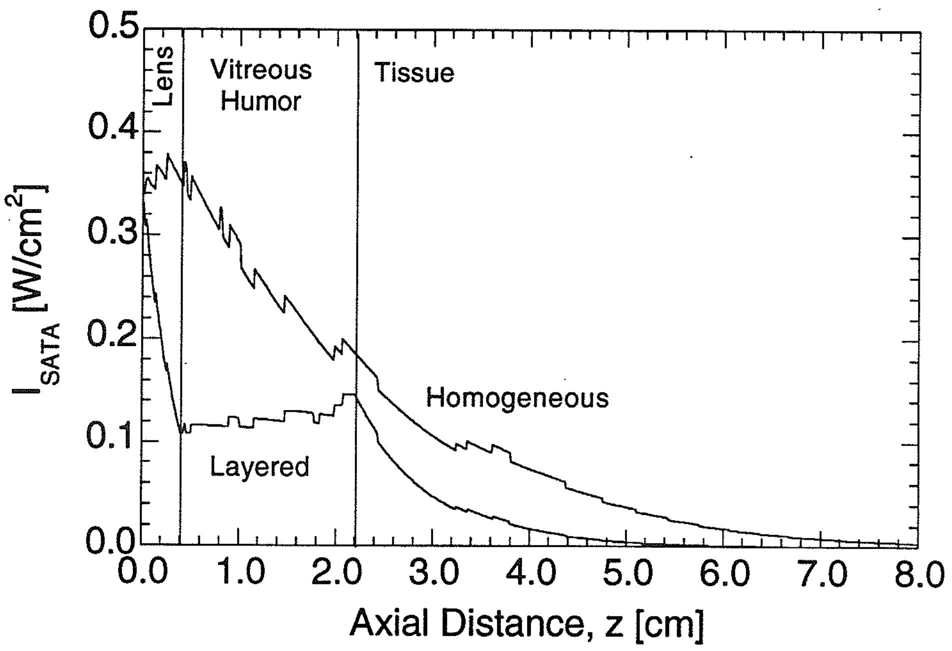

Figure E.11: The spatial average temporal average intensity calculated from the general solution for Case 5 (see Tables 4.1,4.2,5.3,5.4, and 5.5). Also shown is the spatial average temporal average intensity for the homogeneous case.

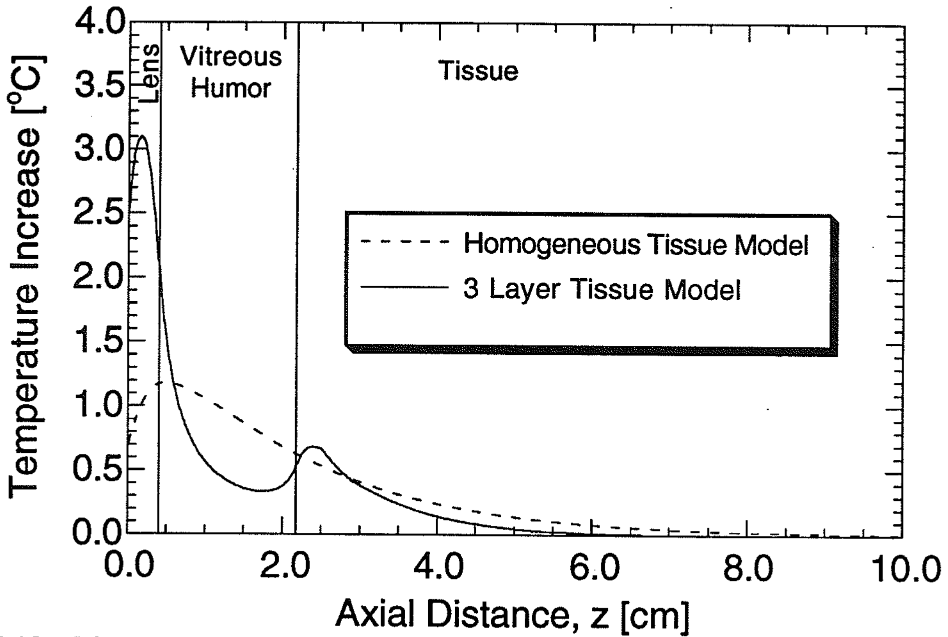

Figure E.12: The temperature increase calculated from the general solution for Case 5 (see Tables 4.1, 4.2, 5.3, 5.4, and 5.5). Also shown is the temperature increase for the homogeneous tissue case.