Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

A THERMOELASTIC MODEL APPLIED TO STRESS CONTROL IN LASER HEATING OF CERAMICS

by
Jeff Alexander Wagner

Localized laser heating is widely used in materials processing. In extending these techniques to materials with relatively low thermal conductivities and ductilities such as ceramics and glasses, existing methods must be modified to control the high thermal stresses which are associated with the localized heating of these materials. Thermal profiles must be designed to minimize damage to regions adjacent to the processed area. To achieve this with single beam sources the power and radius can be varied in time, or the beam can be moved across the surface in a programmed pattern to achieve the desired thermal profile.

In this work the thermoelastic effects associated with fixed and moving beam sources are examined in light of the application described above. Finite difference models of the temperature rise and resulting stresses and strains for the surface heating of a semi-infinite half-space are presented. These simulations are then compared to experimental results obtained with a CO_2 laser aimed with computer controlled optics.

A THERMOELASTIC MODEL APPLIED TO STRESS CONTROL IN LASER HEATING OF CERAMICS

by
Jeff Alexander Wagner

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Applied Physics

Department of Physics

May 1994

APPROVAL PAGE

A THERMOELASTIC MODEL APPLIED TO STRESS CONTROL IN LASER HEATING OF CERAMICS

Jeff Alexander Wagner

Dr. Daniel E. Murnick, Thesis Advisor — Date
Professor of Physics, Rutgers University

Dr. Philip R. Goode, Committee Member — Date
Professor of Physics, NJIT

Dr. Yuan Li, Committee Member — Date
Associate Professor of Physics, Rutgers University

BIOGRAPHICAL SKETCH

Author: Jeff Alexander Wagner

Degree: Master of Science in Applied Physics

Date: May 1994

Undergraduate and Graduate Education:

- Master of Science in Applied Physics,
  New Jersey Institute of Technology,
  Newark, New Jersey, 1994

- Bachelor of Science in Applied Physics,
  New Jersey Institute of Technology, Newark, New Jersey
  and Rutgers University, Newark, New Jersey, 1992

Major: Applied Physics

ACKNOWLEDGMENT

The author gratefully acknowledges the support and guidance of many faculty members in the physics departments of the New Jersey Institute of Technology and Rutgers University, both in the execution of this research and in the undergraduate and graduate programs in general. Specifically, to Dr. Daniel Murnick, for his encouragement and direction in the role of thesis advisor the author extends his sincere gratitude.

The author would also like to thank American Standard Inc. for funding and technical assistance. An award from the NJIT Alumni Association during the course of this work was also much appreciated.

Finally, the author wishes to thank committee members Dr. Philip Goode and Dr. Yuan Li.

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a ..... absorption coeficient

b_i ..... source vector

C ..... specific heat

E ..... Young's modulus

F_0 ..... laser power density on beam axis

F_i ..... body forces

G ..... shear modulus

i, j, k ..... discretization subscripts

K ..... thermal conductivity

P ..... laser power

q ..... heat flux, energy flux

R ..... surface reflectivity

r_0 ..... beam radius

T ..... temperature

t ..... time

u_x, u_y, u_x ... spatial displacements

x, y, z ..... spatial coordinates

α ..... coeficient of linear expansion

γ ..... shear strains

ε_i ..... normal strains

ε ..... emmisivity

LIST OF SYMBOLS
(continued)

κ ..... thermal diffusivity

λ ..... Lamé's constant

ν ..... Poisson's ratio

ρ ..... density (Chapter 2)

ρ ..... beam step size ratio (Chatpter 3)

σ ..... normal stresses

τ ..... shear stresses

CHAPTER 1

INTRODUCTION

1.1 Introduction

Localized laser heating is widely used in materials processing. Laser welding and machining in metal forming applications, and laser induced or assisted surface modification in semiconductor manufacture are common examples (1, 2). Laser heating is also used in surface sealing of ceramics and in the surface treatment of glasses (3, 4, 5). In applying these techniques to materials with relatively low thermal conductivities such as ceramics and glasses, a stategy must be adopted to control the high thermal stresses which are associated with the localized heating of these materials. While stresses in the target region may be relieved through plastic flow, adjacent areas will undergo large temperature changes while the material acts in the elastic regime. These stresses can easily exceed the elastic limits of the material leading to cracking or other undesired effects. In the surface sealing application stress cracking is controlled by injecting chemical modifiers into the processed area (3, 15). This technique is applicable in part because the entire surface is being treated. In some applications, such as the cosmetic repair of localized surface defects in glazed ceramics, these modifiers must be limited to those which allow the refired region to blend in visually with the surrounding material.

Another possible method for lowering these stresses is to supply additional heat to the peripheral areas during some part of the process to allow a partial annealing to take

place. To achieve this with single beam sources the power and radius can be varied in time, or the beam can be moved across the surface in a programmed pattern to achieve the desired thermal profile.

In this work the thermoelastic effects associated with localized laser heating are examined in the context of spot refiring of glazed porcelain surfaces. The heat source is taken to be a single Gaussian spot typical of TEM_{00} mode laser operation. Numeric models of the temperature rise and resulting stresses and strains for the surface heating of a semi-infinite half-space are presented. These models are compared with analytic solutions for the simple case of a fixed position Gaussian profile heat source. The models are then applied to more complex heating programs and compared to experimental results.

The selection of thermal and elastic material constants used in the simulations are discussed in Chapter 2. In Chapter 3 a numeric model for the temperature rise due to laser heating is presented. The temperature field calculated here is used as input for the thermoelastic displacement model. Chapter 4 is a description of the numeric model for thermoelastic displacements, stresses and strains. Also included here is a comparison to available analytic solutions. The models are then applied to the case of a fixed position source, a fixed position source where the beam radius is varied in time, and finally, a moving source. In Chapter 5 the model predictions are compared with experimental results using a CO_2 laser and glazed porcelain targets. Conclusions and suggestions concerning possible further work are given in Chapter 6. For completeness, source code for the numeric models and a brief description of program organization are included in the appendix.

1.2 History

There are few references available which specifically address the laser spot refiring of glazed porcelain. However, important components of the problem have been investigated in analogous situations.

Analytic treatments of the laser heating of solids have been presented by several authors. An integral expression for the temperature profile due to surface heating with a Gaussian beam source, and a closed form for the temperature on the beam axis at the surface has been presented by Ready (6) and Duley (1). Ultimate temperature rise characterized in terms of ratio of beam radius to absorption depth is given by Lax (7). In these cases radiation from the heated surface is not considered.

In Bentini et. al. (8) the thermal stresses resulting from surface heating with a strip heater along with a radiative boundary condition were estimated analytically under the assumption that the normal stress components could be considered independently . Welsh et. al. (2) extend this work to fixed Gaussian sources and give the stresses as analytic expressions which require a single numeric integration . These semi-analytic results are compared with results from the stress-strain model presented here for the simple case of a fixed Gaussian heat source.

The mechanical properties of ceramics and glasses treated with CO_2 laser heating are discussed by Petitbon et. al. (3), Yi and Strutt (4), and by Glasser and Jing (9) among others.

Dallaire and Cielo (10) specifically address the thermal aspects of the laser spot refiring of glazed porcelain, but do not include a thermoelastic stress analysis .

CHAPTER 2

TARGET MATERIALS AND CONSTANTS

2.1 Target Materials

In this study, the application of interest was the repair of surface defects in glazed porcelain. These defects typically consist of bubbles or small voids left after impurities are removed by mechanical means. The defects are filled with a paste of fritted glass and fired with a CO_2 laser which melts the filler and fuses it to the surrounding material. Details of the filler composition are beyond the scope of the present paper.

Due to the similarity in thermal properties between the glaze and body, as a first approximation the structure is modeled as a single homogeneous material. After fusion, filler materials are also considered to have similar properties to the surrounding material. It is further assumed that heating program design does not depend on the detailed nature of the boundary between the filler and surrounding materials, but will take into consideration the gross size of the defect. This is a necessary simplification since the defects are somewhat random in shape. Target thickness is much greater than defect depth, and will not be considered. Thus the region modeled will be a uniform half-space.

2.2 Material Constants

2.2.1 Optical and Thermal Properties

In this and in the following section, the selection of values for the various relevant material constants is addressed. The values chosen were either determined experimentally and

compared with published values, or taken from the literature. Where a range of values appears in the literature a typical value was chosen.

The temperature model presented in Chapter 3 requires that the following material constants be specified as input: surface reflectivity, R; thermal conductivity, K; and the thermal diffusivity, κ.

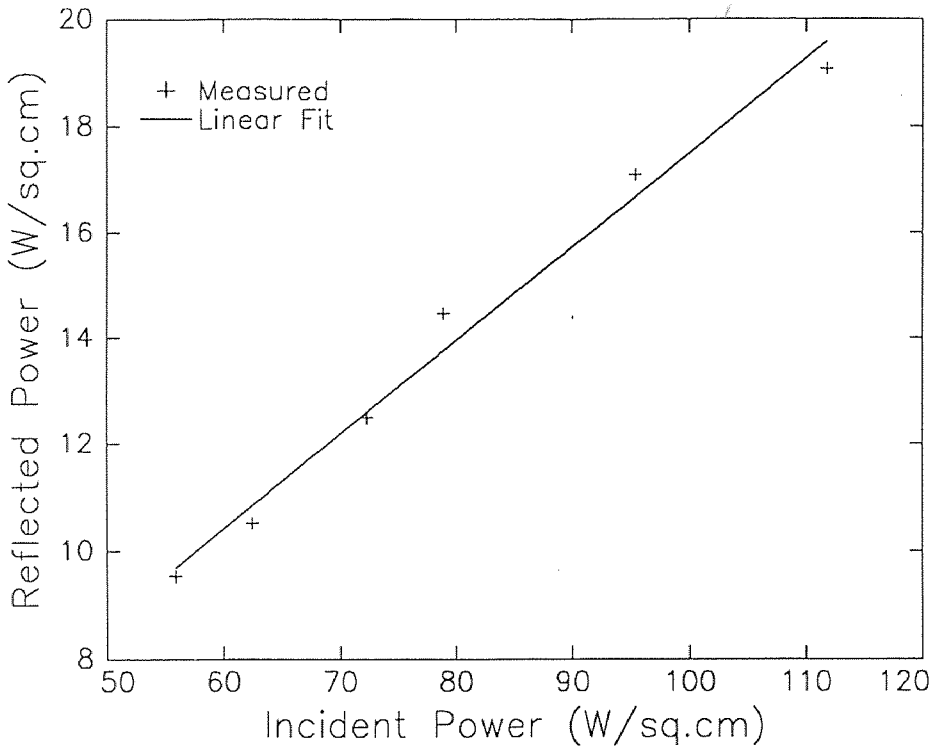


Figure 2.1 Reflected versus incident power density for 10.6μ laser radiation incident on a glazed ceramic target.

In these experiments the target was heated with the incident beam at or near normal incidence. The surface reflectivity near normal incidence (15° from perpendicular) was taken to be the ratio of reflected to incident power as measured by a laser power meter. Due to the glassy nature of the target surface the reflected beam was sharply defined, so it is reasonable to assume that all of the reflected power was collected by the

meter. The measurements for various incident power densities and a value for R, taken from the slope of a linear fit, are shown in Fig. 2.1 and Table 2.1. This value is consistent with that measured by Dallaire and Cielo (10) for a similar target material .

Table 2.1 Material constants used in the simulations.

Reflectivity, R	0.18
Thermal conductivity, K	2.82 W/(m°C)
Thermal diffusivity, κ	1.89 m^2/s
Coefficient of linear expansion, α	4.7×10^{-6}/°C
Young's modulus, E	1.03×10^{11} Pa
Poisson's ratio, ν	0.25
Compressive stress limit	4.14×10^{8} Pa
Tensile stress limit	8.96×10^{7} Pa

The thermal constants were also determined experimentally by comparing temperature data obtained using a time and space resolved optical pyrometer with an analytic expression. Ready (1) gives the following relation for axisymmetric temperature rise due to a instantaneous Gaussian heat source incident on the surface of an infinite half space:

$$T(r=0, z=0, t) = \frac{\varepsilon F_0 r_0}{K\sqrt{\pi}} \tan^{-1}\left(\frac{\sqrt{4\kappa t}}{r_0}\right) \tag{2.1}$$

where

$$F_0 = \frac{P}{\pi r_0^2}.$$

P is the total power incident on the target, r_0 is the beam radius, and ε is the emissivity, equal to 1 - R for opaque materials .

A non-linear least squares (NLS) fit using this form with K and κ as free parameters resulted in the values given in Table 2.1. The measurements and a fit line are shown in Fig. 2.2. Implicit in this procedure is the assumption that thermal conduction in the target dominates over other forms of heat transfer. The calculated value for K is consistent with that given in Nagai.(11). This reference also gives typical ranges for density, ρ, and specific heat, C. Diffusivities calculated from these values through $\kappa = K/\rho C$ are consistent with the value used here.

The value for α was supplied by American Standard, Inc., and is consistent with ranges of values reported in the literature (see for example Kingery (12) and Nagai (11)).

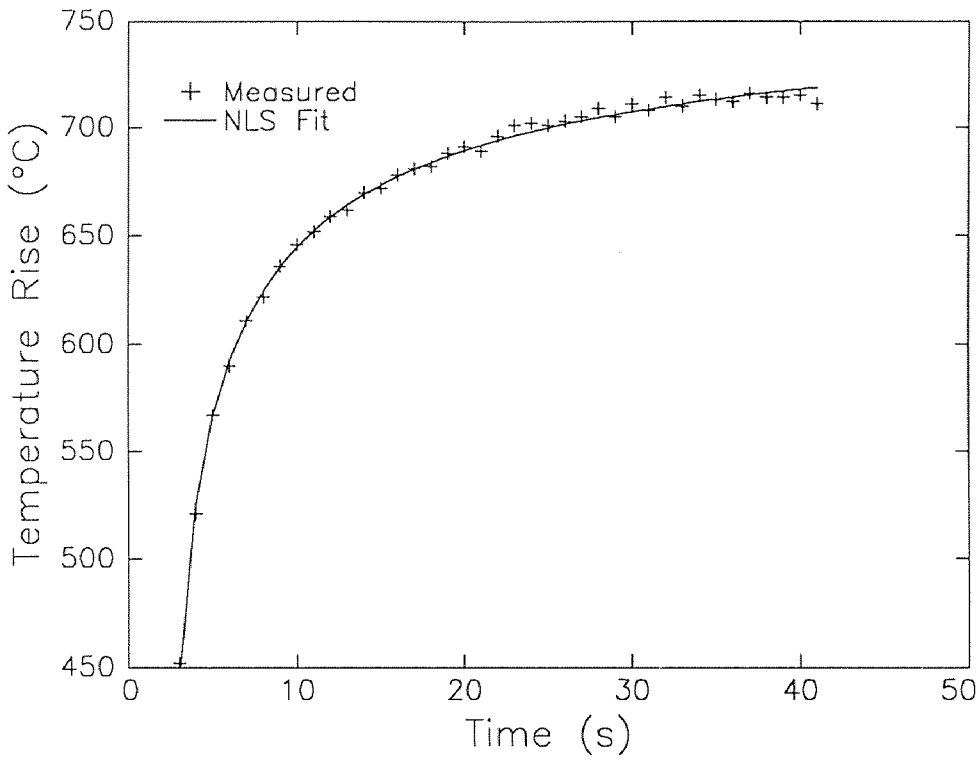


Figure 2.2 On axis surface temperature rise versus time for P = 21W, r_0= 2.2mm.

2.2.2 Elastic Properties

The values for Young's modulus and the compressive and tensile breaking stresses are taken from Chipman and Knapp (13). These are typical of values widely reported in the literature. Poisson's ratio was not known for the specific materials used, but the value of 0.25, taken from Kingery, is also typical of reported values for ceramics in general (12). In any case, the model solutions are not sensitive to small changes in ν.

CHAPTER 3

TEMPERATURE MODEL

3.1 Physical Model

3.1.1 Assumptions

The temperature rise at representative points in the target during laser heating was modeled using a finite difference approximation to the diffusion equation in three dimensions.

The target material was considered to be isotropic with respect to the thermal constants. Conductivity, K and diffusivity, κ were taken to be constant. Although in general thermal constants will have some dependence on temperature, displacement, and the state of stress, it is often possible to neglect these effects over a limited temperature range. A constant value was also assumed for the surface reflectivity, R.

The temperature profile obtained from laser heating is also dependent on the absorption depth (1/a), however for macroscopic targets where the absorption depth is much smaller than any other characteristic length in the system, this effect can be neglected. Lax (7) found this approximation to be valid when the ratio of beam radius to absorption depth is greater than approximately order 10 . Therefore, the heating action of the laser will be handled as a boundary condition and the temperature profile in depth will be controlled solely by thermal conduction.

3.1.2 Governing Equations

The diffusion equation was solved here in Cartesian coordinates. Previous studies of Gaussian heat sources have been done using cylindrical coordinates which have a natural advantage for fixed sources in that the circumferential dependencies may be eliminated by inspection. When the heat source is allowed to move across the target surface along a general path, as in direct laser writing, this advantage is lost.

The diffusion equation for an isotropic solid with no internal heat source in Cartesian coordinates is:

$$\nabla^2 T(x,y,z,t) - \frac{1}{\kappa}\frac{\delta T(x,y,z,t)}{\delta t} = 0 \tag{3.1}$$

3.1.3 Boundary Conditions

This equation is solved subject to the following boundary conditions: At the heated surface the energy flux, q, due to the laser (Eq. 3.2) is equated with the normal temperature gradient through Fourier's law for heat conduction (Eq. 3.3).

$$q = F_0 e^{-r^2/r_0^2} \tag{3.2}$$

$$q = -K\frac{\delta T(x,y,z,t)}{\delta z} \tag{3.3}$$

In Equation 3.2 r is the radial distance from the beam center on the surface. When the beam is located at (x_o, y_o), then $r^2 = (x - x_o)^2 + (y - y_o)^2$.

At the other boundaries, which are internal planes taken to be far from the heated region, the second derivative of the temperature with respect to position was set equal to zero. In other words the variation of the temperature gradients with position far from the heated region were considered negligible.

3.2 Numeric Model

3.2.1 Discretization

In Equation 3.2 the spatial derivatives were replaced with the standard centered difference approximation. Fictitious points were introduced outside the boundary planes to allow the use of centered differences on the boundary. These points were then eliminated using discretized forms of the boundary equations. For example to eliminate fictitious points above the heated boundary the discretized form of Equation 3.3 is solved for the temperature at the fictitious point, T_{i-1} :

$$T_{i-1} = T_{i+1} + \frac{2\Delta z}{K} F_0 e^{-r^2/r_0^2}$$

Now occurrences of T_{i-1} can be eliminated in favor of the associated internal point, T_{i+1}. The time derivative was replaced by a simple forward difference approximation. The resulting equation is explicit in time, and can be applied in an iterative manner to move the solution forward from some set of known initial temperatures.

3.2.2 Stability and Step Size Selection

The maximum time step is limited by the following stability criterion which results from von Neuman analysis:

$$s_x + s_y + s_z < \frac{1}{2} \tag{3.4}$$

where

$$s_x = \kappa \frac{\Delta t}{(\Delta x)^2}, \text{ etc.}$$

and Δt, Δx are the time and space steps (14).

The above restriction can be removed by the use of a fully implicit algorithm such as the Crank-Nicholson method, however these schemes are more difficult to implement on three dimensional grids, and they require that a large linear system be solved at each time step. For this simulation the maximum time step is not limited by Equation 3.4, but rather by the rapid motion of the heat source across the boundary surface. To guarantee proper resolution of the moving source the time step was selected to limit the movement of the heat source to less than one space step per time step. Given this more stringent restriction it is expected that an implicit scheme would be less efficient than that described above.

The space step must be set small enough to resolve peaks in the temperature profile. The size of these features will vary with the power, radius, and motion of the heat source. Fig. 3.1 shows a comparison of the analytic solution for a fixed source (Eq. 2.1) with values predicted by the model using various spatial step sizes characterized in terms

of the beam radius to step size ratio, ρ. For this work step sizes on the order of a few tenths of a millimeter were found to be adequate.

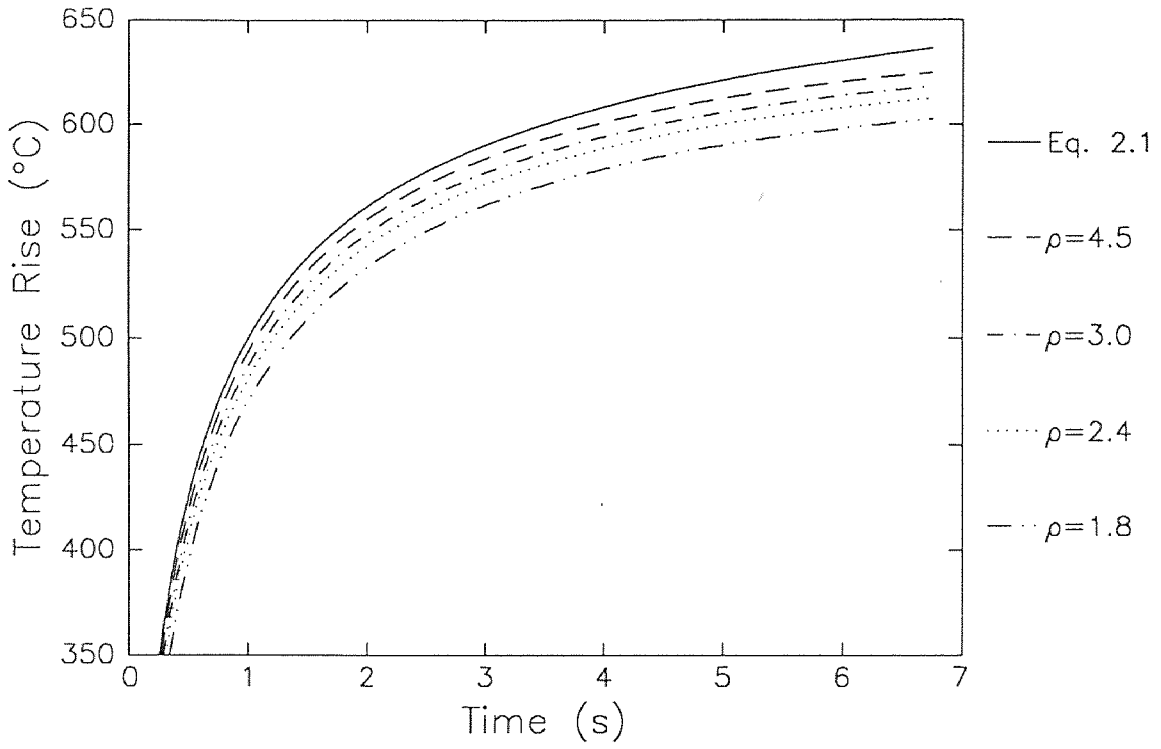


Figure 3.1 Comparison of analytic and numeric temperature rise for various space step sizes characterized in terms of the beam radius to step size ratio, ρ.

3.2.3 Results

For initial conditions all representative points were set to room temperature. This is not strictly necessary since the thermoelastic analysis depends only on temperature changes, but it facilitates comparison with data obtained from the pyrometer.

In most cases each second of simulation time required less than one minute of execution time on an engineering workstation. Upon completion the temperature and

temperature gradient for each representative point were written to a file in a form suitable for input to the thermoelastic displacement model. Typical temperature profiles at selected depths for typical heating parameters are shown in Figure 3.2. In Figure 3.3 a contour surface of temperature rise versus position on the target surface is shown. The heat source for this case was a beam of fixed power and radius moving along a circular path.

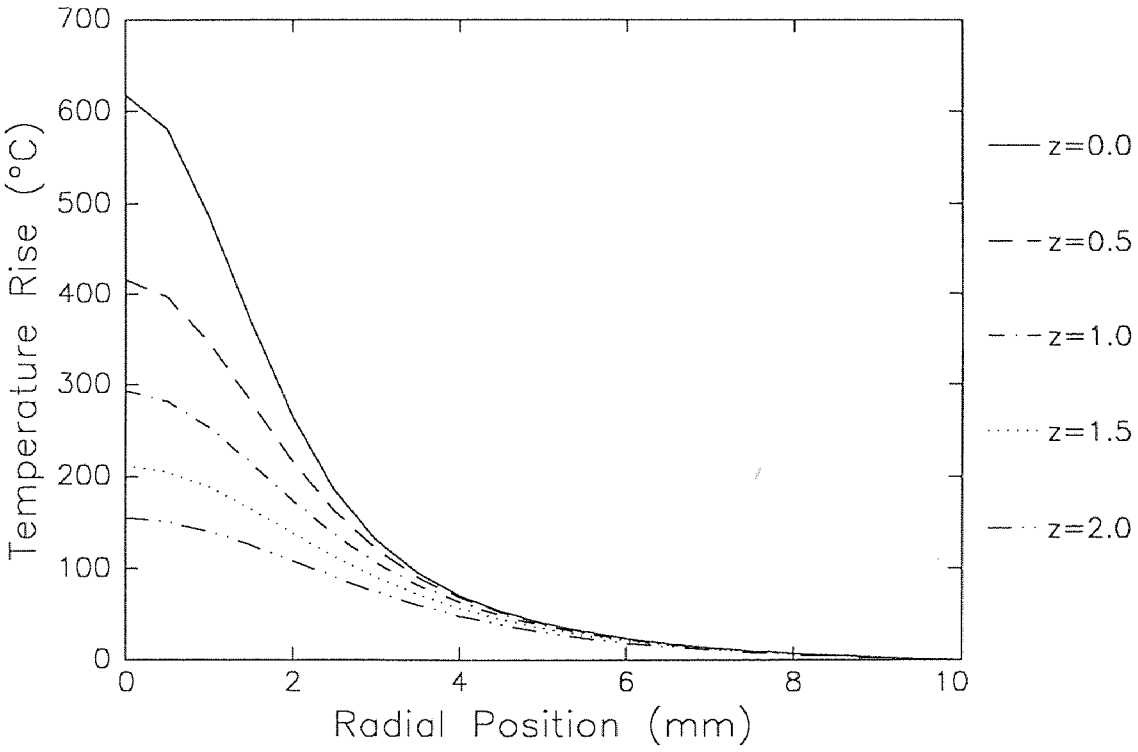


Figure 3.2 Temperature profiles at selected depths for P=11W, r_0=1.5mm at t=5.75 s, depths (z) in mm.

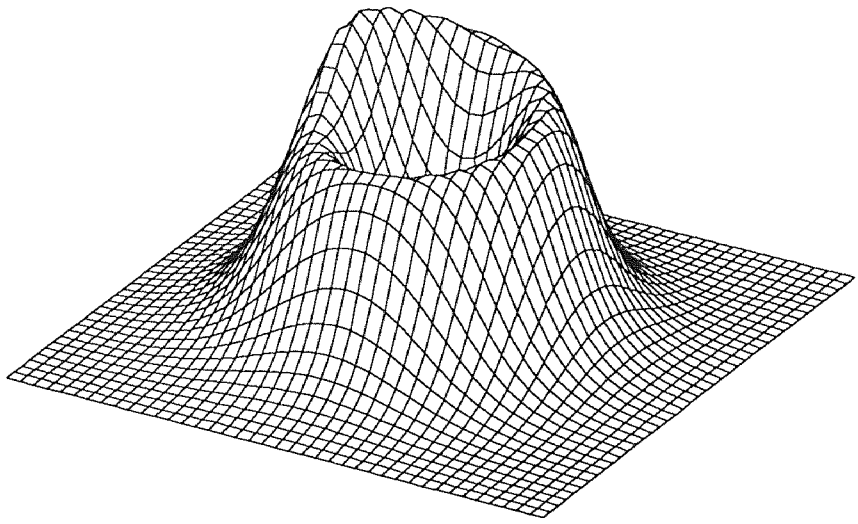


Figure 3.3 Simulated surface temperature rise versus position for a moving beam source.

CHAPTER 4

THERMOELASTIC DISPLACEMENT MODEL

4.1 Physical Model

4.1.1 Assumptions

Elastic displacements resulting from non-uniform heating are modeled in a similar manner to the temperature rise. The target is modeled as an isotropic solid deformed in the elastic range. Obviously this assumption fails with the onset of melting, but as flow serves only to reduce stresses, it continues to serve as a conservative approximation throughout. If the target can be considered annealed after resolidification the temperature changes during cooling can be used to approximate the residual stresses.

Young's modulus, E; Poisson's ratio, ν; and the compressive and tensile breaking stresses were taken to be independent of temperature and temperature history. Only static stresses were considered. The time dependence entered only through the temperature model.

4.1.2 Governing Equations

In the following sets of equations the ellipsis points indicate cyclic permutation of x, y, and z. The thermoelastic stress-displacement relations in Cartesian coordinates:

$$\sigma_x = (2G+\lambda)\frac{\delta u_x}{\delta x} + \lambda\left(\frac{\delta u_y}{\delta y} + \frac{\delta u_z}{\delta z}\right) - \frac{E}{1-2\nu}\alpha T \qquad (4.1a)$$

$$\sigma_y = \quad \ldots \qquad (4.1b)$$

$$\sigma_z = \quad \ldots \qquad (4.1c)$$

$$\tau_{xy} = G\left(\frac{\delta u_x}{\delta y} + \frac{\delta u_y}{\delta x}\right) \tag{4.1d}$$

$$\tau_{yz} = \quad \ldots \tag{4.1e}$$

$$\tau_{zx} = \quad \ldots \tag{4.1f}$$

are substituted into the equilibrium equations:

$$\frac{\delta \sigma_x}{\delta x} + \frac{\delta \tau_{xy}}{\delta y} + \frac{\delta \tau_{zx}}{\delta z} + F_x = 0 \tag{4.2a}$$

$$\ldots \tag{4.2b}$$

$$\ldots \tag{4.2c}$$

to get three equations in terms of the three unknown displacements, u_i (15).

$$(2G + \lambda)\frac{\delta^2 u_x}{\delta x^2} + G\left(\frac{\delta^2 u_x}{\delta y^2} + \frac{\delta^2 u_x}{\delta z^2}\right) + (G + \lambda)\left(\frac{\delta^2 u_y}{\delta x \delta y} + \frac{\delta^2 u_z}{\delta z \delta x}\right) = \frac{\alpha E}{1 - 2\nu}\frac{\delta T}{\delta x} \tag{4.3a}$$

$$\ldots \tag{4.3b}$$

$$\ldots \tag{4.3c}$$

Here the body forces, F_i in Equations 4.2 are neglected, and the shear modulus, G and Lamé's constant, λ are related to E and ν by:

$$G = \frac{E}{2(1 + \nu)} \tag{4.4}$$

$$\lambda = \frac{\nu E}{(1 + \nu)(1 - 2\nu)} \tag{4.5}$$

and α is the coefficient of linear expansion.

4.1.3 Boundary Conditions

Here again a mixed boundary condition is required. At the target surface (z = 0) stresses normal to the surface are set equal to zero:

$$\sigma_z = \tau_{yx} = \tau_{zx} = 0 \tag{4.6}$$

At the other boundary planes, which again are internal planes considered to be removed from the heated region, the displacements are set equal to zero. The validity of this boundary condition will be re-examined later.

4.2 Numeric Model

4.2.1 Discretization

As in the temperature model, the spatial derivatives in Equations 4.3 are replaced with their finite difference equivalents. After discretization, the RHS of Equations 4.3 have the form:

$$\left(\frac{-2(2G+\lambda)}{(\Delta x)^2}+\frac{-2G}{(\Delta y)^2}+\frac{-2G}{(\Delta z)^2}\right)u_{x_{i,j,k}}+$$

$$\frac{(2G+\lambda)}{(\Delta x)^2}\left(u_{x_{i+1,,j,k}}+u_{x_{i-1,,j,k}}\right)+$$

$$\frac{G}{(\Delta y)^2}\left(u_{x_{i,,j+1,k}}+u_{x_{i,,j-1,k}}\right)+\frac{G}{(\Delta z)^2}\left(u_{x_{i,,j,k+1}}+u_{x_{i-1,,j,k-1}}\right)+$$

$$\frac{(G+\lambda)}{4\Delta x\Delta y}\left(u_{y_{i+1,,j+1,k}}-u_{y_{i+1,,j-1,k}}-u_{y_{i-1,,j+1,k}}+u_{y_{i-1,,j-1,k}}\right)+$$

$$\frac{(G+\lambda)}{4\Delta z\Delta x}\left(u_{z_{i+1,,j,k+1}}-u_{z_{i+1,,j,k-1}}-u_{z_{i+1,,j,k+1}}+u_{z_{i-1,,j,k-1}}\right)$$

and the LHS, which becomes the source vector is:

$$\frac{\alpha E}{1-2\nu}\frac{\delta T}{\delta x} \tag{4.7a}$$

... (4.7b)

... (4.7c)

Here the derivative of temperature with respect to position is not cast in finite difference form since the temperature model output includes the evaluated gradients.

Fictitious points above the z plane are eliminated through Equations 4.1 under the surface condition (Eq. 4.6). At the other boundaries, fictitious points are not required. Since the displacements are set equal to zero, references to points outside the grid are ignored. Spatial step sizes were chosen to be compatible with the temperature model. It should be noted here that grids which are fine enough to resolve the details of the temperature distribution may not suffice for the displacement field where the features are determined by the temperature gradients. A single axisymmetric temperature peak will produce a displacement field with a zero at the center due to symmetry and absolute maximum displacements near the radius where the maximum temperature gradients occur, as shown in Fig. 4.1. This roughly halves the minimum feature size which the discretizing grid must support.

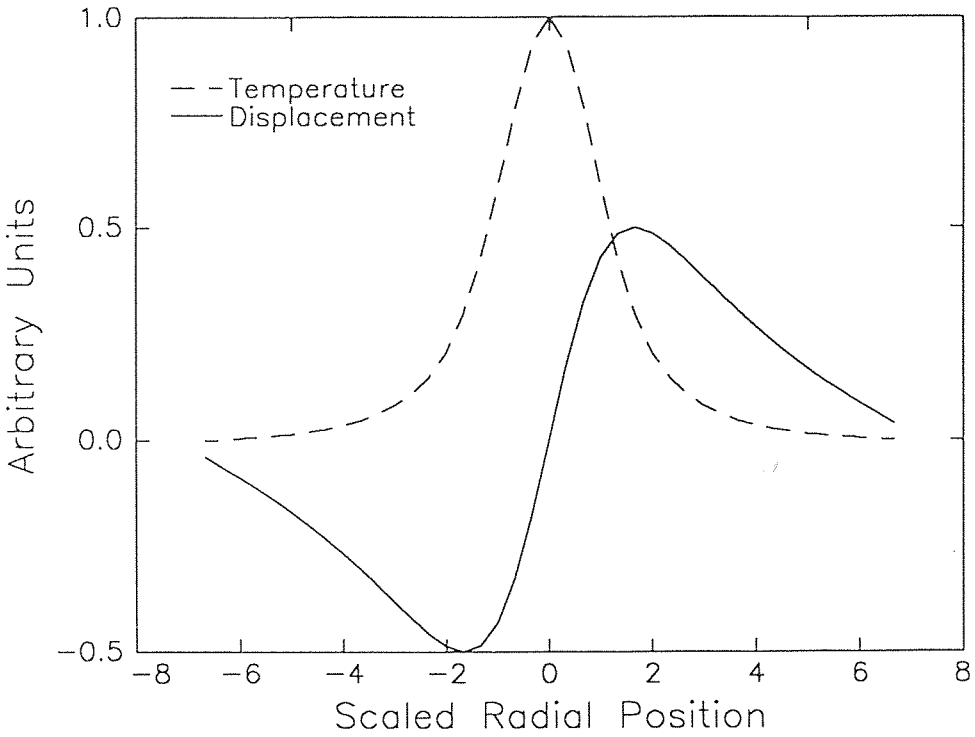


Figure 4.1 Typical radial displacement as a function of scaled radial position (r/r_0) for a fixed Gaussian heat source.

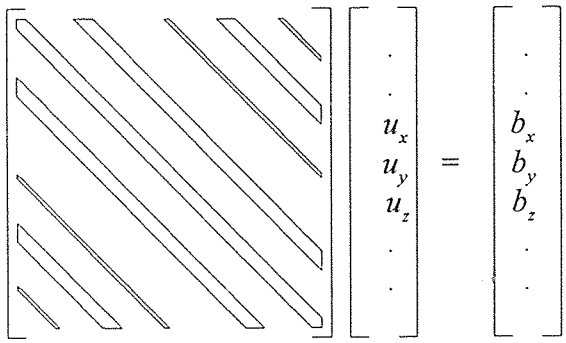
4.2.2 Method of Solution

If the solution vector of Equations 4.7 is ordered as follows:

$$\left(\ldots, u_{x_{n-1}}, u_{y_{n-1}}, u_{z_{n-1}}, u_{x_n}, u_{y_n}, u_{z_n}, u_{x_{n+1}}, u_{y_{n+1}}, u_{z_{n+1}}, \ldots\right)$$

where n collectively represents the indices required to specify the nth spatial point in the target, then the coefficient matrix will have the symmetric block banded structure shown schematically in Equation 4.8. For a volume of N_x by N_y by N_z spatial points with three equations per point the bandwidth is $3N_xN_y$, suggesting that a simple direct elimination would require on the order of $(3N_xN_y)^3N_z^2$ operations. For the target size and spatial

resolution required here, the number of equations ($3N_xN_yN_z$) is approximately 50,000 making direct elimination impractical.



(4.8)

Fortunately, despite the large bandwidth, the matrix is quite sparse. For systems such as this where direct methods destroy the sparsity of the matrix by filling in coefficients between the upper bands, iterative techniques such as successive over relaxation (SOR) are recommended (16).

To guarantee the convergence of an iterative method it is generally necessary to show that the iteration matrix associated with that method has a spectral radius less than one. For a system such as this, the determination of the spectral radius is a problem of the same order as the solution of the system itself. A simpler test for convergence is to check that the matrix of coefficients is strictly diagonally dominant. This a sufficient, but not a necessary condition (17). The coefficient matrix for this model does not satisfy this weaker test, however it was found in practice that a unique solution was obtained for a variety of initial solution vectors.

For this system the SOR method was used with an experimentally determined acceleration parameter of 1.5. Displacements were initially set equal to zero. The

solution was considered converged when the 1-norm of a vector of sample points was found to change by less than a small arbitrarily chosen percentage.

4.2.3 Results

Once the displacements are calculated, the stresses are determined by Eq. 4.1 The total strains are defined as:

$$\varepsilon_x = \frac{\delta u_x}{\delta x} \tag{4.9a}$$

... (4.9b)

... (4.9c)

$$\gamma_{xy} = \frac{\delta u_x}{\delta y} + \frac{\delta u_y}{\delta x} \tag{4.9d}$$

... (4.9e)

... (4.9f)

and the elastic normal strains, ε' are related to the total normal strains by:

$$\varepsilon_x' = \varepsilon_x - \alpha T \tag{4.10a}$$

... (4.10b)

... (4.10c)

Typical solution times for the results presented here were on the order of ten minutes using an engineering workstation.

For model validation a simple case was examined first. A fixed Gaussian beam with radius r_0 = 1.5mm and total absorbed power P_0 = 11W irradiates the target for 5.75 seconds. The temperature profile along radials at various depths is again that shown previously in Fig. 3.2. These profiles are consistent with those resulting from analytical treatments in reference 7.

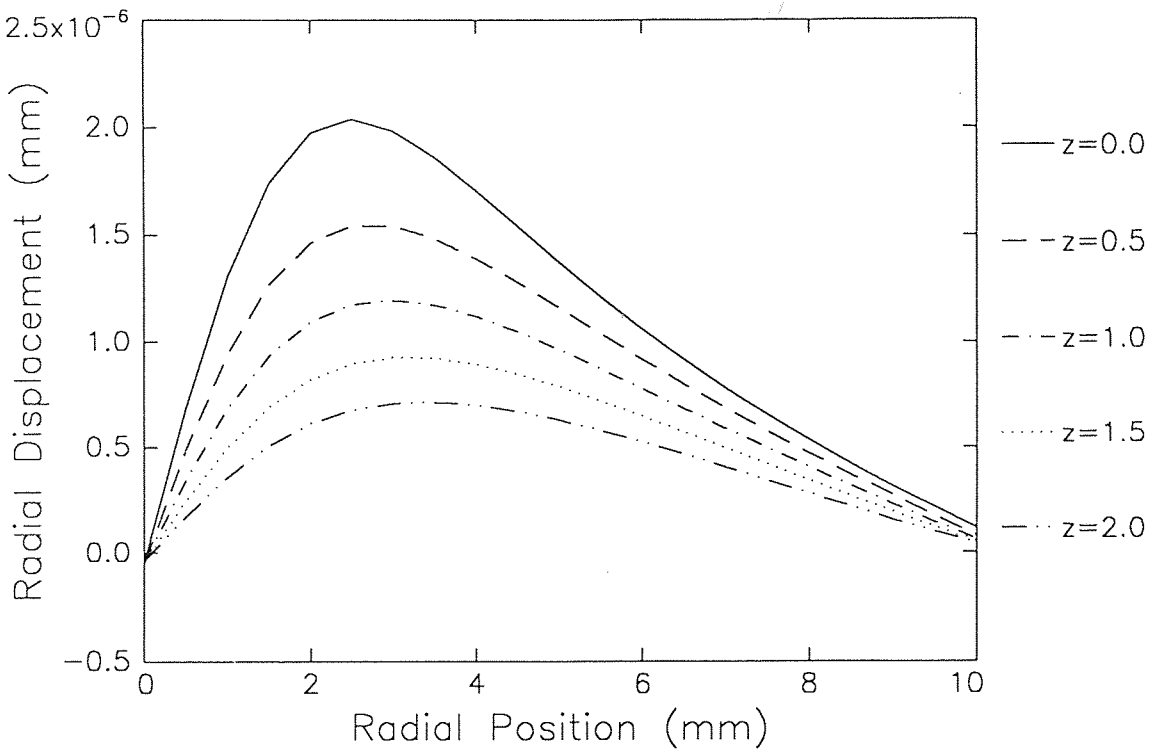


Figure 4.2 Radial displacement versus position for a fixed Gaussian source, r_0=1.5mm, P=11W, t=5.75s, depths (z) in mm.

Fig. 4.2 shows radial displacement vs. radial position. Note that near the right-hand boundary (r=10mm) the displacement is not approaching the axis in a fully asymptotic manner. This shows that even when the boundary plane is removed from the heated region (cf. Fig 3.2) there may still be significant displacement due to the elastic

response of the material. This is seen more clearly in Fig. 4.3 where the total strain and elastic strain are seen to converge at a finite value. This effect diminishes as successively larger regions are modeled without any significant effect on the values or locations of the maximal stresses and strains which are of interest here. Because of this artifact the tail region of the displacement cannot be compared directly with analytic solutions where the radial boundary is at infinity.

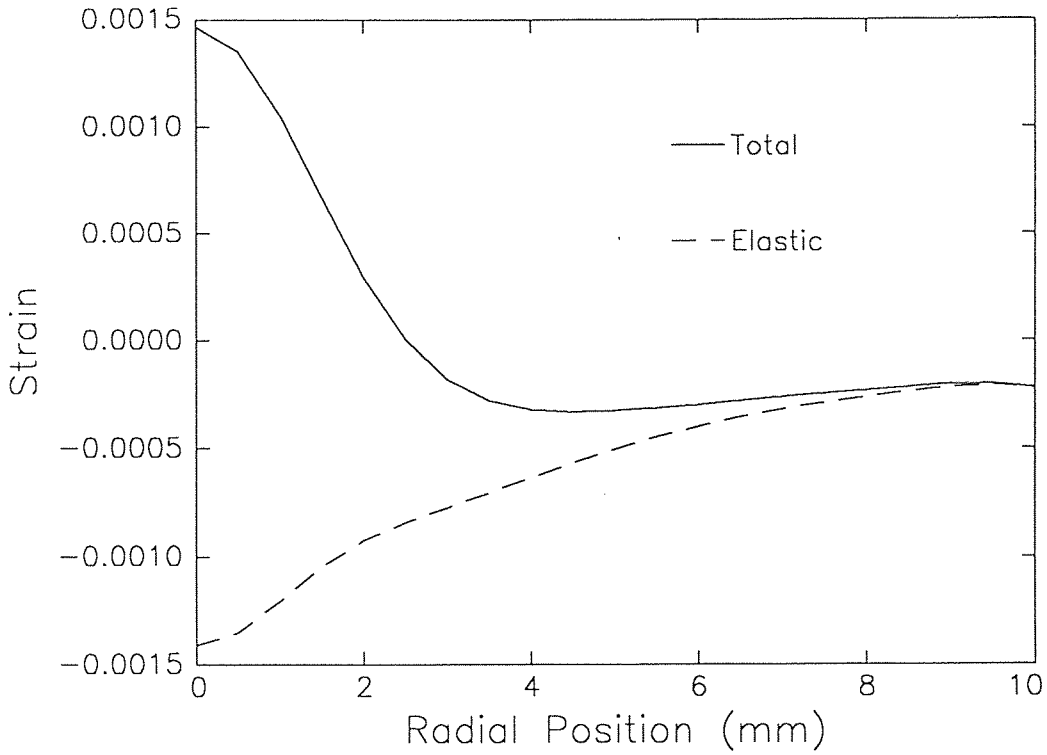


Figure 4.3 Total and elastic normal radial strains versus position for a fixed Gaussian source.

Fig. 4.4 shows the three normal stresses along the x axis and at various depths. Here the x axis is labeled as radial position because the solutions are symmetric with respect to rotation about the z axis. σ_y and ε_y at points on the x axis are the tangentially

directed normal stress and strain respectively. σ_x and σ_y are the largest normal stresses. Both decrease monotonically, with σ_y falling off more rapidly with increasing position than σ_x. σ_z and τ_{zx} (not shown) are both zero on the surface as required by the boundary condition. σ_z changes from compressive to tensile stress with increasing distance from the heat source. The total normal strains, ε_x, ε_y, and ε_z, at various depths are given in Fig. 4.5. Again the x and y components are of the same order of magnitude, but here ε_z is the largest component. These results are qualitatively consistent with the trends noted in the semi-analytic solutions presented by Welsh, et al. (2) with the exception that the depth directed components are somewhat larger relative to the x and y components for both stresses and strains . This is to be expected, since the temperature distributions used in Welsh, et al. are at the equilibrium state, i.e. infinitely long heating, whereas the temperature profiles used here are the result of short intense heating as required by the spot glazing application.

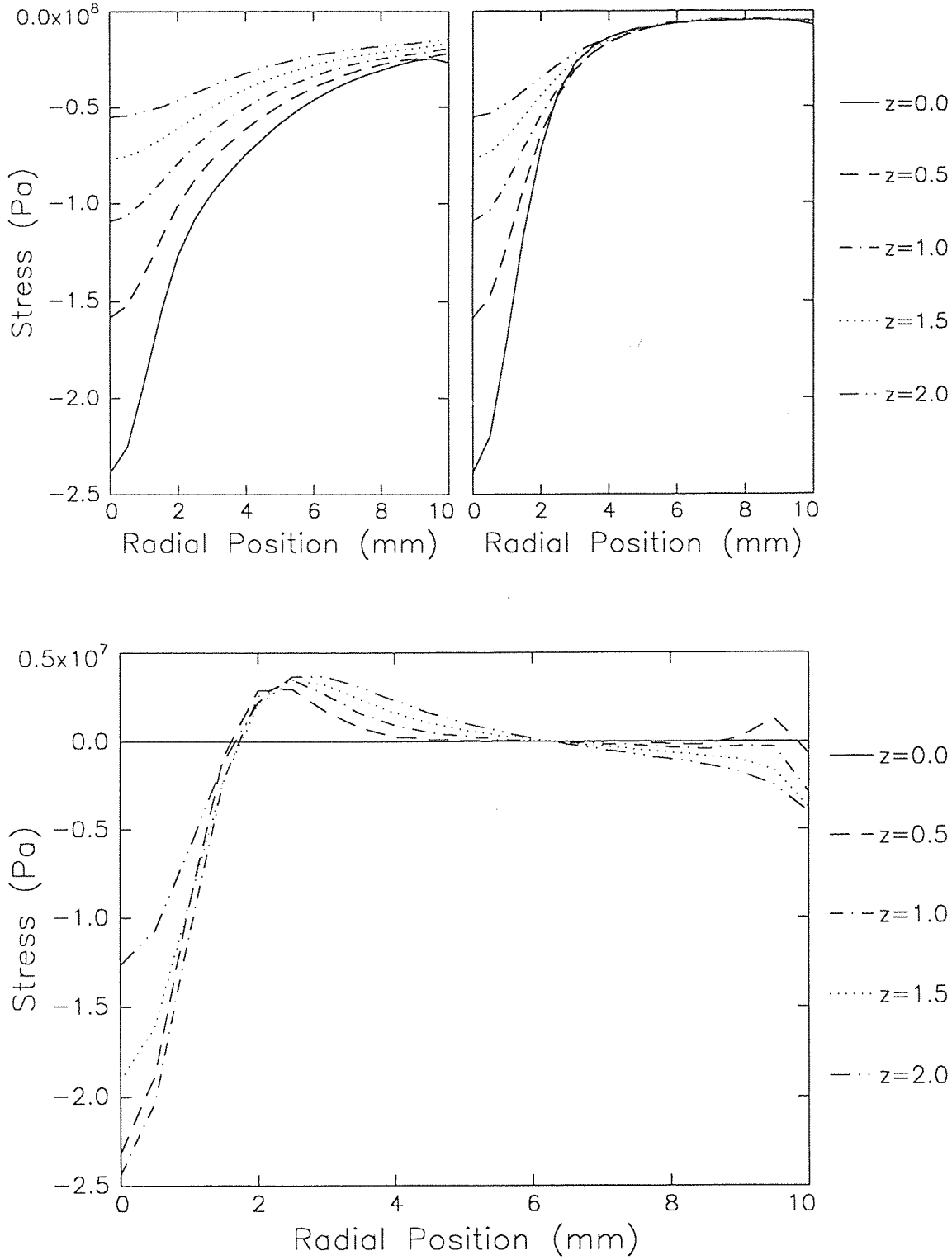


Figure 4.4 Normal stresses versus radial position for a fixed Gaussian source. Top left, σ_x; top right, σ_y; bottom , σ_z.

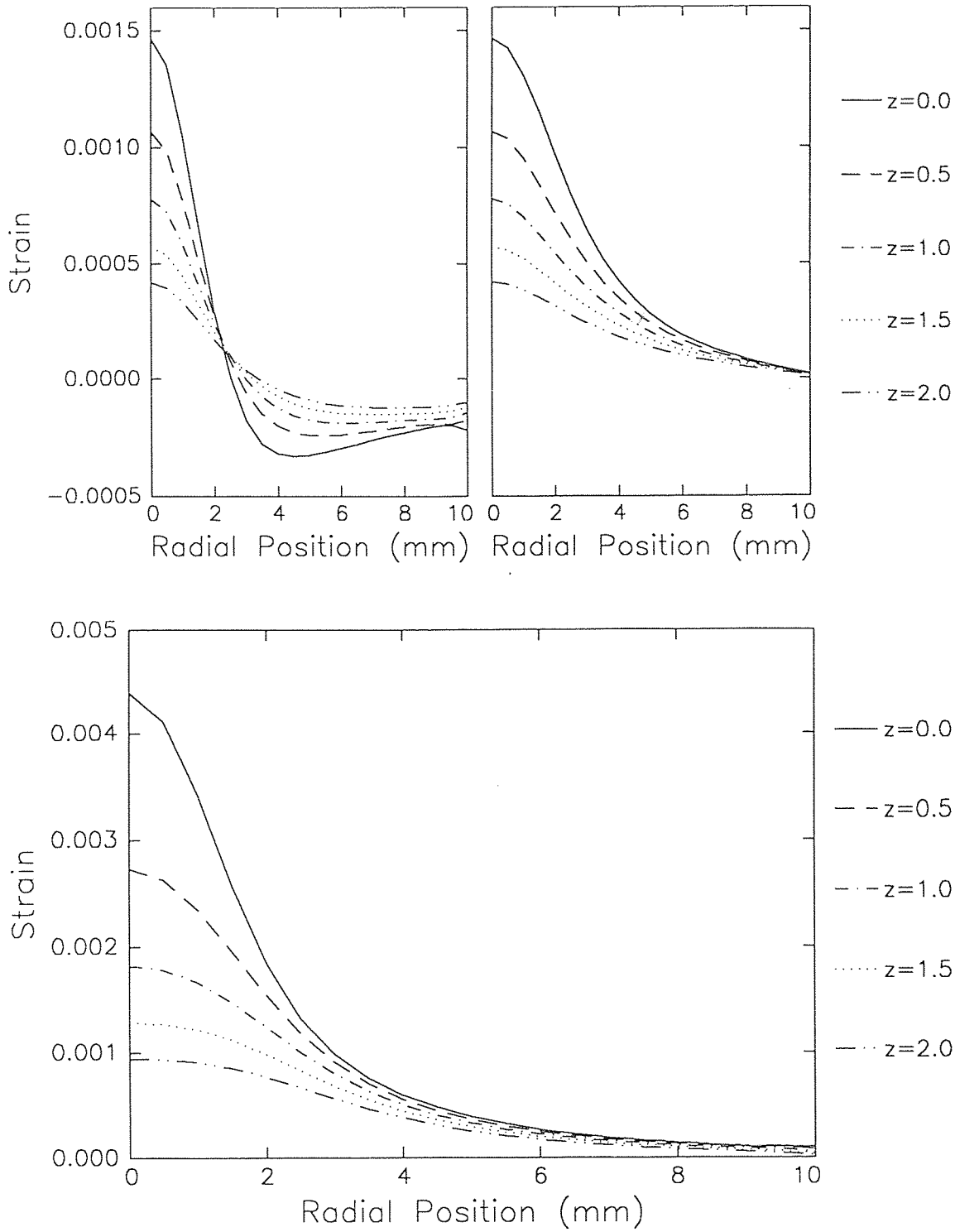


Figure 4.5 Normal strains versus radial position for a fixed Gaussian source. Top left, ε_x; top right, ε_y; bottom, ε_z.

4.2.4 Application to Time Varying Sources

In this section the thermoelastic displacement model is used to simulate the stresses which result when the heat source is allowed to change in time. First the effect of varying the radius of a fixed position beam is considered. For the test case the total absorbed power is fixed at 11W, and the beam radius is allowed to increase linearly with time from 1 to 2 mm over 6 seconds. Figure 4.6 shows the resulting surface temperature profiles for this case and for the fixed radius case where the maximum temperatures are scaled to 1 for comparison.

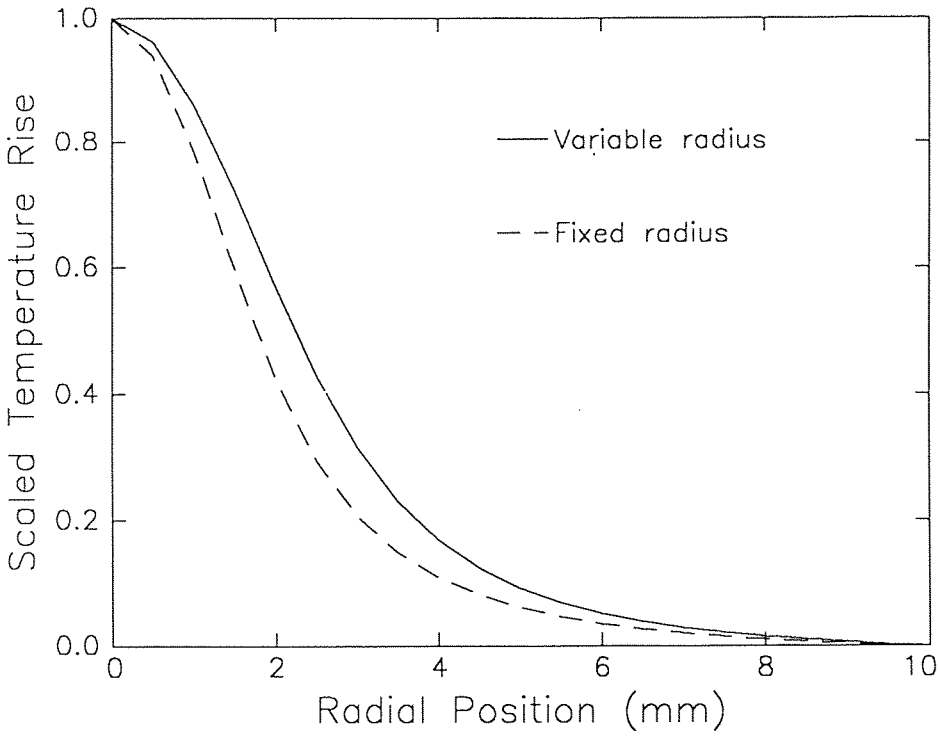


Figure 4.6 Surface temperature profiles for fixed and variable radius heating.

The temperature profile for the variable radius case is broader than that of the fixed radius case. The displacement, stress, and strain profiles all show a similar broadening, but are otherwise identical to those shown in Figures 4.2, 4.4, and 4.5.

While varying the radius of a fixed beam does provide additional heating to areas adjacent to the target region, it does not heat these areas preferentially. For fixed Gaussian sources the maximum power density is always on the beam axis. This can lead to significant overheating of the target center while peripheral regions are maintained at the desired process temperature. In the spot glaze repair application this effect was observed for fixed beam heating. When peripheral areas were sufficiently heated to achieve fusion the central region was often over-fired, typically resulting in a transparent glassy region where the opacifiers failed.

Within the restriction of single beam heating, there are two possible techniques which can be used to preferentially heat adjacent areas. One method is to tune the heating laser to operate in a doughnut mode, where the power density on axis is at a local minimum. Spann, et al. (5) have successfully applied this technique to the end melting of ceramic rods. The second method involves rapidly moving the heating beam across the target surface in a programmed pattern to achieve the desired temperature profile. For spot defects this generally means moving the beam along circular or spiral paths.

In this section the models are used to simulate the latter technique, although a simple change in the boundary conditions of the temperature model would allow the modeling of other beam profiles. For this test case a 2 mm radius beam circles the origin at a radius of 3 mm at 16 revolutions per second. The total absorbed power is 20W. The temperature profiles for this type of heating are not axisymmetric. There is localized temperature peak near the instantaneous position of the beam (cf. Figure 3.3 where the beam was moving counter-clockwise on the far side of the origin). This effect is always

present, although it diminishes at high rates of revolution. Figure 4.7 shows typical radial temperature profiles at various depths. Note that as opposed to the fixed position profiles, a much broader area can be maintained at a relatively constant elevated temperature. In the spot glazing application the use of this heating technique eliminated the glaze decomposition problems associated with the fixed beam heating.

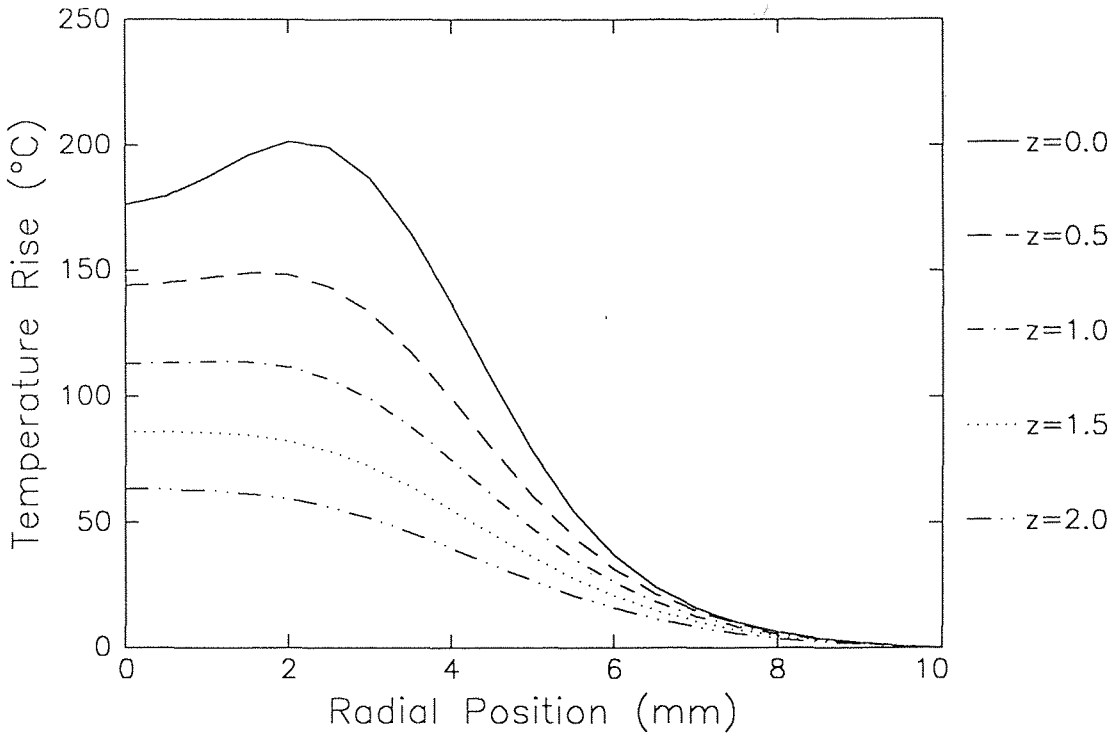


Figure 4.7 Temperature rise versus position along a typical radial for heating with a Gaussian beam moving on a circular path.

The simulated normal stresses for this temperature distribution are given in Figure 4.8. The maximum stresses are shifted out from the origin to a position roughly equal to the beam path radius. Samples heated beyond their elastic limits using this profile generally showed an annulus shaped region of crazing at roughly this same radius.

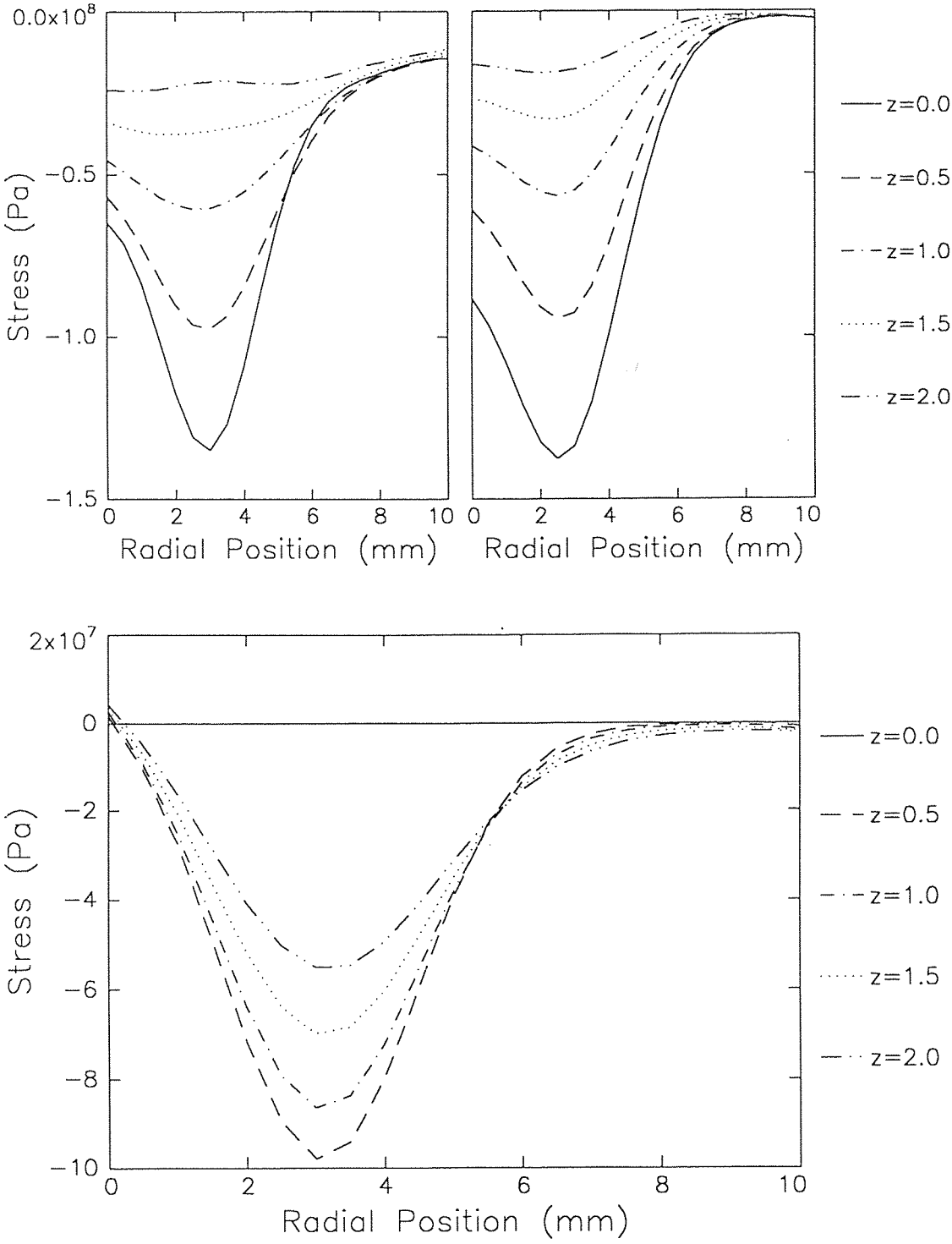


Figure 4.8 Normal stresses versus radial position for a fixed Gaussian source. Top left, σ_x; top right , σ_y; bottom, σ_z.

CHAPTER 5

EXPERIMENT

5.1 Physical Experiment

Up to this point the model results have been examined in a qualitative manner. In this chapter a simple experiment is described which allows a comparison of the simulated stresses with published values for the breaking stresses.

The experimental apparatus is shown schematically in Figure 5.1. A 50 Watt $C0_2$ surgical laser operating in CW mode was used. This laser was modified to allow the total beam power to be controlled in real time using a personal computer with an analog output board. The beam was focused through a servo motor driven telescope with a dedicated programmable controller. To aim the beam the final mirror was rotated and tilted using stepping motors, also under computer control. Time and space resolved temperature data were collected with a Pyrometer Dual Pyrofiber optical pyrometer. In the current work these data are used only as a diagnostic tool in designing the heating profiles, but the system could easily be modified to use the real-time temperature data to adjust the heating programs in response to the target conditions. The system is capable of executing complex heating programs simultaneously varying beam power, radius and position in a repeatable manner.

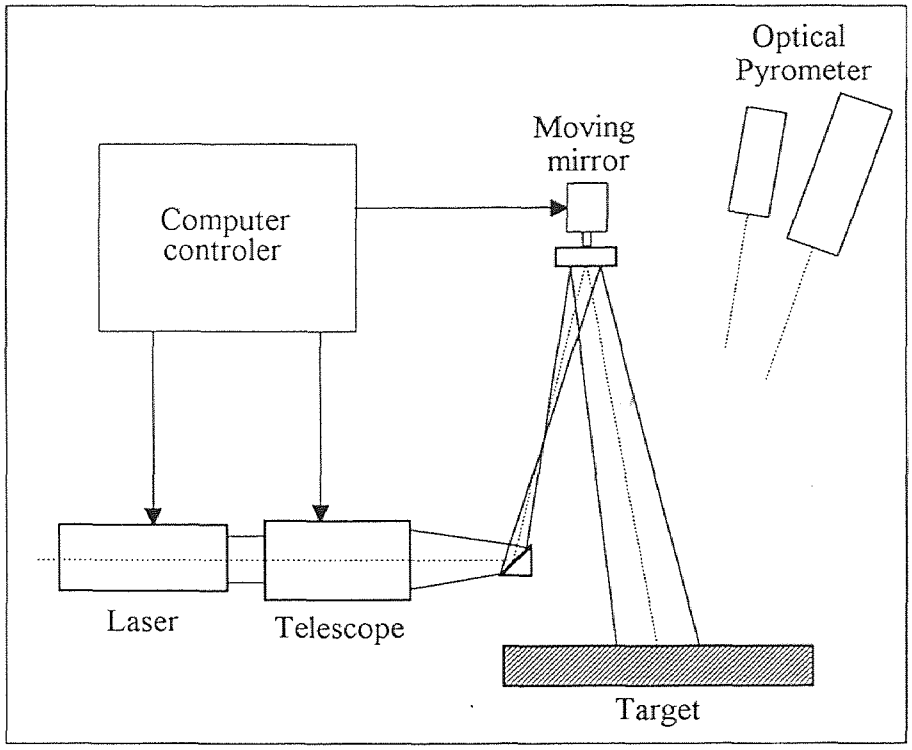


Figure 5.1 Schematic of the experimental apparatus.

5.2 Analysis with Simulation

The targets used in these experiments were glazed porcelain tiles. If a localized region of the target surface is heated above the strain point of the glaze, then elastic strains will be converted to plastic strains. For a fixed position beam the resulting stress field will be symmetric with respect to the beam. In the central region where the temperature is above the strain point the compressive stresses induced by thermal expansion will be reduced as the strains are converted. This area is surrounded by a region where the temperature is below the strain point and the material is in a state of elastic compression. For areas far from the heating beam there are no thermally induced stresses, although there typically will be a significant compression in the glaze (roughly 10,000 psi) which is built in during the

manufacture of the porcelain to prevent cracks from propagating from surface defects into the body (12).

As the target is allowed to cool the outer regions will return to the original stress state, but the central region will either be in a state of reduced compression or in a state of tension, depending on the length of time allowed for plastic effects. If a state of tension exists, and the tensile limit is exceeded, cracks will develop during or after cooling. Due to symmetry these cracks typically are ring-shaped, centered on the heated area.

A quantitative knowledge of the residual stress field is critical to the design of successful localized heating algorithms for brittle materials. It is towards this end that the models presented here were developed. To apply these models, which do not explicitly account for plastic effects, a simplifying approximation must be made. If the target is considered to be fully annealed at maximum temperature (all elastic stains relieved through plastic effects) then the temperature changes during cooling can be used to calculate the final stress state, or alternatively the compression state at maximum temperature can be used (with the appropriate change in sign) as the residual tensile stress state. Any plastic effects that occur during cooling would reduce the residual stresses below the calculated values.

Using this approximation the model was used to calculate the residual stress fields for a number of samples heated using the apparatus described above. These samples were heated with fixed position beams with various radii and total powers. Each sample was sufficiently over heated so that a ring crack occurred upon cooling. These cracks were observed to start at a point removed from the center of heating and then to propagate

along a roughly circular path forming the ring. The ring crack radii were then measured. The temperature and normal radial stress at the surface corresponding to each ring crack radius was then extracted from the simulation output for each sample. Ideally each ring crack would occur at a position in the stress field corresponding to the tensile limit of the material, however a number of factors such as local inhomogeneities in the material, variations in material parameters with temperature, variations in the laser power, etc. all contribute to the spread of the data. The breaking stress as a function of temperature is shown in Figure 5.2, along with two representative published values for room temperature.

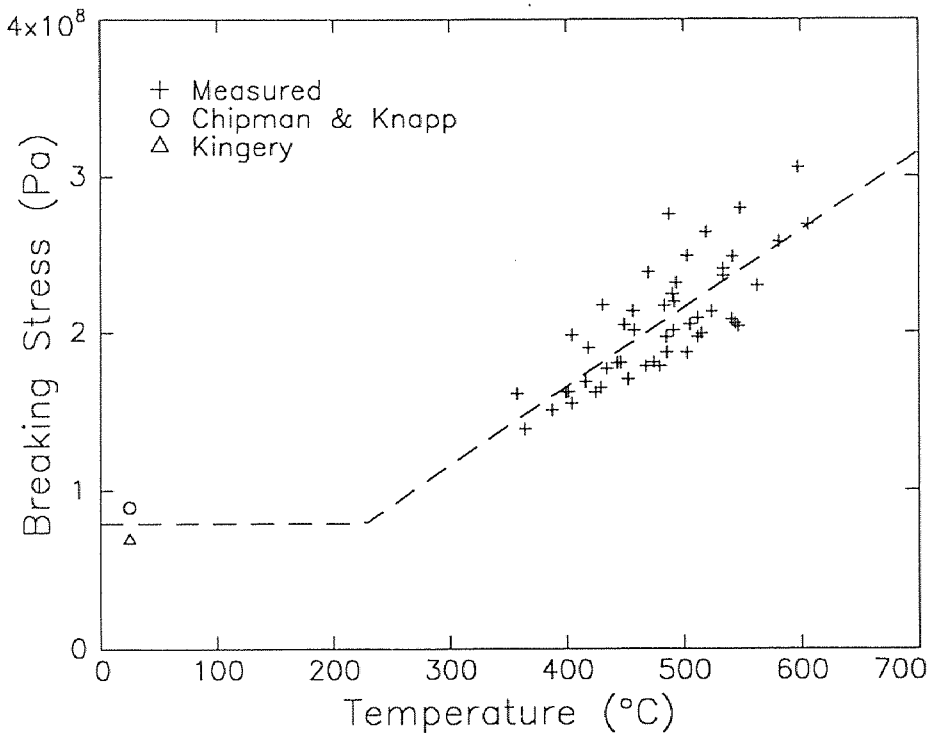


Figure 5.2 Simulated breaking stress versus temperature for thermally induced ring cracks.

Data could not be collected below approximately 350°C because the thermally induced stresses were not high enough to cause cracking. At high temperatures the data are unreliable as unmodeled effects such as vaporization become important. The dashed line in the figure is an estimate of the behavior in the untested region based on observations by Weyl quoted in Chu (18). Weyl found that the breaking strength of porcelain maintained a constant value with increasing temperature, then increased linearly as the annealing range was approached. This effect is the result of the healing of internal flaws at higher temperatures.

CHAPTER 6

CONCLUSION

Numeric models for thermoelastic effects resulting from localized heating with Gaussian profile laser beams have been presented in the context of ceramic materials processing. The solutions were compared qualitatively with an existing semi-analytic solution for the case of a source with fixed position and radius. The models then were applied to time varying heating programs used to generate more useful thermal profiles. Finally the models are used to predict the breaking stresses associated with thermally induced cracks in porcelain. These results generally agreed with published values.

The predictive power of these models could be enhanced by relaxing some of the simplifications made during the initial analysis. The variation of the material properties with temperature, once determined, could be included in a relatively straightforward fashion. For composite targets it may be necessary in some cases to account for the material properties of the individual components separately.

A further extension of the models would be to explicitly account for the effects of melt and flow. In the porcelain defect work this might be quite difficult as the filler materials often consist of components with widely varied melting temperatures.

These models are useful in designing and testing heating programs used in laser surface modification. For brittle materials, where there may be only a narrow window in the parameter space for successful processing, such modeling is virtually essential.

APPENDIX

PROGRAM CODE

The models presented in this paper were written in C and compiled and executed on a Sun Microsystems workstation under the UNIX operating environment. Source code for the two models is presented below, along with commentary on program organization. Executable statements are indented to distinguish them from the commentary.

A.1 Temperature Model

The temperature model takes as input an ASCII file consisting of eight columns of decimal numbers. Each line of this input file represents a single stage of the heating program during which the beam radius, beam position radius, and beam position angle either remain fixed or change in a linear manner. The first number gives the total beam power in Watts, the second number is the duration of the stage in seconds. The third and forth numbers are the starting and ending beam radii in millimeters. The fifth and sixth numbers are the starting and ending beam position radii in millimeters. The last two numbers are the starting and ending beam position angles expressed in units of revolution, i.e. 1.0 represents a single orbit of the beam around the center point. This file is redirected into the model at execution.

The values of constants used in the simulations and a summary of the heating program are output to file called 'legend.txt'. Final surface temperatures are output to file called 'lasttemp.dat'. Maximum surface temperature gradients, and the temperatures at

which they occur are output to 'max_grad.dat' and 'maxgtemp.dat', respectively. Information required by the thermoelastic displacement model is output to a file called 'tempinfo.dat'.

```
/* This is an explicit finite difference model (FDM) for laser surface
   heating of a solid. */

/* compiled under UNIX using command: cc lg.c -fsingle -lm -O
   example of execution: a.out <in.dat >t.dat */

#include <stdio.h>
#include <math.h>

#define PI 3.141593

/* N_ are numbers of grid points used in the discretization.
   2 through N_-1 are points internal to the volume, 1 & N_ are
   boundary points, 0 & N_+1 are "fictitious" points.  Keeping
   these values odd allows the (N_+1)/2 point to lie at the
   center of each axis. */
#define NX 41
#define NY 41
#define NZ 11

/* _MAX, _MIN are the positions of the boundary planes in meters. _MIN
are
   associated with subscript 1. _MAX are associated with subscript N_. */
#define XMIN -7.50e-3
#define XMAX 7.50e-3
#define YMIN -7.50e-3
#define YMAX 7.50e-3
#define ZMIN 0.0e-3
#define ZMAX 5.0e-3

float  T_amb;          /* ambient temperature (deg C) */

void main()
{
 /* function & variable declarations */

 /* q_source() contains the normalized spatial power distribution of
    the laser which is incident on the xy plane at z = ZMIN.  The local
```

```
   power density in W/m^2 is returned as a function of position, time,
   beam radius, & total power. */
 float q_source();

 /* T0[i][j][k] is the volume of grid points which have known temperatures
   at the begining of each time step.  T1[i][j][k] is the volume of grid
   points at the advanced time, which are calculated during each time step
*/
 float T0[NX + 2][NY + 2][NZ + 2],
       T1[NX + 2][NY + 2][NZ + 2],

 /* TM[i][j] is maximum temperature reached for each surface grid point
*/
       TM[NX + 2][NY + 2],

 /* GM[i][j] is maximum surface temperature gradient reached for each
   surface grid point */
       GM[NX + 2][NY + 2],

 /* GT[i][j] is the temperature at which GM[i][j] was recorded */
       GT[NX + 2][NY + 2];

 float t,              /* time (seconds) */
       dt,             /* delta t (seconds) */
       dx, dy, dz,     /* delta position (meters) */
       Kappa,          /* thermal diffusivity (meters^2 / second) */
       sx, sy, sz,     /* coefficients to simplify expressions */
       x, y,           /* position (meters) */
       kc,             /* thermal conductivity (Watts / (meter * deg C)) */
       r0,             /* initial beam radius (meters) */
       r1,             /* final beam radius (meters) */
       dr,             /* beam radius increment (meters) */
       w,              /* total beam power (Watts) */
       duration,       /* time a given set of parameters is in use (s) */
       tchange,        /* time to get next set of laser parameters (s) */
       rho0,           /* initial beam position radius (meters) */
       rho1,           /* final beam position radius (meters) */
       drho,           /* beam position increment (meters) */
       phi0,           /* initial beam position angle (radians) */
       phi1,           /* final beam position angle (radians) */
       dphi,           /* beam position angle  increments (radians) */
       gradx,          /* dT/dx */
       grady,          /* dT/dy */
       gradz,          /* dT/dz */
       surfgrad,       /* (gradx^2 + grady^2)^0.5 */
```

```
        cx, cy;         /* beam position in cartesian coordinates */

int     i, j, k,        /* subscripts for gridpoints */
        pstep;          /* program step number */

char    fname[14];      /* output file name used after each program step */

long    tcount = 0;     /* count of total number of time steps taken */

FILE    *legend,        /* pointer for output to legend text file */
        *dataout,       /* pointer for output of grid data */
        *dataout2;              /* pointer for output of grid data */

/* "executable" statements begin here */

legend = fopen("legend.txt", "w");

/* calculate delta positions */
dx = (XMAX - XMIN) / (float)(NX - 1);       /* meters */
dy = (YMAX - YMIN) / (float)(NY - 1);
dz = (ZMAX - ZMIN) / (float)(NZ - 1);

/* set material constants */
/* thermal diffusivity */
Kappa = 1.0e-6;                     /* m^2 / s */

/* thermal conductivity */
kc = 1.881;                         /* W / (m * deg C) */

/* room temp */
T_amb = 20.0;                       /* deg C */

/* the time step is fixed by a stability restriction for explicit
   finite difference methods.  The restiction is usually expressed
   as: sx+sy+sz < 0.5.  Strictly speaking, this restiction holds
   only for linear systems, so a more conservative value is used
   below to account for the nonlinear source in this case. Ref.
   C. A. J. Fletcher, Computational Techniques for Fluid Dynamics,
   Vol. 1, Springer-Verlag, Berlin, 1988. p250. */
dt = 0.4 * (dx*dx * dy*dy * dz*dz) /
    (Kappa * (dy*dy*dz*dz + dx*dx*dz*dz + dy*dy*dz*dz));

/* start legend text */
fprintf(legend, "Parameters:\n");
fprintf(legend, "thermal diffusivity:      %e (m^2 / s)\n", Kappa);
```

```
fprintf(legend, "thermal conductivity:      %e (W / (m * deg C))\n", kc);
fprintf(legend, "ambient temperature:       %e (deg C)\n", T_amb);
fprintf(legend, "sample dimensions, XMAX:   %e (meters)\n", XMAX);
fprintf(legend, "                   XMIN:   %e (meters)\n", XMIN);
fprintf(legend, "                   YMAX:   %e (meters)\n", YMAX);
fprintf(legend, "                   YMIN:   %e (meters)\n", YMIN);
fprintf(legend, "                   ZMAX:   %e (meters)\n", ZMAX);
fprintf(legend, "                   ZMIN:   %e (meters)\n", ZMIN);
fprintf(legend, "spatial discretization, x: %e (meters)\n", dx);
fprintf(legend, "                        y: %e (meters)\n", dy);
fprintf(legend, "                        z: %e (meters)\n", dz);
fprintf(legend, "temporal discretization:   %e (seconds)\n", dt);

fprintf(legend, "\nProgram:\n");
fprintf(legend,
"power    time     beam radius        position radius     position angle\n");
fprintf(legend,
"(W)      (sec)    (mm)               (mm)                (revolutions)\n");

/* these are the customary coef.s used in FDM diffusion problems */
sx = Kappa * dt / dx / dx;
sy = Kappa * dt / dy / dy;
sz = Kappa * dt / dz / dz;

/* zero out maximum values */
for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
  {
    TM[i][j] = 0.0;
    GM[i][j] = 0.0;
    GT[i][j] = 0.0;
  }

/* use ambient temperature as the initial conditions */
for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
    for (k = 1; k <= NZ; k++)
      T0[i][j][k] = T_amb;

/* this value of tchange forces the first set of parameters to be read */
tchange = 0.0;
pstep = 0;

/* this is the main loop which carries the model forward in time. 60
   seconds is an arbitrary upper limit to keep the model from running
```

```
   forever in the case of an error */
 for (t = 0; t <= 60; t += dt, tcount++)
 {
  /* check to see if it is time to read new parameters
     this should always be true for first iteration */
  if (t >= tchange)
  {

    pstep++;

    /* read in new parameters */
    scanf("%f %f %f %f %f %f %f %f",
          &w, &duration, &r0, &r1, &rho0, &rho1, &phi0, &phi1);
    /* if duration is entered as zero then exit the time loop */
    if (duration == 0.0)
        break;

    /* print to legend text file */
    fprintf(legend, "%5.3e %5.3e %5.3e %5.3e %5.3e %5.3e %5.3e
%5.3e\n",
             w, duration, r0, r1, rho0, rho1, phi0, phi1);

    /* calculate next time for a change of parameters */
    tchange += duration;

    /* convert input data from millimeters to meters */
    r0 *= 1.0e-3;
    r1 *= 1.0e-3;
    rho0 *= 1.0e-3;
    rho1 *= 1.0e-3;

    /* convert input data from revolutions to radians */
    phi0 *= (2.0 * PI);
    phi1 *= (2.0 * PI);

    /* calculate beam radius increment */
    dr = (r1 - r0) / (duration / dt);

    /* calculate beam position radius increment */
    drho = (rho1 - rho0) / (duration / dt);

    /* calculate beam position angle increment */
    dphi = (phi1 - phi0) / (duration / dt);

  }
```

```
/* calculate beam position in cartesian coordinates */
cx = rho0 * cos(phi0);
cy = rho0 * sin(phi0);

/* set boundary conditions on z = ZMIN and z = ZMAX planes */
for (i = 1; i <= NX; i++)
 for (j = 1; j <= NY; j++)
 {
  /* calculate positions for q_source() */
  x = XMIN + dx * (float)(i - 1);
    y = YMIN + dy * (float)(j - 1);

    /* this is the condition on the plane z = ZMIN which represents
    the incident beam. It comes from the fundamental eq. for
    thermal conductivity: flux = - kc * dT/dz.  Ref. Carslaw &
    Jaeger, Conduction of Heat in Solids, Oxford, London, 1948. p7.
    The derivative is replaced by the FDM equivalent, then the
    expression is solved for the Temperaure at the advanced
    time step. */
  T0[i][j][0] = T0[i][j][2]
              + 2.0 * dz * q_source(x - cx, y - cy, r0*r0, w) / kc;

  /* the condition at z = ZMAX is the second derivative of
    temperature with respect to z is set equal to zero. */
  T0[i][j][NZ + 1] = 2.0 * T0[i][j][NZ] - T0[i][j][NZ - 1];
 }

/* set boundary conditions on y = YMIN and y = YMAX planes */
for (i = 1; i <= NX; i++)
 for (k = 1; k <= NZ; k++)
 {
  /* the conditions at y = YMAX and y = YMIN are the second
    derivative of temperature with respect to y is set
    equal to zero. */
  T0[i][0][k] = 2.0 * T0[i][1][k] - T0[i][2][k];
  T0[i][NY + 1][k] = 2.0 * T0[i][NY][k] - T0[i][NY - 1][k];
 }

/* set boundary conditions on x = XMIN and x = XMAX planes */
for (j = 1; j <= NY; j++)
 for (k = 1; k <= NZ; k++)
 {
  /* the conditions at x = XMAX and x = XMIN are the second
    derivative of temperature with respect to x is set
```

```
        equal to zero. */
      T0[0][j][k] = 2.0 * T0[1][j][k] - T0[2][j][k];
      T0[NX + 1][j][k] = 2.0 * T0[NX][j][k] - T0[NX - 1][j][k];
    }

   /* this is where the volume of future values is calculated from the
      volume of known values.  The basic diffusion equation:
      dT/dt = Kappa*"del squared"T is discretized using the standard
      explicit FDM, and then the result is solved for the temperature
      at the advanced time.  (Fletcher p250) */
   for (i = 1; i <= NX; i++)
    for (j = 1; j <= NY; j++)
     for (k = 1; k <= NZ; k++)
     {
      T1[i][j][k] = sx * (T0[i - 1][j][k] + T0[i + 1][j][k])
      + sy * (T0[i][j - 1][k] + T0[i][j + 1][k])
      + sz * (T0[i][j][k - 1] + T0[i][j][k + 1])
      + (1.0 - 2.0 * (sx + sy + sz)) * T0[i][j][k];
     }
/************************************************************/
/* at this point print out any results of interest
  for each time step */
/*   printf("%e %e %e\n", t, T0[(NX+1)/2][(NY+1)/2][1], r0); */
/************************************************************/

   /* save maximum values */
   k = 1;
   for (i = 1; i <= NX; i++)
    for (j = 1; j <= NY; j++)
    {
        if (T0[i][j][k] > TM[i][j])
          TM[i][j] = T0[i][j][k];

        gradx = (T0[i + 1][j][k] - T0[i - 1][j][k]) / (2.0 * dx);
        grady = (T0[i][j + 1][k] - T0[i][j - 1][k]) / (2.0 * dy);
        surfgrad = sqrt(gradx*gradx + grady*grady);
        if (surfgrad > GM[i][j])
        {
         GM[i][j] = surfgrad;
         GT[i][j] = T0[i][j][k];
        }
    }

   /* now copy future values over current values */
   for (i = 1; i <= NX; i++)
```

```
    for (j = 1; j <= NY; j++)
      for (k = 1; k <= NZ; k++)
        T0[i][j][k] = T1[i][j][k];

  /* increment beam radius */
  r0 += dr;

  /* increment beam position radius */
  rho0 += drho;

  /* increment beam position angle */
  phi0 += dphi;

 } /* <== this marks the end of the time loop */

 /* close legend text file */
 fclose(legend);

/*********************************************/
/* at this point print out any final results */

 dataout = fopen("lasttemp.dat", "w");
 k = 1;
 for (i = 1; i <= NX; i++)
  {
   x = XMIN + dx * (float)(i - 1);
   for (j = 1; j <= NY; j++)
   {
    y = YMIN + dy * (float)(j - 1);
      fprintf(dataout, "%e %e %e\n", x, y, T1[i][j][k]);
   }
  }
 fclose(dataout);
/*
 dataout = fopen("max_grad.dat", "w");
 for (i = 1; i <= NX; i++)
  {
   x = XMIN + dx * (float)(i - 1);
   for (j = 1; j <= NY; j++)
   {
    y = YMIN + dy * (float)(j - 1);
      fprintf(dataout, "%e %e %e\n", x, y, GM[i][j]);
   }
  }
 fclose(dataout);
```

```
 dataout = fopen("maxgtemp.dat", "w");
 for (i = 1; i <= NX; i++)
  {
   x = XMIN + dx * (float)(i - 1);
   for (j = 1; j <= NY; j++)
   {
    y = YMIN + dy * (float)(j - 1);
      fprintf(dataout, "%e %e %e\n", x, y, GT[i][j]);
   }
  }
 fclose(dataout);
*/
 dataout = fopen("tempinfo.dat", "w");
 for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
   for (k = 1; k <= NZ; k++)
      fprintf(dataout, "%e\n", T0[i][j][k] - T_amb);

 for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
   for (k = 1; k <= NZ; k++)
   {
      gradx = (T0[i + 1][j][k] - T0[i - 1][j][k]) / (2.0 * dx);
      grady = (T0[i][j + 1][k] - T0[i][j - 1][k]) / (2.0 * dy);
      gradz = (T0[i][j][k + 1] - T0[i][j][k - 1]) / (2.0 * dz);
      fprintf(dataout, "%e\n%e\n%e\n", gradx, grady, gradz);
   }
 fclose(dataout);

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

} /* <== this marks the end of the "executable" statements */

/* this is the laser power function described above */
float q_source(x, y, r0s, w)
float x, y, r0s, w;
{
   return (w / PI / r0s * exp(-1.0 * (x*x + y*y) / r0s));
}
```

A.1 Thermoelastic Displacement Model Code

The thermoelastic displacement code solves the sparse banded system as described in Chapter 4 using the SOR method. The functions makeEqns(), fillb(), and fillu() take the information generated by the temperature model and create the matrix, source, and solution vectors. The main function then repeated calls SOR() until the convergence condition is satisfied.

```
#include <stdio.h>
#include <math.h>
#define NX 41
#define NY 41
#define NZ 11
#define SIZE (NX*NY*NZ*3)
#define NUMEQS 3
#define MAXTERMS 15

#define XMIN -7.50e-3
#define XMAX 7.50e-3
#define YMIN -7.50e-3
#define YMAX 7.50e-3
#define ZMIN 0.0e-3
#define ZMAX 5.0e-3

#define TRUE 1
#define FALSE 0

long vectorindex();
float duxdx(), duydx(), duzdx(),
    duxdy(), duydy(), duzdy(),
    duxdz(), duydz(), duzdz();

int terms[NUMEQS*2],
  isub[NUMEQS*2][MAXTERMS],
  jsub[NUMEQS*2][MAXTERMS],
  ksub[NUMEQS*2][MAXTERMS],
  xyzsub[NUMEQS*2][MAXTERMS];

float coef[NUMEQS*2][MAXTERMS],
    u[SIZE+1],
    b[SIZE+1],
    deltatemp[NX + 1][NY + 1][NZ + 1];
```

```
float E, v, alpha, G, lambda, dx, dy, dz;

void main()
{
 void fillu(), makeEqns(), fillb(), SOR(), writeu();
 float norm();

 float newnorm, oldnorm;
 int done;

 E = 1.10e11;
 v = 0.35;
 alpha = 4.7e-6;
 G = E / (2.0 * (1.0 + v));
 lambda = v * E / ((1.0 + v) * (1.0 - 2.0 * v));
 dx = (XMAX - XMIN) / (float)(NX - 1);      /* meters */
 dy = (YMAX - YMIN) / (float)(NY - 1);
 dz = (ZMAX - ZMIN) / (float)(NZ - 1);

 makeEqns();
 fillu();
 fillb();

 done = FALSE;
 oldnorm = 0.0;
 while (!done)
 {
  newnorm = norm();
  if (oldnorm)
   done = ( fabs((oldnorm - newnorm) / oldnorm) < 0.0002 );

  SOR();
  printf("%f\n",newnorm);
  oldnorm = newnorm;
 }
 writeu();

 return;
}

/* This function is used to generate the matrix of coeficients. */
void makeEqns()
{
 void insert();
```

```
int eq, term;
float Ax, Ay, Az, Bx, By, Bz, Cxy, Cyz, Czx, D;

for (eq = 0; eq < (NUMEQS * 2); eq++)
 for (term = 0; term < MAXTERMS; term++)
 {
  coef[eq][term] = 0.0;
  isub[eq][term] = 0;
  jsub[eq][term] = 0;
  ksub[eq][term] = 0;
  xyzsub[eq][term] = 0;
 }

Ax = (2.0 * G + lambda) / dx / dx;
Ay = (2.0 * G + lambda) / dy / dy;
Az = (2.0 * G + lambda) / dz / dz;

Bx = G / dx / dx;
By = G / dy / dy;
Bz = G / dz / dz;

Cxy = (lambda + G) / 4.0 / dx / dy;
Cyz = (lambda + G) / 4.0 / dy / dz;
Czx = (lambda + G) / 4.0 / dz / dx;

D = lambda / (2.0 * G + lambda);

term = 0;
insert(0,  0,  0,  0,  0, -2.0 * (Ax + By + Bz), term++);  /* x eqn */
insert(0,  1,  0,  0,  0, Ax, term++);
insert(0, -1,  0,  0,  0, Ax, term++);
insert(0,  0,  1,  0,  0, By, term++);
insert(0,  0, -1,  0,  0, By, term++);
insert(0,  0,  0,  1,  0, Bz, term++);
insert(0,  0,  0, -1,  0, Bz, term++);
insert(0,  1,  1,  0,  1, Cxy, term++);
insert(0,  1, -1,  0,  1, -Cxy, term++);
insert(0, -1,  1,  0,  1, -Cxy, term++);
insert(0, -1, -1,  0,  1, Cxy, term++);
insert(0,  1,  0,  1,  2, Czx, term++);
insert(0,  1,  0, -1,  2, -Czx, term++);
insert(0, -1,  0,  1,  2, -Czx, term++);
insert(0, -1,  0, -1,  2, Czx, term++);
terms[0] = term;
```

```
term = 0;
insert(1,  0,  0,  0,  1, -2.0 * (Ay + Bz + Bx), term++);       /* y eqn */
insert(1,  0,  1,  0,  1, Ay, term++);
insert(1,  0, -1,  0,  1, Ay, term++);
insert(1,  0,  0,  1,  1, Bz, term++);
insert(1,  0,  0, -1,  1, Bz, term++);
insert(1,  1,  0,  0,  1, Bx, term++);
insert(1, -1,  0,  0,  1, Bx, term++);
insert(1,  0,  1,  1,  2, Cyz, term++);
insert(1,  0,  1, -1,  2, -Cyz, term++);
insert(1,  0, -1,  1,  2, -Cyz, term++);
insert(1,  0, -1, -1,  2, Cyz, term++);
insert(1,  1,  1,  0,  0, Cxy, term++);
insert(1,  1, -1,  0,  0, -Cxy, term++);
insert(1, -1,  1,  0,  0, -Cxy, term++);
insert(1, -1, -1,  0,  0, Cxy, term++);
terms[1] = term;

term = 0;
insert(2,  0,  0,  0,  2, -2.0 * (Az + Bx + By), term++);       /* z eqn */
insert(2,  0,  0,  1,  2, Az, term++);
insert(2,  0,  0, -1,  2, Az, term++);
insert(2,  1,  0,  0,  2, Bx, term++);
insert(2, -1,  0,  0,  2, Bx, term++);
insert(2,  0,  1,  0,  2, By, term++);
insert(2,  0, -1,  0,  2, By, term++);
insert(2,  1,  0,  1,  0, Czx, term++);
insert(2,  1,  0, -1,  0, -Czx, term++);
insert(2, -1,  0,  1,  0, -Czx, term++);
insert(2, -1,  0, -1,  0, Czx, term++);
insert(2,  0,  1,  1,  1, Cyz, term++);
insert(2,  0,  1, -1,  1, -Cyz, term++);
insert(2,  0, -1,  1,  1, -Cyz, term++);
insert(2,  0, -1, -1,  1, Cyz, term++);
terms[2] = term;

term = 0;                                   /* x eqn @ surf*/
insert(3,  0,  0,  0,  0,
        Czx * 2.0 * D * dz / dx  - 2.0 * (Ax + By + Bz), term++);
insert(3,  1,  0,  0,  0, Ax, term++);
insert(3, -1,  0,  0,  0, Ax, term++);
insert(3,  0,  1,  0,  0, By, term++);
insert(3,  0, -1,  0,  0, By, term++);
insert(3,  0,  0,  1,  0, 2.0 * Bz, term++);
insert(3,  1,  0,  0,  2, Bz * dz / dx, term++);
```

```
insert(3, -1,  0,  0,  2, -Bz * dz / dx, term++);
insert(3,  1,  1,  0,  1, Cxy - Czx * D * dz / dx, term++);
insert(3,  1, -1,  0,  1, -Cxy + Czx * D * dz / dx, term++);
insert(3, -1,  1,  0,  1, -Cxy + Czx * D * dz / dx, term++);
insert(3, -1, -1,  0,  1, Cxy - Czx * D * dz / dx, term++);
insert(3, -2,  0,  0,  0, -Czx * D * dz / dx, term++);
insert(3,  2,  0,  0,  0, -Czx * D * dz / dx, term++);
terms[3] = term;

term = 0;                                   /* y eqn @ surf*/
insert(4,  0,  0,  0,  1,
        Cyz * 2.0 * D * dz / dy  - 2.0 * (Ay + Bx + Bz), term++);
insert(4,  0,  1,  0,  1, Ay, term++);
insert(4,  0, -1,  0,  1, Ay, term++);
insert(4,  1,  0,  0,  1, Bx, term++);
insert(4, -1,  0,  0,  1, Bx, term++);
insert(4,  0,  0,  1,  1, 2.0 * Bz, term++);
insert(4,  0,  1,  0,  2, Bz * dz / dy, term++);
insert(4,  0, -1,  0,  2, -Bz * dz / dy, term++);
insert(4,  1,  1,  0,  0, Cxy - Cyz * D * dz / dy, term++);
insert(4,  1, -1,  0,  0, -Cxy + Cyz * D * dz / dy, term++);
insert(4, -1,  1,  0,  0, -Cxy + Cyz * D * dz / dy, term++);
insert(4, -1, -1,  0,  0, Cxy - Cyz * D * dz / dy, term++);
insert(4,  0, -2,  0,  1, -Cyz * D * dz / dy, term++);
insert(4,  0,  2,  0,  1, -Cyz * D * dz / dy, term++);
terms[4] = term;

term = 0;                                   /* z eqn @ surf */
insert(5,  0,  0,  0,  2,
        2.0 * (Czx * dz / dx + Cyz * dz / dy - (Az + Bx + By)), term++);
insert(5,  0,  0,  1,  2, 2.0 * Az, term++);
insert(5,  1,  0,  0,  0, Az * D * dz / dx, term++);
insert(5, -1,  0,  0,  0, -Az * D * dz / dx, term++);
insert(5,  0,  1,  0,  1, Az * D * dz / dy, term++);
insert(5,  0, -1,  0,  1, -Az * D * dz / dy, term++);
insert(5,  1,  0,  0,  2, Bx, term++);
insert(5, -1,  0,  0,  2, Bx, term++);
insert(5,  0,  1,  0,  2, By, term++);
insert(5,  0, -1,  0,  2, By, term++);

insert(5, -2,  0,  0,  2, -Czx * dz / dx, term++);
insert(5,  2,  0,  0,  2, -Czx * dz / dx, term++);
insert(5,  0, -2,  0,  2, -Cyz * dz / dy, term++);
insert(5,  0,  2,  0,  2, -Cyz * dz / dy, term++);
terms[5] = term;
```

```
 return;
}

void insert(eq, i, j, k, xyz, c, term)
int eq, i, j, k, xyz;
float c;
{

 isub[eq][term] = i;
 jsub[eq][term] = j;
 ksub[eq][term] = k;
 xyzsub[eq][term] = xyz;
 coef[eq][term] = c;

 return;
}

/* This function loads the b vector from a file of temperature gradients
   generated by the temperature model. */

void fillb()
{
 int i, j, k, eq;
 float grad, temp, c0, c1;
 FILE *handle;

 c0 = alpha * E / (1.0 - 2.0 * v);
 c1 = (lambda + G) / (2.0 * G + lambda);

 handle = fopen("tempinfo.dat", "r");

 for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
    for (k = 1; k <= NZ; k++)
        fscanf(handle, "%e", &deltatemp[i][j][k]);

 for (i = 1; i <= NX; i++)
  for (j = 1; j <= NY; j++)
  {
    k = 1;

    eq = 0;
    fscanf(handle, "%e", &grad);
    b[vectorindex(i, j, k, eq)] = (1.0 - c1) * c0 * grad;
```

```
    eq = 1;
    fscanf(handle, "%e", &grad);
    b[vectorindex(i, j, k, eq)] = (1.0 - c1) * c0 * grad;

    eq = 2;
    fscanf(handle, "%e", &grad);
    b[vectorindex(i, j, k, eq)] = c0 * grad +
                                  2.0 / dz * c0 * deltatemp[i][j][k];

    for (k = 2; k <= NZ; k++)
        for (eq = 0; eq < NUMEQS; eq++)
        {
         fscanf(handle, "%e", &grad);
         b[vectorindex(i, j, k, eq)] = c0 * grad;
        }
  }
 fclose(handle);
 return;
}

/* This function loads the u vector with an initial estimate. */
void fillu()
{
 long m;

 for (m = 1; m <= SIZE; m++)
  u[m] = 0.0;

 return;
}

/* This function writes the u vector to a file. */
void writeu()
{
 int i, j, k, n;
 float sxx, syy, szz, sxz, exx, eyy, ezz, exz, dT;

 j = (NY / 2) + 1;

 for (k = 1; k <= 5; k++)
 {
  for (i = 1; i <= NX; i++)
  {
   dT = deltatemp[i][j][k];
```

```
    sxx = lambda * (duxdx(i, j, k) + duydy(i, j, k) + duzdz(i, j, k));
    sxx += 2.0 * G * duxdx(i, j, k);
    sxx -= alpha * E / (1.0 - 2.0 * v) * dT;

    syy = lambda * (duxdx(i, j, k) + duydy(i, j, k) + duzdz(i, j, k));
    syy += 2.0 * G * duydy(i, j, k);
    syy -= alpha * E / (1.0 - 2.0 * v) * dT;

    szz = lambda * (duxdx(i, j, k) + duydy(i, j, k) + duzdz(i, j, k));
    szz += 2.0 * G * duzdz(i, j, k);
    szz -= alpha * E / (1.0 - 2.0 * v) * dT;

    sxz = G * (duzdx(i, j, k) + duxdz(i, j, k));

    exx = duxdx(i, j, k) + alpha * dT;

    eyy = duydy(i, j, k) + alpha * dT;

    ezz = duzdz(i, j, k) + alpha * dT;

    exz = duzdx(i, j, k) + duxdz(i, j, k);

    printf("%e %e %e %e %e %e %e %e %e %e %e\n",
           dT, u[vectorindex(i, j, k, 0)],
           u[vectorindex(i, j, k, 1)],
           u[vectorindex(i, j, k, 2)],
           sxx, syy, szz, sxz, exx, eyy, ezz, exz);
   }
   printf("\n");
  }
  return;
}

void SOR()
{
  long m, n;
  int i, j, k, eq, term, i0, j0, k0, eq0;
  float du, accel;

  m = 0;

  accel = 1.5;

  for (i = 1; i <= NX; i++)
```

```
   for (j = 1; j <= NY; j++)
     for (k = 1; k <= NZ; k++)
         for (eq = 0; eq < NUMEQS; eq++)
         {
          m++;
          du = b[m];

          if (k == 1)
            eq0 = eq + NUMEQS;
          else
            eq0 = eq;

          for (term = 0; term < terms[eq0]; term++)
          {
           i0 = i + isub[eq0][term];
           j0 = j + jsub[eq0][term];
           k0 = k + ksub[eq0][term];

           if ( (i0 >= 1) && (j0 >= 1) && (i0 <= NX) && (j0 <= NY) &&
                 (k0 <= NZ) )
           {
            n = vectorindex(i0, j0, k0, xyzsub[eq0][term]);
            du -= coef[eq0][term] * u[n];
           }
          }

          du *= accel / coef[eq0][0];

          u[m] += du;
         }
  return;
}

float norm()
{
 float ret;
 long m;
 ret = 0.0;
 for (m = 1; m <= SIZE; m++)
   ret += fabs(u[m]);

 return (ret);
}

float duxdx(i, j, k)
```

```
int i, j, k;
{
 float ulow, uhigh;

 if (i == 1)
  ulow = 0.0;
 else
  ulow = u[vectorindex(i - 1, j, k, 0)];

 if (i == NX)
  uhigh = 0.0;
 else
  uhigh = u[vectorindex(i + 1, j, k, 0)];

 return ( (uhigh - ulow) / (2.0 * dx) );
}

float duydx(i, j, k)
int i, j, k;
{
 float ulow, uhigh;

 if (i == 1)
  ulow = 0.0;
 else
  ulow = u[vectorindex(i - 1, j, k, 1)];

 if (i == NX)
  uhigh = 0.0;
 else
  uhigh = u[vectorindex(i + 1, j, k, 1)];

 return ( (uhigh - ulow) / (2.0 * dx) );
}

float duzdx(i, j, k)
int i, j, k;
{
 float ulow, uhigh;

 if (i == 1)
  ulow = 0.0;
 else
  ulow = u[vectorindex(i - 1, j, k, 2)];
```

```
 if (i == NX)
   uhigh = 0.0;
 else
   uhigh = u[vectorindex(i + 1, j, k, 2)];

 return ( (uhigh - ulow) / (2.0 * dx) );
}

float duydy(i, j, k)
int i, j, k;
{
 float ulow, uhigh;

 if (j == 1)
   ulow = 0.0;
 else
   ulow = u[vectorindex(i, j - 1, k, 1)];

 if (j == NY)
   uhigh = 0.0;
 else
   uhigh = u[vectorindex(i, j + 1, k, 1)];

 return ( (uhigh - ulow) / (2.0 * dy) );
}

float duxdy(i, j, k)
int i, j, k;
{
 float ulow, uhigh;

 if (j == 1)
   ulow = 0.0;
 else
   ulow = u[vectorindex(i, j - 1, k, 0)];

 if (j == NY)
   uhigh = 0.0;
 else
   uhigh = u[vectorindex(i, j + 1, k, 0)];

 return ( (uhigh - ulow) / (2.0 * dy) );
}

float duzdy(i, j, k)
```

```
int i, j, k;
{
 float ulow, uhigh;

 if (j == 1)
  ulow = 0.0;
 else
  ulow = u[vectorindex(i, j - 1, k, 2)];

 if (j == NY)
  uhigh = 0.0;
 else
  uhigh = u[vectorindex(i, j + 1, k, 2)];

 return ( (uhigh - ulow) / (2.0 * dy) );
}

float duzdz(i, j, k)
int i, j, k;
{
 float ulow, uhigh, ret;

 if (k == 1)
 {
  ret = -lambda * (duxdx(i, j, k) + duydy(i, j, k));
  ret += E * alpha * deltatemp[i][j][k] / (1.0 - 2.0 * v);
  ret /= (lambda + 2.0 * G);
 }
 else
 {
  ulow = u[vectorindex(i, j, k - 1, 2)];

  if (k == NZ)
   uhigh = 0.0;
  else
   uhigh = u[vectorindex(i, j, k + 1, 2)];

  ret = (uhigh - ulow) / (2.0 * dz);
 }
 return (ret);
}

float duxdz(i, j, k)
int i, j, k;
{
```

```
 float ulow, uhigh, ret;

 if (k == 1)
  ret = -1.0 * duzdx(i, j, k);
 else
 {
  ulow = u[vectorindex(i, j, k - 1, 0)];

  if (k == NZ)
   uhigh = 0.0;
  else
   uhigh = u[vectorindex(i, j, k + 1, 0)];

  ret = (uhigh - ulow) / (2.0 * dz);
 }
 return (ret);
}

float duydz(i, j, k)
int i, j, k;
{
 float ulow, uhigh, ret;

 if (k == 1)
  ret = -1.0 * duzdy(i, j, k);
 else
 {
  ulow = u[vectorindex(i, j, k - 1, 1)];

  if (k == NZ)
   uhigh = 0.0;
  else
   uhigh = u[vectorindex(i, j, k + 1, 1)];

  ret = (uhigh - ulow) / (2.0 * dz);
 }
 return (ret);
}

long vectorindex(i, j, k, xyz)
int i, j, k, xyz;
{
 long ret;

 ret = ( (i - 1) * NY * NZ * NUMEQS
```

```
    + (j - 1) * NZ * NUMEQS
    + (k - 1) * NUMEQS )
    + xyz + 1;

 return (ret);
}
```

WORKS CITED

1. Duley, W. W. *CO_2 Lasers: Effects and Applications.* Quantum Electronics Series. New York: Academic Press, 1976.

2. Welsh, Lisa P., Judah A. Tuchman, and Irving P. Herman. "The Importance of Thermal Stresses and Strains Induced in Laser Processes with Focused Gaussian Beams." *J. Appl. Phys.* 64 (1988): 6274-6286.

3. Petitbon A., D. Guignot, R. Fisher, and M. Guillemot. "Laser Surface Treatment of Ceramic Coatings." *Materials Science and Engineering.* A121 (1989): 545-548.

4. Yi, Jay J. L. and Peter R. Strutt. "Surface Modification of SiO_2 Glass by Laser Processing." *J. Non-Crystalline Solids.* 120 (1990): 283-287.

5. Spann, J. R., et al. "Laser Processing of Ceramics." *Emergent Process Methods for High Technology Ceramics.* R. F. Davis, H. Palmour III, and R. L. Porter, eds. New York: Plenum Press, 1984.

6. Ready, J. F. "Effects Due to Absorption of Laser Radiation." *J. Appl. Phys.* 36 (1964): 462-468.

7. Lax, M. "Temperature Rise Induced by a Laser Beam." *J. Appl. Phys.* 48 (1977): 3919-3924.

8. Bentini, G., L. Correra, and C. Donolato. "Defects Introduced in Silicon Wafers During Rapid Isothermal Annealing: Thermoelastic and Thermoplastic Effects." *J. Appl. Phys.* 56 (1984): 2922-2929.

9. Glasser, F. P. and Xiping Jing. "Laser Melting of Refractory Al_2O_3-ZrO_2 Ceramics." *Br. Ceram. Trans. J.* 91 (1992): 195-198.

10. Dallaire, S. and P. Cielo. "Laser Spot Glazing of Whitewares." *Ceramic Engineering and Science Proceedings.* 5 (1984): 936-940.

11. Nagai, Akira and Yoshitaka Kimura. "Synthetic Raw Materials for Ceramics." *Advanced Technical Ceramics.* Shigeyuki Somiya, ed. San Diego: Academic Press, Inc., 1989.

12. Kingery, W. D. *Introduction to Ceramics.* Wiley Series on the Science and Technology of Materials. New York: John Wiley & Sons, Inc., 1960.

WORKS CITED
(continued)

13. Chipman, R. D. and W. J. Knapp. "Elementary Considerations for Structural Use." *Modern Ceramics: Some Principles and Concepts.* J. E. Hove and W. C. Riley, eds. New York: John Wiley & Sons, Inc., 1965.

14. Fletcher, C. A. J. *Computational Techniques for Fluid Dynamics 1: Fundamental and General Techniques.* Vol. 1. Springer Series in Computational Physics. New York: Springer-Verlag, 1988.

15. Chou, Pei Chi and Nicholas J. Pagano. *Elasticity: Tensor, Dyadic, and Engineering Approaches.* New York: Dover Publications, Inc., 1967.

16. Smith, G. D. *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* 3rd ed. Oxford Applied Mathematics and Computing Science Series. Oxford: Oxford University Press, 1985.

17. Dahlquist, G. and A. Bjorck. *Numerical Methods.* Englewood Cliffs: Prentice Hall, 1974.

18. Chu, Gorgon P. K. "Microstructure of Complex Ceramics." *Ceramic Microstructures: Their Analysis, Significance, and Production.* R. M. Fulrath and J. A. Pask, eds. New York: John Wiley & Sons, Inc., 1966.