Adiabaticity of High Strain Rate Compression Testing Using the Split Hopkinson Pressure Bar Apparatus

FRANK STEWART WALTON
B.Sc., University of Lethbridge, 1975

A Thesis
Submitted to the Council on Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

LETHBRIDGE, ALBERTA
January 30, 1997

To my family,
Donna, Chris, Ben, Kevin and Matthew.

Abstract

In the development of explosively formed projectiles (EFPs), researchers are faced with the problem of testing prospective metals at high strain rates. So far it has been assumed that, relative to the cooling time, the deformation time is practically instantaneous indicating that the test is adiabatc: none of the heat generated within the metal is lost to conduction. In this paper we construct a model that subtracts out the effects of kinetic energy and uses specific heat as a function of temperature. In this way we can focus on the energy change in the specimen that can be attributed to temperature and determine just how adiabatic the high strain rate test is.

Acknowledgements

I am now convinced that the only people who can really appreciate an Acknowledgement page are those who have been through this process themselves, either in person or by proxy - my wife being the chief proxy.

I have learned a great deal about mathematics, about writng and rewriting and about myself. In Dr. Kaminski's office I have watched him rough out calculations that I have seen in many math textbooks but never really understood how they could be applied in other settings. My appreciation for having a solid foundation of math fundamentals has been entrenched. I have turned work into Dr. Kaminski many times only to have my careful editing come back even more carefully edited. He demanded my best, even when I was too tired to give it. At the onset I had a very rudimentary understanding of metal and its properites. Pat Gallagher faithfully answered my questions, occasionally meeting with me in his home over a plate of pizza. I first met Dr. Cowan as the instructor of an undergraduate math class I was taking to help sharpen my skills for graduate work. It was his encouragement that started me on this journey. All of my advisors: Dr. Kaminski, Dr. Cowan and Pat Gallagher have been very supportive.

To the University of Lethbridge, my advisors and those who volunteered to be on my examining committee I offer my thanks. It was a particularly challenging road for me but with their help I made it.

A_b	cross-sectional area of the bars
A_s	cross-sectional area of the specimen
c_e	elastic wave velocity
r_b	radius of the bars
r_s	radius of specimen
E	Young's modulus
ϵ	strain
IE	internal energy
ke	kinetic energy
ϵ_e	elastic strain
ϵ_p	plastic strain
ϵ_i	incident strain
ϵ_r	reflected strain
ϵ_t	transmitted strain
f	force
L_s	length of the specimen
L_b	length of the bars
m	mass
V	volume
ρ	density
ρ_s	specimen density
v	velocity
σ	stress
σ_i	incident stress
σ_r	reflected stress
σ_s	specimen stress
σ_t	transmitted stress
T	temperature
t	time
c_p	heat capacity at constant pressure
c_v	heat capacity at constant volume
k	thermal diffusivity constant
u_1	displacement of the incident bar/specimen interface
u_2	displacement of the specimen/transmitter bar interface
$\mathcal{P}$	Poisson's ratio
$\mathcal{P}_f$	Poisson's ratio for specimen material

Chapter 1

Introduction

For the past several years the Defence Research Establishment Suffield has been studying explosively formed projectiles (EFPs). Upon detonation, the metal used in such a device undergoes tremendous deformation at a high strain rate, but despite these harsh conditions, the integrity of the metal cannot be compromised. In order to test metals at the appropriate strain rates for use as an EFP, an apparatus known as a Split Hopkinson Pressure Bar is used. In this test, a small sample of the test material is sandwiched between two bars and compressed at a high strain rate. Of particular interest during the test is the amount of heat dissipated by the specimen during deformation. One may think that at very high rates the deformation would be so rapid the process would be adiabatic—no heat gain or loss during the test. This is the area explored in this thesis. We treat our specimen as a material with a continuous density distribution and use the basic principles of physics to study the forces on the particles and their resulting motion.[12]

To calculate the heat dissipated from the specimen into the adjoining bars we discretized the specimen and used first principles of motion and thermodynamics to build conservation equations. A simulation of the deformation was generated using a computer program. The results could be validated using data obtained from heat sensors placed along the bars during an actual test.

To follow the development of the equations and resulting computer algorithm, the reader needs to be familiar with partial differential equations and finite difference techniques. A brief review of each area is provided as required, but for those wishing a more extensive review we refer to Carrier and Pearson [5] and Lapidus and Pinder[15].

1.1 Stress and Strain

The strength of a metal is its ability to resist changing its size or shape when external forces are applied. When stress (σ), defined as force/unit area, is applied to a metal it changes shape. This change in shape is known as strain (ϵ) and is expressed as a ratio, $\epsilon = \Delta L/L_0$, where L_0 is the original length and ΔL is the change in length. Thus, a standard uniaxial tensile test involves stretching a prepared sample over a certain time period and measuring the change in length as a ratio to the original length. For a cylindrically shaped specimen there will be a corresponding reduction in cross-sectional area or transverse strain which we record as a change in diameter. Poisson's ratio ($\mathcal{P}$) is the ratio of this transverse strain to the axial longitudinal strain:

$$\mathcal{P} = \frac{\text{change in transverse dimensions}}{\text{change in longitudinal dimensions}} = \frac{\epsilon_T}{\epsilon_x}.$$

If the stress is not too great, the strain induced by the stress is not permanent and the metal returns to its original shape after the load is removed. However, if the stress is too great—above the elastic limit of the metal—the deformation is permanent. A typical stress versus strain plot is shown in Figure 1.1.

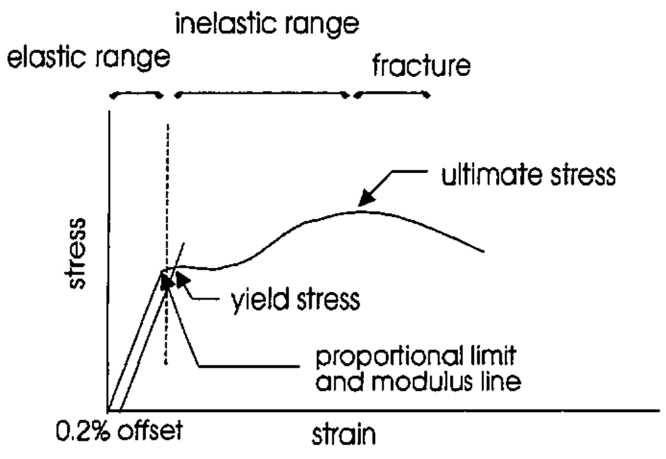

Figure 1.1: Stress vs Strain Plot

The elastic range corresponds to the linear portion of the graph. Within this elastic range, the ratio of stress to strain is known as Young's Modulus, or the modulus of elasticity, and is denoted $E = \sigma/\epsilon$ (also known as Hooke's Law). Hooke's Law states that stress (load) is directly proportional to strain (extension) up to a certain limit. This limit has been named the proportional limit. Often the proportional limit is extremely difficult to accurately determine so a small offset is used to calculate the yield strength. The yield strength shows the point at which the relationship between stress and strain is no longer constant. On continued straining, the applied stress will increase until the ultimate stress has been reached and then fall as fracture is approached.

A very important factor in such a test is the rate at which the load is being applied and consequently the strain rate taking place. The strain rate ($\dot{\epsilon}$) is the rate of change of strain with respect to time, $\dot{\epsilon}= d\epsilon/dt$, the units being inverse time (s^{-1}).

Since $\epsilon = \Delta L/L_0$ then

$$\frac{d\epsilon}{dt} = \frac{1}{L_0} \cdot \frac{dL}{dt} = \frac{v}{L_0}$$

where L is the length of the specimen of original length L_0, and v is the velocity at which the specimen is being deformed. To test materials at strain rates of the order of 10^4 s^{-1}, experimental techniques that make use of impact dynamics and wave propagation phenomena are employed. One common apparatus used for high strain rate testing is the Split Hopkinson Pressure Bar (SHPB)[10].

1.2 The Split Hopkinson Pressure Bar

The SHPB consists of a striker, an incident bar, a specimen and an output bar. To test the specimen, a pressure wave is induced in the incident bar by the striker bar, the size and duration of the pulse being controlled by the length and velocity of the striker bar. This wave causes a stress in the incident bar and an accompanying strain which is measured at strain gauge A, the midpoint of the incident bar (see Figure 1.2). At the incident bar/specimen interface the wave is partially reflected and the rest transmitted into the specimen. The wave moves through the specimen and is again partially reflected and partially transmitted at the specimen/output bar interface. The transmitted portion induces a stress and accompanying strain in the transmitter bar which is measured at strain gauge B—the midpoint of the transmitter bar. The reflected portion moves back along the incident bar and the resulting strain is again measured at strain gauge A. Knowing these strains, we are able to determine the strain history of the specimen which was not directly measurable.

There are numerous articles detailing the operation and analysis of the SHPB,

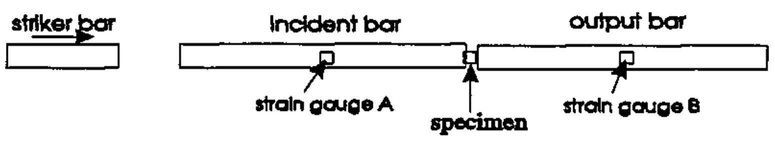

Figure 1.2: Split Hopkinson Pressure Bar, compression configuration

and since it is not the focus of this research, it is not necessary to go through those in detail here; the interested reader can consult [2] and [11] for additional details. We do however need to provide a brief account to augment the reader's background.

1.2.1 Overview and Analysis

The most fundamental assumption made about the SHPB apparatus is that the incident and transmitter bars remain elastic throughout the test. This assumption, along with testing techniques that call for, among other things, accurate alignment of the bars, the use of a lubricant between the bars and the specimen, and appropriate sizing of the components, provide a uniaxial, uniform deformation compression test that is mathematically predictable.

Upon impact by the striker bar, an elastic pressure wave begins to move along the incident bar. By considering the elastic rod as a coupled spring mass system, it can be shown that the velocity of this elastic wave (c_e) is $c_e^2 = E/\rho$ where ρ is the density. Gallagher [13] has shown that if an element of the rod were initially at rest, this velocity $(v) = \epsilon c_e$. Since displacement is the integral of velocity, in terms of incident, transmitted, and reflected pulses we have

$$u_1 = c_\epsilon \int_0^t (\epsilon_i(t) - \epsilon_r(t))dt,$$

and

$$u_2 = c_e \int_0^t \epsilon_t(t)dt,$$

where u_1 is the displacement at the incident bar/specimen interface, and u_2 is the displacement at the specimen/transmitter bar interface. The difference between u_1 and u_2 would be the specimen displacement and hence the specimen stain is

$$\epsilon_s = \frac{c_e}{L_s} \int_0^t (\epsilon_i(t) - \epsilon_r(t) - \epsilon_t(t))dt$$

where L_s is specimen length, and the strain rate for the specimen, $d\epsilon_s/dt$, would be the time derivative of this.

The stress on the specimen would be the average between the forces at each interface. Recall that

$$\sigma = f/A$$

where f is force and A is the area over which the force is applied. For the elastic bars we can also apply Hooke's law, $\sigma = \epsilon E$. Combining these shows the force at the incident bar/specimen interface to be

$$f_1 = (\epsilon_i(t) + \epsilon_r(t))\, E\, A,$$

and at the specimen/transmitter bar interface,

$$f_2 = \epsilon_t(t)\, E\, A,$$

so at the specimen,

$$\sigma_s = \frac{A_b}{A_s} \frac{[\epsilon_i(t) + \epsilon_r(t) + \epsilon_t(t)]\ E}{2}$$

where A_b/A_s is the ratio of the cross sectional area of the bar to the specimen. Furthermore, after a very short time, considered negligible when compared to the duration of the test, $f_1 = f_2$; the stress on the specimen will be in equilibrium. This is equivalent to

$$\epsilon_i(t) + \epsilon_r(t) = \epsilon_t(t),$$

which can be used to simplify the above formulas leaving

$$\begin{aligned} \sigma_s &= \frac{A}{A_s}\epsilon_t(t)\,E, \\ \epsilon_s &= \frac{-2c_e}{L_s}\int_0^t \epsilon_r(t)\,dt\,, \end{aligned}$$

and

$$\frac{d\epsilon_s}{dt} = \frac{-2c_e}{L_s}\epsilon_r(t)\,.$$

Thus the stress-strain behavior of the specimen is determined by measurements made on the elastic bars.

Figure 1.3 shows the strain history of a SHPB test. One can see how the three strains collectively induce a strain on the specimen. The incident strain is the initial strain induced in the incident bar. The reflected strain, working in the opposite direction, appears as a negative strain. Finally the transmitted strain is the amount of strain that passes through the specimen and reacts with the transmitter bar. Notice the slight recovery, the negative region on the specimen strain towards the end of the test, of the specimen after the load is dispersed. The specimen strain is depicted in Figure 1.4.

It should be noted that in the case of the specimen deformation during the split Hopkinson test, strains as high as 30 to 100% are taking place. Initially the strain

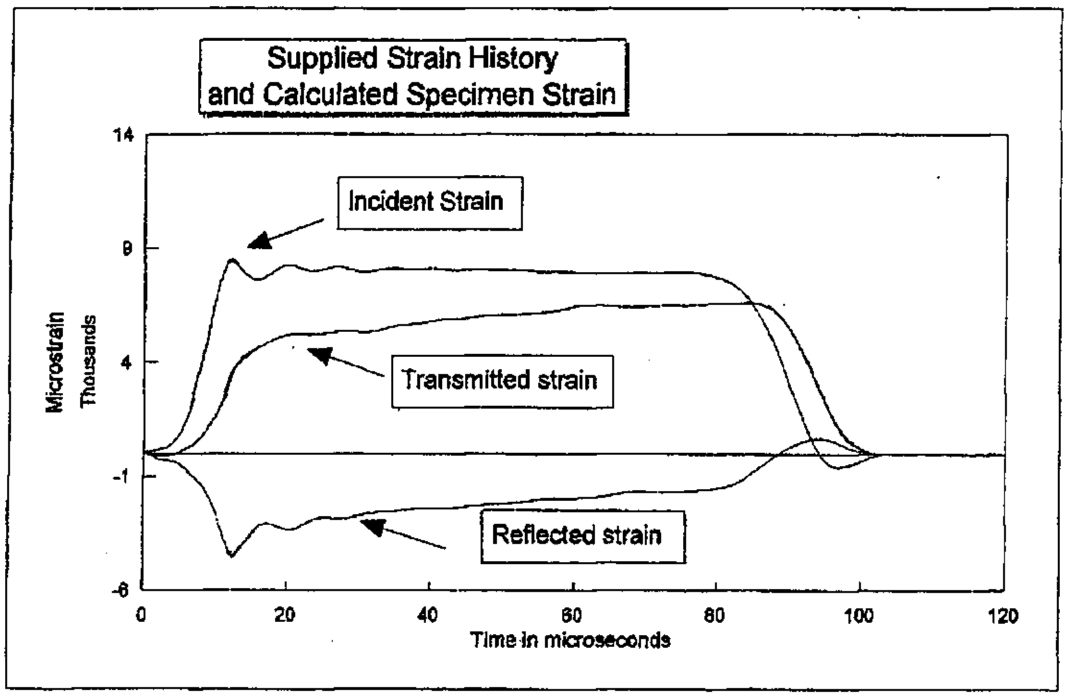

Figure 1.3: The three strains, ϵ_r, ϵ_t and ϵ_i.

on the metal is elastic. In this range every bit of energy that goes into deforming the specimen is stored as recoverable internal strain energy. This limit has been named the proportional limit and it is that point where the stress strain curve deviates from a straight line. Since this point of divergence is not easy to determine, another value, the yield strength, is used. The yield strength has a built in correction factor that states: a permanent deformation of 0.2% in the material is allowed. Beyond this point we have unrecoverable deformation. This 0.2% offset has been chosen arbitrarily as the maximum allowable permanent deformation before the material is considered to pass its yield strength [18]. As we proceed in our paper and calculate the strain history of the specimen, we are assuming the specimen begins to deform as soon as

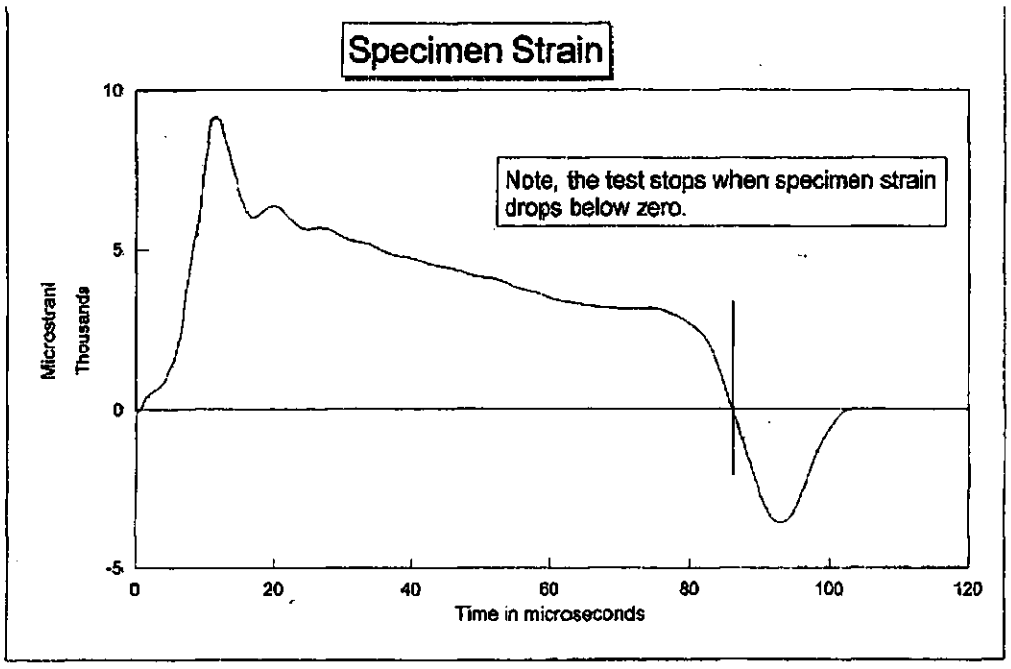

Figure 1.4: The specimen strain

the load is applied; see Figure 1.5.

As mentioned previously, this is not meant to be a detailed account of wave propagation in the SHPB. The references cited give more complete presentations on how the model must be corrected to allow for such matters as wave dispersion, friction, timing and inertia. Each of these areas offers an opportunity for more detailed examination. This paper focuses on the computation of temperature.

1.3 Heat and Temperature

Certainly metals are good conductors of heat but the process of heat transfer by conduction takes time. In the case of high strain rate testing, it has always been

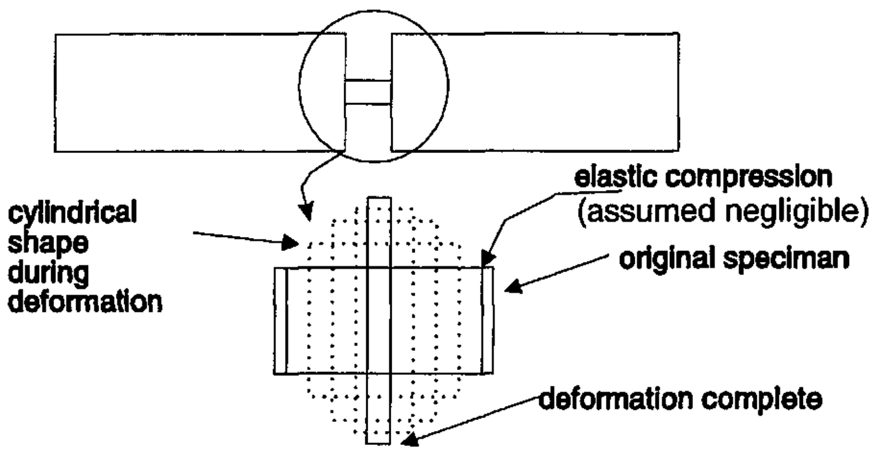

Figure 1.5: Specimen profile during deformation

assumed that since the deformation happens so quickly, on the order of $60\mu s$, there has not been time for the heat to dissipate from the localized area of deformation. If this internal heating then is not dissipated, the resulting rise in temperature softens the metal, which in turn allows for more deformation, which in turn increases the temperature, which further softens the metal and so on. A report prepared by the University of Waterloo [1] studied this self-feeding process and its end result, adiabatic shear banding. An *adiabatic* process is one in which no heat is dissipated outside the system.

The first law of thermodynamics states the algebraic sum of all the energy transfers across a system boundary must be equal to the change in energy of the system. Heat and work are the only forms of energy that may cross a system boundary. Under the assumption of an adiabatic process, all the work of deformation is converted to heat energy and can be used to calculate a change in temperature.

The work of deformation is

$$W = \int_{L_0}^{L_f} f\,dl,$$

where f is the applied load during which the specimen deformed from its original length L_o to its final length L_f . Since $\sigma(\epsilon) = f/A$ and $\epsilon = \Delta L/L_o$ we can rewrite the above formula as

$$W = A_s \cdot L_o \cdot \int_0^{\epsilon_f} \sigma(\epsilon)\,d\epsilon,$$

where ϵ_f is final specimen strain. The change in internal energy for one-dimensional analysis is

$$\Delta IE = \int_{T_i}^{T_f} \rho v_s c_p dT$$

and if we attribute the total change in energy to the work done in deformation we end up with

$$\Delta T = \frac{1}{\rho c_p} \int_0^{e_f} \sigma(\epsilon)\,d\epsilon\ ;$$

see [10].

But as we do this we err on two counts: the process is not necessarily adiabatic, and the heat capacity of the specimen is not constant. The focus of this research is predicting how adiabatic the compression test is.

Since the specimen is sandwiched between the incident and transmitter bars there will be diffusion in both directions. Due to this symmetry this paper only examines diffusion within the specimen and at the incident bar/specimen interface.

Chapter 2

Balancing Work

As previously mentioned, the only forms of energy that may cross a system boundary are heat and work and by the first law of thermodynamics the algebraic sum of all the energy transfers across the system boundary must be equal to the total change in energy of the system. Our system is the SHPB apparatus. The measured load being applied to the striker sends a pressure wave down the incident bar to the incident bar/specimen interface. Here one of two events can take place. Either the entire specimen is merely pushed along (only kinetic energy) or there is a division of the energy applied with a portion going to crush the metal past its yield strength into a plastic state. We, of course, have the latter and will use the following equation to build our model

$$E_{\text{supplied}} = E_{\text{work}} + KE.$$

The energy applied to the specimen must equal the energy used to overcome the yield strength, which we will attribute to heat energy, plus the energy used to accelerate the specimen material.

2.1 The Geometry of Deformation

The configuration of our specimen is that of a cylinder and, although not exactly correct, is assumed to maintain this shape throughout deformation. (A lubricant is applied at the specimen interfaces to reduce the effects of friction but even with these measures there is some distortion and the specimen radius is least constrained at the center leading to a characteristic barrel shape.) To study the interaction between elements of our specimen we will carve our sample up in the following way: we cut the cylinder into disks of uniform thickness, then take each disk and divide it into congruent pie shapes and finally take each pie shape and divide it into pieces of equal volume. See Figure 2.1.

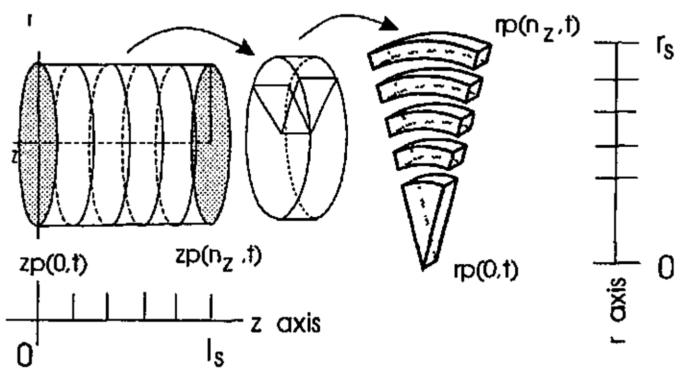

Figure 2.1: Carving up the specimen

As previously mentioned, specimen movement is tracked by measuring the three strains, $\varepsilon_i, \varepsilon_t$, and ε_r. Thus

$$\Delta l = (\varepsilon_i - \varepsilon_t - \varepsilon_r) \cdot l_s$$

and at each time step we can calculate a new specimen length, l_s, in cm. We will use the array $zp(i,t)$ for $0 \leq i \leq n_z$ and $t \geq 0$ to keep track of these longitudinal changes. Initially the incident bar/specimen interface is at $z = 0$ so $zp(0,0) = 0$ cm and at the specimen/transmitter bar interface $zp(n_z, 0) = l_s$ cm. During compression the position of the specimen/transmitter bar interface remains fixed so $zp(n_z, t) = l_s$ cm, while at the incident bar/specimen interface $zp(0,t)$ is increasing. At each time step the position of $zp(0,t)$ is calculated based on the strain data and the $zp(i,t)$, $i \neq n_z$, are updated to ensure each element has the same thickness. Therefore

$$zp(n_z, t) = l_s \text{ for all } t,$$

$$\Delta l(t) = (\varepsilon_i - \varepsilon_t - \varepsilon_r)(t) \cdot zp(n_z, t-1),$$

$$zp(0,t) = zp(0,t-1) + \Delta l(t),$$

and

$$zp(i,t) = zp(i-1,t) + \frac{zp(n_z,t)}{n_z}.$$

Clearly, as this compression is taking place the radius of the specimen is expanding. We will use $rp(j,t)$ for $0 \leq j \leq n_r$ and $t \geq 0$, to monitor the changing radial length of each element in centimeters. Making use of symmetry we will begin at the center of the specimen and assign $rp(0,0) = 0$ cm. Then $rp(n_r, 0)$ would denote the specimen radius, r_s, in cm. In this case it is the center of the specimen that remains fixed, $rp(0,t) = 0$ cm for all t, and the rest of the array must be updated at each time step. To solve for this array we impose the restriction that each element have the same volume. The first element is a sector whose volume is equal to

$$uvol = \frac{1}{n_{arcs}} \cdot \pi \cdot rp(1,t)^2 \cdot \Delta z$$

where n_{arcs} denotes the number of sectors into which the disk has been divided and $\Delta z = (zp(n_z,t) - zp(0,t))/n_z$. Then, if equal volume is be maintained for each element, as we increment j

$$\frac{\frac{1}{n_{arcs}} \cdot \pi \cdot rp(j,t)^2 \cdot \Delta z}{j} = \frac{\frac{1}{n_{arcs}} \cdot \pi \cdot rp(n_r,t)^2 \cdot \Delta z}{n_r},$$

whence

$$rp(j,t) = \sqrt{\frac{j}{n_r}} \cdot rp(n_r,t).$$

This is the formula we use to update $rp(j,t)$.

To compute kinetic energy we need to examine the forces on a typical element. In doing so we will treat each element as though it were a single particle, then calculate the velocity of the particle as deformation takes place. A natural representation point would be the centroid; see Figure 2.2.

Since we have symmetry with respect to θ and there is no twisting taking place during deformation, we can always orient an arc in the plane so that θ is zero. Thus only the centroids in the r and z directions need to be calculated. Along the z axis the centroid is simply the average of $zp(i,t)$ and $zp(i+1,t)$. Using $z(i,t)$ to store the z coordinate of the centroid,

$$z(i,t) = \frac{zp(i,t) + zp(i+1,t)}{2}.$$

To calculate the centroid in a radial direction we have, in Cartesian coordinates

$$\bar{x} = \frac{1}{\text{area of } R} \int\int_R x\, dA.$$

The area of our region R is

$$\frac{\pi \left(rp(j+1,t)^2 - rp(j,t)^2\right)}{n_{arcs}}.$$

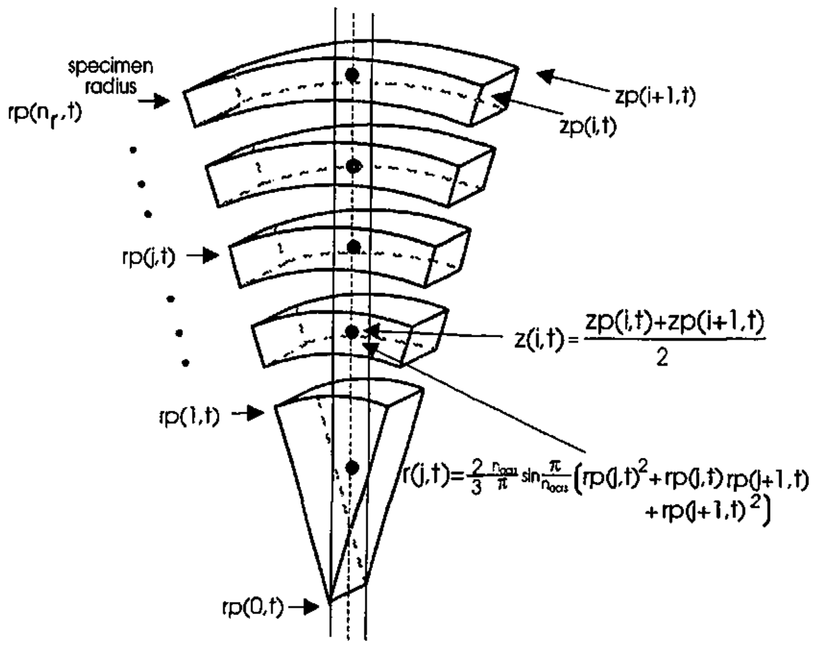
specimen radius
rp(n_r,t)
zp(i+1,t)
zp(i,t)
rp(j,t)
z(i,t)= (zp(i,t)+zp(i+1,t))/2
rp(1,t)
r(j,t)=2/3 n_arcs/π sin π/n_arcs (rp(j,t)²+rp(j,t)rp(j+1,t)+rp(j+1,t)²)
rp(0,t)

Figure 2.2: Calculating the centroid

Denoting this area as C and, with $x = r\cos\theta$ and $dA = r\,drd\theta$ we have

$$\begin{aligned} r(j,t) &= \frac{\iint_R r\cos\theta\ (r\,drd\theta)}{C} \\ &= \left(\frac{1}{3}\right)\cdot\frac{n_{arcs}}{\pi}\cdot\sin\frac{2\pi}{n_{arcs}}\cdot\frac{rp(j+1,t)^2+rp(j+1,t).rp(j,t)+rp(j,t)^2}{rp(j+1,t)+rp(j,t)} \end{aligned}$$

with $r(j,t)$ denoting the radial component of the centroid.

Now that we have time dependent arrays to store the centroid of each element for all t, it is a simple matter to solve for kinetic energy using $ke = \frac{1}{2}\cdot m\cdot v^2$.

We will use the two second difference equations given below,

$$vz(i,t) = \frac{z(i,t+1)-z(i,t-1)}{2\cdot\Delta t}$$

and

$$vr(j,t) = \frac{r(j,t+1)-r(j,t-1)}{2\cdot\Delta t},$$

respectively to approximate the velocity in the z and r directions for the i^{th} and j^{th} element for time $1 < t < n_t$. At $t = 0$ both $vz(i,t)$ and $vr(j,t)$ are set to zero. The resultant speed, denoted $vres(i,j,t)$, is

$$vres(i,j,t) = \sqrt{vz(i,t)^2 + vr(j,t)^2}.$$

For $t = n_t$ we reduce our approximation to a first order backward difference equation.

Recall that each element has volume equal to *uvol* so the mass of each element becomes

$$umass = uvol \cdot density$$

and finally

$$ke(i,j,t) = \frac{1}{2} \cdot umass \cdot vres(i,j,t)^2.$$

To calculate the work going in we need to know the amount of movement that is generated as a result of the supplied force. Combining the equations defining *Work* and *Stress(pressure)* with Hooke's law we have

$$Work = (strain \cdot E) \cdot area \cdot distance.$$

From [13] we know there is a stress being applied from both the incident and transmitter bars, that is,

$$\sigma_1 = (\varepsilon_i + \varepsilon_r) \cdot E, \qquad F_1 = A_b \cdot \sigma_1,$$

and

$$\sigma_2 = (\varepsilon_t) \cdot E, \qquad F_2 = A_b \cdot \sigma_2,$$

where A_b is the crossectional area of the bar and σ_1 and F_1 are the stress and force respectively at the incident bar, and σ_2 and F_2 are the stress and force respectively at the transmitter bar. The force on the specimen is the average of these applied forces

$$F_s = \left(\frac{F_1 + F_2}{2}\right).$$

Since we have sectioned the specimen this applied force must be further reduced to obtain the force applied to each element

$$F_e = \frac{F_s}{n_{arcs} \cdot n_r}.$$

What is required is the work going in to each element. Again, from [13], we have

$$\sigma_s = \left(\frac{A_b}{A_s}\right) \cdot \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)(t)}{2}\right) \cdot E.$$

Using

$$\sigma_s = \frac{n_{arcs} \cdot n_r \cdot F_e}{A_s}$$

we can rewrite the above equation as

$$\frac{n_{arcs} \cdot n_r \cdot F_e}{A_s} = \left(\frac{A_b}{A_s}\right) \cdot \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)(t)}{2}\right) \cdot E,$$

and after a final substitution we obtain

$$W = \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)(t)}{2}\right) \cdot E \cdot \frac{1}{n_{arcs} \cdot n_r} \cdot \pi \cdot r_b^2 \cdot (\varepsilon_i - \varepsilon_r - \varepsilon_t) \cdot L_s.$$

where r_b denotes the radius of the incident bar in cm. For work to done on the specimen it is necessary for $\varepsilon_i > (\varepsilon_r + \varepsilon_t)$.

Deformation throughout the specimen will be uniform and therefore symmetric with respect to the specimen center. The work going into deformation must be evenly distributed along the z axis so

$$W_e = \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)(t)}{2}\right) \cdot E \cdot \frac{1}{n_{arcs} \cdot n_r} \cdot \pi \cdot r_b^2 \cdot \frac{(\varepsilon_i - \varepsilon_r - \varepsilon_t) \cdot L_s}{n_z}.$$

This is the equation we will use to calculate the work going into deforming each element.

2.2 Internal Energy

We wish to balance the equation given at the beginning of this chapter. We have calculated the amount of work going into the specimen and the amount of energy going into accelerating the specimen. What remains is the unrecoverable work or the work we attribute to the change in internal energy, ΔIE . From [23] we have

$$\Delta IE = \rho \cdot V \cdot \int_{T_i}^{T_f} c_v(T) \cdot dT$$

where $c_v(T)$ denotes the specific heat at constant volume as a function of temperature. From [20], hereafter referenced simply as TAPP, we obtain the specific heat of a solid at constant pressure in J/(mol·K) as

$$c_p(T) = -c_3 - 2(c_4 \cdot 10^{-3})T - 2(c_5 \cdot 10^6)T^{-2} - 6(c_6 \cdot 10^{-6})T^2 + \left(\frac{c_8}{4}\right)T^{-0.5}$$

Although there is a difference between $c_p(T)$ and $c_v(T)$ from [14] or [23] we know that, at least in the temperature regions in which we will be working,

$$c_v(T) \cong c_p(T).$$

The exact relationship is given by

$$c_p - c_v = 9\alpha^2 BVT$$

where α is the temperature coefficient of linear expansion, V is volume and B the bulk modulus. At room temperatures the difference is minimal.

The TAPP database supplies the constants required in the $c_p(T)$ equation for tantalum as

$$\begin{aligned}
c_3 &= -47.6677, \\
c_4 &= 5.45091, \\
c_5 &= -0.21647, \\
c_6 &= -0.59727, \\
c_8 &= -1669.643,
\end{aligned}$$

but these will need to be scaled to match our units. To reduce round-off errors we will use a modified cgs system of units: cm, g, μs and Mbars. The units for work (and internal energy) will be g$\cdot$cm^2/ μs^2 so we need the TAPP constants in $\mathrm{g} \cdot \mathrm{cm}^2/\mu\mathrm{s}^2 \cdot K$-mol. To convert from mks to our system requires

$$\frac{\mathrm{kg} \cdot \mathrm{m}^2}{\mathrm{s}^2 \cdot K \cdot \mathrm{moles}} \times \frac{10^3\ \mathrm{g}}{1\ \mathrm{kg}} \times \left(\frac{10^2\ \mathrm{cm}}{1\ \mathrm{m}}\right)^2 \times \left(\frac{1\mathrm{s}}{10^6\ \mu\mathrm{s}}\right)^2,$$

a scaling factor of 10^{-5}.

We also need a unit correction factor in the internal energy equation. Instead of merely $\rho \cdot V$ we need to add an additional constant in mol/g to complete the conversion. For tantalum the atomic weight is 180.9479 g/ mol. Therefore the needed constant is

$$\frac{1\ \mathrm{mol}}{180.9479\ \mathrm{g}} = 5.526453 \times 10^{-3}\ \mathrm{mol}/\mathrm{g},$$

so for us

$$\begin{aligned}\Delta IE &= c_0 \cdot \int_{T_i}^{T_f} \Big[-c_3 - 2(c_4 \cdot 10^{-3})T - 2(c_5 \cdot 10^6)T^{-2} \\ &\quad -6(c_6 \cdot 10^{-6})T^2 + \left(\frac{c_8}{4}\right) T^{-0.5}\Big]\, dT\end{aligned}$$

where $c_0 = umass \cdot (5.526453 \times 10^{-3}\ \mathrm{mol}/\mathrm{g})$. After performing the integration,

$$\begin{aligned}\Delta IE &= c_0 \Big[-c_3 T \,\Big|_{T_i}^{T_f} - c_4 \cdot 10^{-3} T^2 \,\Big|_{T_i}^{T_f} + 2(c_5 \cdot 10^6)\frac{1}{T} \,\Big|_{T_i}^{T_f} \\ &\quad -2(c_6 \cdot 10^{-6})T^3 \,\Big|_{T_i}^{T_f} + \left(\frac{c_8}{2}\right)\sqrt{T} \,\Big|_{T_i}^{T_f}\Big]\end{aligned}$$

Therefore at each time step t the temperature at $t-1$ can be used to solve for the next temperature value beginning with the initial temperature of $T_i = 293\,K$; see Figure 2.3

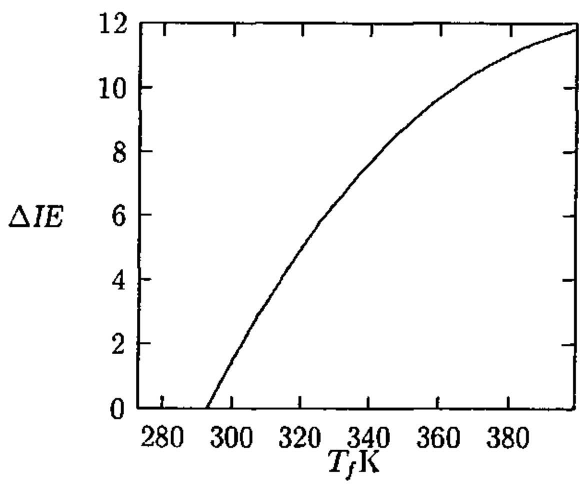

Figure 2.3: A plot of ΔIE vs $T_f K$

Chapter 3

Diffusion and the Heat Equation

Before impact the specimen and adjoining bars were in a state of thermal equilibrium—all elements of the system were at room temperature. Certainly after the test, and as we are trying to show, before deformation is complete, this will no longer be the case. The energy spent in deforming the specimen will, in part, manifest itself as heat and the resulting temperature differential will initiate heat conduction within the specimen and in both the incident and transmitter bars.

3.1 The Heat Equation

We begin with the heat equation, one of the classical partial differential equations of mathematical physics used to describe the conduction of heat in a solid body. In three dimensions we have

$$T_t = k\,\Delta T.$$

The temperature, T, is a function of x, y, z, and t, and k is the *thermal diffusivity* constant, one of the material properties of the solid. For our geometry we introduce a change of variables from (x, y, z, t) to cylindrical coördinates (r, θ, z, t) so that

$$\frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2}\right] .$$

Making use of radial symmetry $(\partial T/\partial \theta = 0)$ we can reduce our heat equation to

$$\frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right] . \tag{3.1}$$

The heat equation in cylindrical coördinates involves a singularity at $r = 0$. Since our domain includes $r \geq 0$ and we know we have a bounded solution at $r = 0$, we need to determine how to represent a solution of our heat equation to avoid the singularity. As $r \to 0$ we reduce our cylinder to a one dimensional object which will only experience heat diffusion in the z axis. Therefore as $r \to 0$, $\partial T/\partial r \to 0$ and we have the indeterminate form $0/0$. We can apply l'Hôpital's rule to the quotient leaving

$$\lim_{r \to 0} \frac{1}{r}\frac{\partial T}{\partial r} = \frac{\partial^2 T}{\partial r^2} .$$

Thus, for small r, we can express the heat equation as

$$\frac{\partial T}{\partial t} = k\left[2\frac{\partial^2 T}{\partial r^2} + \frac{\partial^2 T}{\partial z^2}\right] .$$

3.1.1 Numerical Solution of the Heat Equation

Although there are many references that present schemes for the solution of the two-dimensional heat equation with time dependence, few discuss the condition of both spatially and temporally dependent boundary conditions. One reference to present such a solution was Carslaw [6], using Duhamel's theorem, but the solution is not representable in terms of well-known special functions. Since the data being supplied is discrete, and since a solution in closed form is not available, a numerical solution is required.

Finite Difference Equations

To express the partial differential equations we are using in our model as finite differences we turn to Taylor's theorem. Recall that when T and its derivatives are singled-valued, finite, continuous functions of x, then

$$T(x+\Delta x) = T(x) + \Delta x\, T'(x) + \frac{(\Delta x)^2}{2!} T''(x) + \frac{(\Delta x)^3}{3!} T'''(x) + \cdots$$

and

$$T(x-\Delta x) = T(x) - \Delta x\, T'(x) + \frac{(\Delta x)^2}{2!} T''(x) - \frac{(\Delta x)^3}{3!} T'''(x) + \cdots .$$

Solving the first expansion for $T'(x)$ yields

$$T'(x) = \frac{T(x+\Delta x) - T(x)}{\Delta x} + O(\Delta x),$$

a forward difference expression involving $\frac{(\Delta x)}{2!} T''(x) + \frac{(\Delta x)^2}{3!} T'''(x) + \cdots$ which we group together as $O(\Delta x)$. Addition of these Taylor expansions yields

$$T(x+\Delta x) + T(x-\Delta x) = 2T(x) + (\Delta x)^2 T''(x) + O\,(\Delta x)^4 ,$$

where $O\,(\Delta x)^4$ denotes other terms containing fourth and higher powers of Δx. Rearranging this equation to solve for the second derivative gives

$$T''(x) = \frac{1}{(\Delta x)^2}\left[T(x+\Delta x) - 2T(x) + T(x-\Delta x)\right] + O\,(\Delta x)^2 .$$

Since we are dealing with approximations, if we assume the terms involving Δx , or $(\Delta x)^2$ in the case of the second derivative, are negligible in comparison to the other terms we have the well-known approximations

$$\begin{aligned} T'(x) &\doteq \frac{T(x+\Delta x)-T(x)}{\Delta x}, \\ T''(x) &\doteq \frac{T(x+\Delta x)-2T(x)+T(x-\Delta x)}{(\Delta x)^2}. \end{aligned}$$

Applying these to functions of several variables we have, for $T(r,z,t)$, $\partial T/\partial t$ and $\partial T/\partial r$,

$$\begin{aligned} \frac{\partial T}{\partial t} &\doteq \frac{T(r_0,z_0,t+\Delta t)-T(r_0,z_0,t)}{\Delta t}, \\ \frac{\partial T}{\partial r} &\doteq \frac{T(r+\Delta r,z_0,t_0)-T(r,z_0,t_0)}{\Delta r}, \end{aligned} \tag{3.2}$$

and similarly for $\partial T^2/\partial r^2$ and $\partial T^2/\partial z^2$:

$$\begin{aligned} \frac{\partial^2 T}{\partial r^2} &\doteq \frac{T(r+\Delta r,z_0,t_0)-2T(r,z_0,t_0)+T(r-\Delta r,z_0,t_0)}{(\Delta r)^2}, \\ \frac{\partial^2 T}{\partial z^2} &\doteq \frac{T(r_0,z+\Delta z_0,t_0)-2T(r_0,z,t_0)+T(r_0,z-\Delta z_0,t_0)}{(\Delta z)^2}. \end{aligned}$$

Note that the approximation scheme for $\partial T/\partial r$ has an error $O(\Delta r)$ whereas $\partial T^2/\partial r^2$ has an error $O(\Delta r)^2$. Since our partial differential equation involves the addition of these two derivatives, it would be desirable to keep the error terms consistent so that we could simply lump the error terms together as $O(\Delta r^2)$.

If we subtract the Taylor expansion for $T(x-\Delta x)$ from $T(x+\Delta x)$ we obtain a different approximation scheme for $\partial T/\partial r$, one whose error is now $O(\Delta r)^2$, that is,

$$\begin{aligned} T(x+\Delta x)-T(x-\Delta x) &= \left[T+T'\cdot\Delta x+\frac{T''}{2!}\cdot(\Delta x)^2+\frac{T'''}{3!}\cdot(\Delta x)^3+\cdots\right] \\ &\quad -\left[T-T'\cdot\Delta x+\frac{T''}{2!}\cdot(\Delta x)^2-\frac{T'''}{3!}\cdot(\Delta x)^3+\cdots\right] \\ &= 2T'\cdot\Delta x+O(\Delta x)^3 \end{aligned}$$

where T and its derivative appearing on the right are all evaluated at x. Upon solving for T', we find

$$T'(x) = \frac{T(x+\Delta x) - T(x-\Delta x)}{2\Delta x} + O\left(\Delta x\right)^2 .$$

To allow for combining of error terms on the right hand side of our heat equation, equation (3.1), we will rewrite (3.2) as,

$$\frac{\partial T}{\partial r} \doteq \frac{T(r+\Delta r, z_0, t_0) - T(r - \Delta r_0, z_0, t_0)}{2\Delta r}.$$

To complete our derivation of the finite difference equations, we introduce the following notation:

$$T(j\Delta r, m\Delta z, n\Delta t) \equiv T(r_j, z_m, t_n) \equiv T_{j,m}^{(n)},$$

for $j = 1, \ldots, n_r$, $m = 1, \ldots, n_z$, and $n = 1, \ldots, t_n$. We can therefore approximate our partial differential heat equation

$$\frac{\partial T}{\partial t} = k\left[\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right]$$

in the following form:

$$\begin{aligned}\frac{T_{j,m}^{(n+1)} - T_{j,m}^{(n)}}{\Delta t} &= k\left[\frac{T_{j+1,m}^{(n)} - 2T_{j,m}^{(n)} + T_{j-1,m}^{(n)}}{(\Delta r)^2}\right. \\ &\quad \left. + \frac{1}{j\Delta r}\frac{T_{j+1,m}^{(n)} - T_{j-1,m}^{(n)}}{2\Delta r} + \frac{T_{j,m+1}^{(n)} - 2T_{j,m}^{(n)} + T_{j,m-1}^{(n)}}{(\Delta z)^2}\right].\end{aligned}$$

For the case where r is near 0 we had the partial differential equation

$$\frac{\partial T}{\partial t} = 2k\frac{\partial^2 T}{\partial r^2} + k\frac{\partial^2 T}{\partial z^2}$$

which, for $j = 0$, we can record as

$$\frac{T_{j,m}^{(n+1)} - T_{j,m}^{(n)}}{\Delta t} = 2k\left[\frac{T_{j+1,m}^{(n)} - 2T_{j,m}^{(n)} + T_{j-1,m}^{(n)}}{(\Delta r)^2}\right] + k\left[\frac{T_{j,m+1}^{(n)} - 2T_{j,m}^{(n)} + T_{j,m-1}^{(n)}}{(\Delta z)^2}\right] .$$

We can simplify this equation by making use of the restriction we imposed earlier, namely $\partial T/\partial r \to 0$ as $r \to 0$ so that, since

$$\frac{\partial T}{\partial r} \doteq \frac{T^{(n)}_{j+1,m} - T^{(n)}_{j-1,m}}{2\Delta r},$$

for $j = 0$ we have

$$\frac{T^{(n)}_{j+1,m} - T^{(n)}_{j-1,m}}{2\Delta r} = 0, \quad \text{or } T^{(n)}_{j+1,m} = T^{(n)}_{j-1,m} \,.$$

With this, our finite difference equation becomes

$$\frac{T^{(n+1)}_{0,m} - T^{(n)}_{0,m}}{\Delta t} = 4k\left[\frac{T^{(n)}_{1,m} + T^{(n)}_{0,m}}{(\Delta r)^2}\right] + k\left[\frac{T^{(n)}_{0,m+1} - 2T^{(n)}_{0,m} + T^{(n)}_{0,m-1}}{(\Delta z)^2}\right] .$$

Since we have Dirichlet boundary conditions, that is the boundary temperatures are known, we will substitute the required ambient temperature value when dealing with calculations requiring $T^{(n)}_{-1,m}$ or $T^{(n)}_{j,-1}$.

In order to use these difference equations to "march ahead in time" as we numerically solve the heat equation, we need to separate out terms involving n from those involving $n+1$. That is, the points $T^{(n)}_{j,m}$, $T^{(n)}_{j+1,m}$, $T^{(n)}_{j-1,m}$, $T^{(n)}_{j,m+1}$, and $T^{(n)}_{j,m-1}$ are used to calculate $T^{(n+1)}_{j,m}$ (see Figure 3.1).

For $j \neq 0$, solving our finite difference equation for $u^{(n+1)}_{j,m}$ yields

$$\begin{aligned} T^{(n+1)}_{j,m} &= \frac{s_1}{2j}\left[(2j+1)T^{(n)}_{j+1,m} - 4jT^{(n)}_{j,m} + (2j-1)T^{(n)}_{j-1,m}\right] \\ &\quad + s_2(T^{(n)}_{j,m+1} - 2T^{(n)}_{j,m} + T^{(n)}_{j,m-1}) + T^{(n)}_{j,m}, \end{aligned} \tag{3.3}$$

and for $j = 0$,

$$T^{(n+1)}_{0,m} = 4s_1(T^{(n)}_{1,m} - T^{(n)}_{0,m}) + s_2(T^{(n)}_{0,m+1} - 2T^{(n)}_{0,m} + T^{(n)}_{0,m-1}) + T^{(n)}_{0,m} \tag{3.4}$$

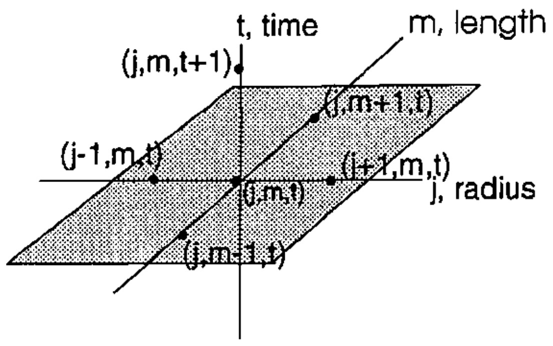

Figure 3.1: Scheme for calculating the n+1 term

where

$$s_1 = \frac{k\Delta t}{(\Delta r)^2} \text{ and } s_2 = \frac{k\Delta t}{(\Delta z)^2} \,.$$

These are the finite difference equations we use in the computer model to simulate heat diffusion.

3.2 Modelling Diffusion at the Interface

This is the heart of our problem. What happens to the heat being generated in the specimen as a result of deformation? If the rate of deformation is fast enough, in essence instantaneous, specimen temperature will rise since there will be no time for diffusion (adiabatic). The slower the deformation rate the more time allowed for diffusion and at the other extreme, if the deformation is slow enough, all the heat from the specimen will diffuse and the specimen temperature will be constant (isothermal).

Heat being generated within the specimen will diffuse within the specimen and into the incident bar. Thus we need to model two diffusions with particular attention to the incident bar/specimen interface.

The face of the incident bar was divided up in the same way as the specimen with one difference; we do not require each sector to be divided into pieces of equal volume. Here we have a uniform mesh such that

$$\frac{r_b}{n_{rb}} = \Delta r_b$$

where n_{rb} denotes the number of divisions along the r axis. As a result there is a mismatch between the non-uniform mesh on the specimen and the point of contact with the incident bar. A cubic spline is used to provide a smooth interface between these two surfaces. (For some background on the use of cubic splines, see [21].)

The problem is two-fold: the mismatch between specimen and incident bar, and the changing radius of the specimen as deformation proceeds.

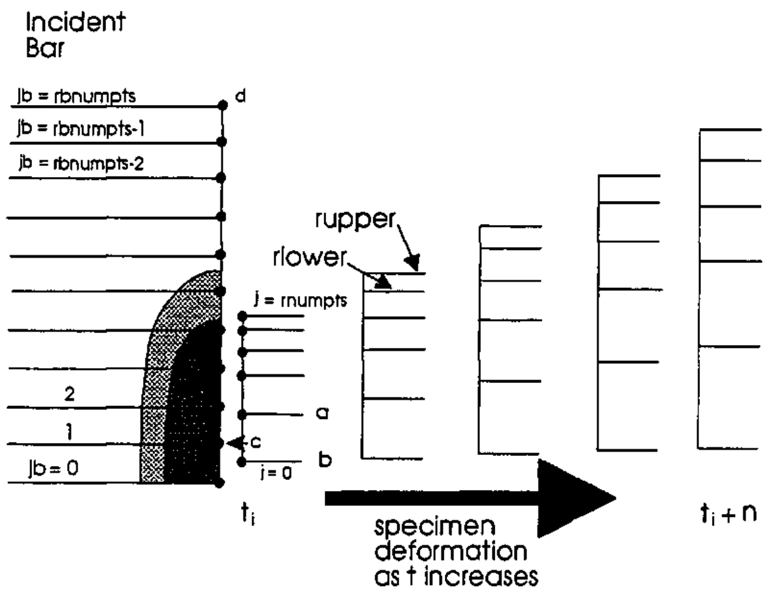

Figure 3.2: Specimen in contact with the incident bar

From the section on internal energy we know we can calculate the temperature of the specimen as a function of both t and r. For temperatures in between the centroid positions on the specimen we make use of a spline profile. Each of these temperatures is represented by a cubic polynomial of the form

$$f_i(r) = a_i + b_i(r - r_i) + c_i(r - r_i)^2 + d_i(r - r_i)^3$$

for $r_i \leq r \leq r_{i+1}$ and $i = 0, 1, \cdots, n_t - 1$. The four parameters a_i, b_i, c_i, and d_i are solved in a subroutine. These polynomials supply data for the temperature calculations at the incident bar/specimen interface.

At each stage of the deformation the specimen radius is increasing. In terms of a match up between the element centroids on the specimen and the lattice laid out on the incident bar several scenarios are possible: there could be an exact match between the specimen and the incident bar, or there could be one , none or several loci on the incident bar that fall between two adjoining centroids on the specimen; see Figure 3.2.

Beginning at the incident bar radius a check is made to see if there is contact with the specimen. Since the deforming specimen will eventually flatten out to cover the entire bar end, we begin at the outer surface of the incident bar, point d; see Figure 3.2. We need to determine if contact is being made with the specimen and where. To do this we initially assign *rupper* to the centroid position corresponding to $j = n_r$ and *rlower* to the next centroid location $j = n_r - 1$. Beginning at $j_b = n_{rb}$, the outside edge of the incident bar, the following check is made

$$rlower < j_b \cdot \Delta r_b \leq rupper$$

where j_b is a counter to determine the position on the incident bar and j is the counter to determine the position on the specimen. If the test is true we have contact and the correct splined temperature is assigned to that point on the specimen. This temperature will then be used in the calculation to determine the diffused temperature on the face of incident bar. If the test is false we do not have contact and the bar temperature remains unchanged. Once a temperature is assigned the counter is reduced by one, $j_b = j_b - 1$ and the check is made again. Eventually, as j_b is decremented, $j_b \cdot \Delta r_b$ will be less than *rlower*. When this occurs we need to slide down the specimen and establish a new range of contact, that is

$$\begin{aligned} rupper &\rightarrow rlower \\ rlower &\rightarrow r(j-1,0,t) \ . \end{aligned}$$

Then the initial check, $rlower < jb \cdot \Delta r_b \leq rupper$, is made again and the process continues until $rlower = r(0,0,t)$. Even though we have reached the last centroid position on the specimen there may be incident bar loci below this value. Any specimen temperatures required below this point will be assigned the same value as the last point of contact. Once this loop has been negotiated we know where and how much heat is ready to be applied to the face of the incident bar. The final step is to calculate diffusion in both the specimen and the incident bar.

We have two general equations (3.1) and (3.2), that must be applied to the nine conditions shown in Figure 3.3.

In both metals we have basically three types of points: internal, boundary and contact. The software uses three temperature arrays to model the diffusion. Two are used to record temperatures in the incident bar, $Tb(i,j,t)$, and specimen $Ts(i,j,t)$. The third, $Tsi(i,j,t)$ is used to record the extra splined temperatures required for diffusion at the interface.

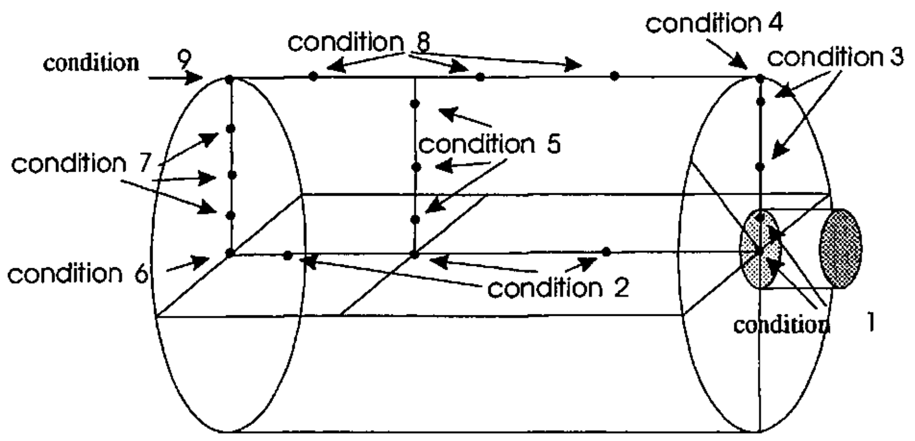

Figure 3.3: Diffusion in the incident bar

1. As explained above, beginning at the incident bar radial boundary, a check is made to see if a locus on the incident bar is in contact with the specimen. If there is contact a splined specimen temperature is calculated, $Tsi(i, j, t)$, and used as the $(j, m-1, t)$ point in the incident bar diffusion calculation. Similarly, on the specimen side, incident bar temperatures are used in specimen diffusion calculations involving $(j, m-1, t)$. Once temperatures have been assigned to the interface a flag is set so this condition is not checked again until t is incremented. That is, this condition has a built in loop that calculates the diffusion temperature for the entire area as soon as it is determined that contact has been made.

2. Along the central axis $j = 0$ so we need (3.2) to calculate the diffusion temperatures.

3. In the annulus only four points are available to calculate the diffusion temperature. Equation (3.1) calls for $(j, m-1, t)$ – see Figure 3.1 – but at the annulus

this point is beyond the boundary. In general two types of boundary problems are discussed in such texts as [15] or [5]. Either the boundary temperature is known at all points, a *Dirichlet problem*, or the normal gradient of temperature is specified, a *Neuman problem*. At $t = 0$ we know the temperature of the boundary but as time progresses this ambient air temperature is not being held constant. Still, since the test is on the order of microseconds, there will be little time for the bar to lose heat to the outside air. For this reason we have assumed Dirichlet boundary conditions throughout the remainder of this discussion. Note also that once condition 1 has been met, there is no further need to check this condition. In the software implementation of this model the status of `con1flag` is used to check this possibility. This condition only exists on the incident bar.

4. These are the points on the radial boundary at the incident bar/specimen interface. For the incident bar the calculation uses ambient air temperatures for both $(j+1, m, t)$, and $(j, m-1, t)$. The specimen calculation only requires ambient air temperature for $(j+1, m, t)$.

5. These are the internal points so equation (3.1) is used in both incident bar and specimen diffusion calculations.

6. This is the single point at the middle of the bar opposite the specimen. The ambient air temperatue is used for $(j, m+1, t)$ and, due to symmetry, the temperature at $(j+1, m, t)$ is used to replace $(j-1, m, t)$.

7. These are the points on the end of the bar opposite the specimen excluding the center and outside boundary point. Ambient air temperatures are used for $(j, m+1, t)$.

8. These are all the radial boundary points. Ambient air temperatures are used for $(j+1, m, t)$.

9. This is the single point on the end opposite the specimen. Ambient air temperatures are used for both $(j+1, m, t)$ and $(j, m+1, t)$.

With these exceptional conditions in place we have constructed a complete numerical model for diffusion in the incident bar, in the specimen and at the incident bar/specimen interface.

Chapter 4

The Model

4.1 Introduction

The energy considerations and element decomposition of Chapter 2, together with the numerical diffusion scheme presented in the preceding section form the basis of a computer program, **energy**, which can be used to examine the adiabaticity of the SHPB text.

The code is written in Fortran 77. Since accuracy is important, double precision was used at the expense of processing speed. Accuracy will also increase as the mesh size for both specimen and incident bar are reduced. All parameters and initialization data can be read in from a data file.

The resulting temperature rise in K, in each of the specimen and incident bars, is calculated at each time step and the results are stored in individual files.

4.2 Computations

Initialization data is read in and stored in arrays, parameters and names. Required initial data includes the:

1) strain history data, $\varepsilon_i, \varepsilon_r$, and ε_t in micro strain, as recorded from the strain

gauges on the SHPB apparatus;

2) uniform time difference between reading in μs;

3) number of time steps (total test time divided by Δt);

4) physical dimensions, radius and length, of the specimen and incident bar in cm;

5) modulus of elasticity of the incident and transmitter bars in Mbars;

6) density of the specimen in g/cm^3;

7) initial temperature of the specimen in K;

8) mesh size of both the specimen and incident bar. Three values are required for the specimen: 1) the number of divisions in the circle, n_{arcs} (same for both specimen and incident bar), 2) the number of divisions along the z-axis, n_z and n_{zb} respectively and 3) the number of divisions along the radius, n_r and n_{rb} respectively;

9) TAPP constants for the specific heat equation based on the specimen material. These constants must be scaled to match the units the code uses.

After the initialization data is read in calculations are done as follows:

1) Based on the strain history data, the position of each element in the specimen is solved for all t;

2) Using the strain history data the amount of work supplied to the specimen is calculated. Using the specimen position data, velocity and kinetic energy calculations are done;

3) The difference between work supplied and kinetic energy yields internal energy;

4) Specimen temperature and the coefficients for the cubic spline polynomial are calculated;

5) The splined temperature is used in diffusion calculations at the incident bar/specimen

interface;

6) The diffusion calculations in both the incident bar and specimen;

The resulting temperature arrays are stored in separate files, named TEM_BAR.DAT for the incident bar and TEM_SPEC.DAT for the specimen. Output is a ordered triple, (z, r, T), showing temperature relative to position. A short listing of each data file is shown in appendices A and B respectively.

By removing the comment designation on several lines of Fortran code a detailed output can be produced. A record of the deformation using the coordinates of the specimen centroid, z and r respectively, are stored in ZCENT.DAT and RCENT.DAT. Speed, work, kinetic energy and internal energy are stored in SPEED.DAT, WORKI.DAT, KINEN.DAT and INTEN.DAT respectively.

4.3 Test Runs

The first test run was generated using amaco iron as the specimen material with:

1. specimen mesh, $n_r = 5$, $n_{arcs} = 100$ and $n_z = 5$,

2. incident bar, $n_{rb} = 10$, $n_{arcs} = 100$ and $n_{zb} = 10$,

3. $n_t = 399$, corresponding to an actual test time of approximately $100\,\mu\,\mathrm{sec}$.

4.4 Stability

Recall our finite difference equations (3.1) and (3.2) involve s_1 and s_2 (ss_1 and ss_2 for specimen diffusion) such that

$$T(t+1) = s1(\text{ terms involving } T(t)) + s2(\text{terms involving } T(t)).$$

Both s_1 and s_2 are calculated using the mesh sizes Δt, Δr and Δz. If the speed at which we move along in time does not correspond with the "speed" we move along spatially there will instability. In our case since

$$s_1 = \frac{k\Delta t}{(\Delta r)^2} \text{ and } s_2 = \frac{k\Delta t}{(\Delta z)^2} ,$$

and Δt is in microseconds, Δr and Δz must be in the order of 10^{-3} so that s_1, $s_2 \leq 1$. Larger values will cause temperature values to soar well above 2000 K. Such values are meaningless since they are above the melting point of the specimen material.

Chapter 5

Conclusions

This problem involves heat generated as a result of deformation and, time permitting, diffusion via conduction. Initially the entire system is at room temperature, 293 K. We know the specimen is under compression and we know time for diffusion is short. Given these bench marks the model does not produce realistic temperatures. Output shows specimen temperature falling below 293 K in some elements even though specimen cooling is not possible. Furthermore, even if we only focus on those portions of the specimen that gain heat, by the end of the test these temperatures have dropped significantly. Such cooling is not possible in 60 - 80μs. In our efforts to explain these results we examined both the strain data and the calculations.

For the specimen to be undergoing compression we required

$$\varepsilon_i - (\varepsilon_r + \varepsilon_t) > 0$$

and

$$\varepsilon_i > 0.$$

Fewer than one quarter of the data satisfied these conditions. Furthermore the qualifying data were irregularly spaced. Typically one would expect unreliable data at the beginning and/or end of the test but in our case we would find several contiguous points, miss a few, find a few more, then miss some and so on. Occasionally there would be gaps of over 100 data representing a $25\mu s$ time interval–one quarter of the test. Such irregularities lead to timing problems in the calculations.

During deformation the specimen gains heat through an increase in internal energy. However, when we checked these calculations we found $\Delta IE < 0$–clearly an impossibility since this will lead to specimen cooling.

Stepping aside for the moment we can illustrate the effect $\varepsilon_i - (\varepsilon_r + \varepsilon_t) > 0$ has on the specimen temperatures by presenting a listing of comparsion temperatures t between 100 and 103 for the following schemes:

1. the status quo (which is to say, no effort to correct for $\Delta IE < 0$)
2. truncation (that is when $\Delta IE < 0$ just throw away the negative ΔIE and replace it with $\Delta IE = 0$)
3. taking absolute value (that is when $\Delta IE < 0$ set $\Delta IE = \mid \Delta IE \mid$)

The arrays are in the form

T(1,1) T(1,2) T(1,3) ... T(1, n_z)

T(2,1) T(2,2) T(2,3) ... T(2, n_z)

...

T(n_r,1) T(n_r,2) T(n_r,3) ... T(n_r, n_z).

```
Temperatures for scheme one.
specimen temperature array at time  100
    0.289939363256E+03    0.290011167677E+03    0.290064993144E+03    0.290100876919E+03    0.290118818845E+03
    0.289940158131E+03    0.290011931619E+03    0.290065757089E+03    0.290101640866E+03    0.290119582794E+03
    0.289940922862E+03    0.290012696387E+03    0.290066521861E+03    0.290102405640E+03    0.290120347568E+03
    0.289941690153E+03    0.290013463711E+03    0.290067289189E+03    0.290103172970E+03    0.290121114900E+03
    0.289942484663E+03    0.290014258219E+03    0.290068083700E+03    0.290103967484E+03    0.290121909415E+03
 specimen temperature array at time  101
    0.291147538727E+03    0.291167826985E+03    0.291183018543E+03    0.291193146181E+03    0.291198212638E+03
    0.291147805027E+03    0.291168042461E+03    0.291183234020E+03    0.291193361658E+03    0.291198425219E+03
    0.291148020715E+03    0.291168258236E+03    0.291183449794E+03    0.291193577431E+03    0.291198640992E+03
    0.291148237130E+03    0.291168474728E+03    0.291183666284E+03    0.291193793920E+03    0.291198857480E+03
    0.291148461244E+03    0.291168698847E+03    0.291183890402E+03    0.291194018039E+03    0.291199081599E+03
 specimen temperature array at time  102
    0.292313706598E+03    0.292314439963E+03    0.292314964402E+03    0.292315314018E+03    0.292315491523E+03
    0.292313759848E+03    0.292314447386E+03    0.292314971824E+03    0.292315321441E+03    0.292315496182E+03
    0.292313767266E+03    0.292314454833E+03    0.292314979272E+03    0.292315328888E+03    0.292315503629E+03
    0.292313774708E+03    0.292314462304E+03    0.292314986742E+03    0.292315336358E+03    0.292315511099E+03
    0.292313782423E+03    0.292314470029E+03    0.292314994467E+03    0.292315344083E+03    0.292315518824E+03
 specimen temperature array at time  103
    0.293199074488E+03    0.293208340782E+03    0.293215309793E+03    0.293219955801E+03    0.293222280831E+03
    0.293199121526E+03    0.293208439653E+03    0.293215408665E+03    0.293220054672E+03    0.293222377675E+03
    0.293199220517E+03    0.293208538632E+03    0.293215507644E+03    0.293220153651E+03    0.293222476654E+03
    0.293199319839E+03    0.293208637942E+03    0.293215606953E+03    0.293220252961E+03    0.293222575963E+03
    0.293199422680E+03    0.293208740770E+03    0.293215709781E+03    0.293220355789E+03    0.293222678791E+03


Temperatures for scheme two.
specimen temperature array at time  100
    0.293000000000E+03    0.292999999544E+03    0.292999999423E+03    0.292999999463E+03    0.292999999964E+03
    0.293000000000E+03    0.292999999542E+03    0.292999999422E+03    0.292999999462E+03    0.292999999880E+03
    0.293000000000E+03    0.292999999541E+03    0.292999999422E+03    0.292999999458E+03    0.292999999811E+03
    0.293000000000E+03    0.292999999541E+03    0.292999999421E+03    0.292999999453E+03    0.292999999766E+03
    0.293000000000E+03    0.292999999540E+03    0.292999999421E+03    0.292999999451E+03    0.292999999751E+03
specimen temperature array at time  101
    0.293000000000E+03    0.292999999549E+03    0.292999999425E+03    0.292999999468E+03    0.292999999964E+03
    0.293000000000E+03    0.292999999548E+03    0.292999999425E+03    0.292999999467E+03    0.292999999884E+03
    0.293000000000E+03    0.292999999547E+03    0.292999999424E+03    0.292999999463E+03    0.292999999820E+03
    0.293000000000E+03    0.292999999546E+03    0.292999999424E+03    0.292999999458E+03    0.292999999778E+03
    0.293000000000E+03    0.292999999546E+03    0.292999999423E+03    0.292999999457E+03    0.292999999764E+03
specimen temperature array at time  102
    0.293000000000E+03    0.292999999554E+03    0.292999999428E+03    0.292999999474E+03    0.292999999964E+03
    0.293000000000E+03    0.292999999553E+03    0.292999999427E+03    0.292999999472E+03    0.292999999889E+03
    0.293000000000E+03    0.292999999552E+03    0.292999999427E+03    0.292999999468E+03    0.292999999828E+03
    0.293000000000E+03    0.292999999551E+03    0.292999999426E+03    0.292999999464E+03    0.292999999789E+03
    0.293000000000E+03    0.292999999551E+03    0.292999999426E+03    0.292999999462E+03    0.292999999776E+03
specimen temperature array at time  103
    0.293199074488E+03    0.293208365629E+03    0.293215334921E+03    0.293219981115E+03    0.293222304651E+03
    0.293199173366E+03    0.293208464505E+03    0.293215433797E+03    0.293220079991E+03    0.293222403088E+03
    0.293199272350E+03    0.293208563488E+03    0.293215532780E+03    0.293220178974E+03    0.293222502071E+03
    0.293199371665E+03    0.293208662802E+03    0.293215632094E+03    0.293220278288E+03    0.293222601385E+03
    0.293199474499E+03    0.293208765634E+03    0.293215734926E+03    0.293220381120E+03    0.293222704217E+03


Temperatures for scheme three.
specimen temperature array at time  100
    0.296061193791E+03    0.295989334640E+03    0.295935493681E+03    0.295899599726E+03    0.295881650147E+03
    0.296060429557E+03    0.295988570356E+03    0.295934729400E+03    0.295898835447E+03    0.295880888048E+03
    0.296059664496E+03    0.295987805361E+03    0.295933964406E+03    0.295898070454E+03    0.295880123055E+03
    0.296058896877E+03    0.295987037800E+03    0.295933196846E+03    0.295897302894E+03    0.295879355496E+03
    0.296058102065E+03    0.295986242973E+03    0.295932402022E+03    0.295896508072E+03    0.295878560674E+03
specimen temperature array at time  101
    0.294852588498E+03    0.294832300680E+03    0.294817111125E+03    0.294806984798E+03    0.294801919740E+03
    0.294852372955E+03    0.294832085026E+03    0.294816895572E+03    0.294806769246E+03    0.294801705936E+03
    0.294852157178E+03    0.294831869277E+03    0.294816679822E+03    0.294806553496E+03    0.294801490186E+03
    0.294851940681E+03    0.294831652805E+03    0.294816463349E+03    0.294806337022E+03    0.294801273712E+03
    0.294851716514E+03    0.294831428639E+03    0.294816239183E+03    0.294806112857E+03    0.294801049547E+03
specimen temperature array at time  102
    0.293686240257E+03    0.293685506345E+03    0.293684983985E+03    0.293684635740E+03    0.293684459548E+03
    0.293686232834E+03    0.293685498922E+03    0.293684976562E+03    0.293684628317E+03    0.293684454147E+03
    0.293686225403E+03    0.293685491503E+03    0.293684969143E+03    0.293684620898E+03    0.293684446727E+03
    0.293686217947E+03    0.293685484059E+03    0.293684961699E+03    0.293684613453E+03    0.293684439283E+03
    0.293686210227E+03    0.293685476343E+03    0.293684953982E+03    0.293684605737E+03    0.293684431566E+03
specimen temperature array at time  103
    0.293199074488E+03    0.293208360126E+03    0.293215329726E+03    0.293219976125E+03    0.293222298367E+03
    0.293199173366E+03    0.293208459006E+03    0.293215428606E+03    0.293220075005E+03    0.293222398204E+03
    0.293199272350E+03    0.293208557994E+03    0.293215527593E+03    0.293220173993E+03    0.293222497192E+03
    0.293199371665E+03    0.293208657312E+03    0.293215626911E+03    0.293220273311E+03    0.293222596510E+03
    0.293199474499E+03    0.293208760148E+03    0.293215729748E+03    0.293220376147E+03    0.293222699346E+03
```

Returning to our discussion of ΔIE, if we go back one step further, the change in internal energy is the difference between work going in to deform the specimen, and kinetic energy. Since kinetic energy is a straightforward calculation we turned to how work going into deformation could be calculated to keep $\Delta IE > 0$. Three approaches were tried but none were successful.

One approach is uniform distribution by volume. In this case

$$W_e = \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)\,(t)}{2}\right) \cdot E \cdot \frac{1}{n_{arcs} \cdot n_r} \cdot \pi \cdot r_b^2 \cdot \frac{(\varepsilon_i - \varepsilon_r - \varepsilon_t) \cdot L_s}{n_z}$$

so that each equal volume element is acted upon by the an equal quantity of work. The second approach called for a change in the constitutive equations given by [13]. In our first calculation work was independent of the changing cross sectional area of the specimen. Using

$$W_e = \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)\,(t)}{2}\right) \cdot E \cdot area\_s \cdot \frac{(\varepsilon_i - \varepsilon_r - \varepsilon_t) \cdot L_s}{n_z}$$

where

$$area\_s = \frac{\pi \cdot [rp(j+1,t)^2 - rp(j,t)^2]}{\pi \cdot r_s^2 \cdot n_{arcs}}$$

we distributed the work proportional to the element's annular area. Our third and last approach followed the reasoning that those elements of the specimen undergoing the greatest movement were having the greatest amount of work being done on them.

We used

$$workin\_e(i,j,t) = workin(t) \cdot \frac{ke(i,j,t)}{tke}$$

where

$$workin(t) = \left(\frac{(\varepsilon_i + \varepsilon_r + \varepsilon_t)(t)}{2}\right) \cdot E \cdot \frac{1}{n_{arcs}} \cdot \pi \cdot r_b^2 \cdot (\varepsilon_i - \varepsilon_r - \varepsilon_t) \cdot L_s$$

to distribute work proportional to the kinetic energy. The $workin(t)$ term is the total work done on one wedge, $workin\_e(i,j,t)$ denotes the work being done on one element, and $tke = \sum_{i=1}^{n_z} \sum_{j=1}^{n_r} ke(i,j,t)$, the total kinetic energy of one wedge. As previously mentioned, none of these schemes kept $\Delta IE > 0$ for all t. It would seem better conditioned data is needed to validate the model.

Bibliography

[1] Beckenhauer, D., N. Qiang, P. Niessen and R.J. Pick. *Numerical and Experimental Simulation of Adiabatic Shear Localizations in Tantalum and Armco Iron.* Technical Report, University of Waterloo, 1993

[2] Bertholf, L. D. and C.H. Karnes. *Two-dimensional Analysis of the Split Hopkinson Pressure Bar System.* Journal of the Mechanics of Solids, Vol.23, Pergamon Press, UK, 1975, pp. 1-19

[3] Bickford and Mikey. *Mechanics of Solids Concepts and Applications*, Irwin Publishing, New York 1994, pp. 100-156

[4] Campbell, J. D. and W.G.Ferguson. *The Temperature and Strain-rate Dependence of the Shear Strength of Mild Steel.* Philosophical Magazine, Vol. 21, 1970, pp 63-82

[5] Carrier, G. F. and C.E.Pearson. *Partial Differential Equations: Theory and Technique*, Second Edition, Academic Press, 1988

[6] Carslaw, H. S. *Conduction of Heat in Solids 2nd* edition. Oxford University Press, 1959

[7] Crank, J. *The Mathematics of Diffusion.* Oxford University Press, 1975

[8] Curran, D. R., J.A. Zukas, T. Nicholas, H.F. Swift, L.B.Greszczuk. *Impact Dynamics.* Krieger Publishing Company, Malabar, Florida, 1992

[9] Dally, J. W. and F.R. William. *Experimental Stress Analysis. Second Edition.* McGraw-Hill Book Company, 1978

[10] Follansbee, P. S. *High Strain Rate Compression Testing, Metals Handbook.* Ninth Edition, Vol. 8, AMS, Metals Park, Ohio, 1985, pp. 190-203

[11] Follansbee, P. S. and C. Frantz. *Wave Propagation in Split Hopkinson Pressure Bar.* ASME Journal of Engineering Materials and Technology, Vol. 105. 1983, pp 61-66

[12] Fung, Y. C. *A First Course In Continuum Mechanics*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1969

[13] Gallagher, P. J. *Split Hopkinson Pressure Bar Apparatus-Overview and Simple Analysis.* Defence Research Establishment, Valcartier, Québec, Memorandum 3048/90, September 1990

[14] Kittel, C. *Introduction to Solid State Physics.* Fifth Edition, Wiley, 1976

[15] Lapidus, L. and G.F. Pinder. *Numerical Solution of Partial Differential Equations in Science and Engineering.* John Wiley and Sons, New York, 1982

[16] Neely, J. E. *Practical Metallurgy and Materials of Industry.* Third Edition, Prentice Hall, New Jersey, 1989

[17] Oden, J. T. *Finite Elements of Nonlinear Continua*, McGraw-Hill, 1972

[18] Popov, E. P. *Introduction to Mechanics of Solids*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1968

[19] Smith, G. D. *Numerical Solution of Partial Differential Equations.* Oxford University Press, London, 1965

[20] (Thermochemical and Physical Properties). E S Microware, Inc., Hamilton Ohio, 1995

[21] Usmani, R. A. *Numerical Analysis for Beginners*, D and R Texts Publications, 1992

[22] Widder, D. V. *The Heat Equation.* Academic Press, London, 1975

[23] Zemansky, Mark W. *Heat and Thermodynamics.* Fifth Edition, McGraw-Hill Book Company, 1968

Appendix A

The ordered triple (z, r, T) showing the temperature of the incident bar

```
0.100000000000E+01    0.635000000000E-01    0.293070296004E+03
     0.100000000000E+01    0.190500000000E+00    0.293066576606E+03
     0.100000000000E+01    0.317500000000E+00    0.293055628076E+03
     0.100000000000E+01    0.444500000000E+00    0.293038808985E+03
     0.100000000000E+01    0.571500000000E+00    0.293026102078E+03
     0.100000000000E+01    0.698500000000E+00    0.293016801554E+03
     0.100000000000E+01    0.825500000000E+00    0.293010212106E+03
     0.100000000000E+01    0.952500000000E+00    0.293005780688E+03
     0.100000000000E+01    0.107950000000E+01    0.293002998003E+03
     0.100000000000E+01    0.120650000000E+01    0.293001425420E+03
     0.300000000000E+01    0.635000000000E-01    0.293001056315E+03
     0.300000000000E+01    0.190500000000E+00    0.293001006884E+03
     0.300000000000E+01    0.317500000000E+00    0.293000869173E+03
     0.300000000000E+01    0.444500000000E+00    0.293000677757E+03
     0.300000000000E+01    0.571500000000E+00    0.293000484827E+03
     0.300000000000E+01    0.698500000000E+00    0.293000317382E+03
     0.300000000000E+01    0.825500000000E+00    0.293000187028E+03
     0.300000000000E+01    0.952500000000E+00    0.293000094759E+03
     0.300000000000E+01    0.107950000000E+01    0.293000035420E+03
     0.300000000000E+01    0.120650000000E+01    0.293000001668E+03
     0.500000000000E+01    0.635000000000E-01    0.293000008989E+03
     0.500000000000E+01    0.190500000000E+00    0.293000008543E+03
     0.500000000000E+01    0.317500000000E+00    0.293000007306E+03
     0.50000000C00E+01    0.444500000000E+00    0.293000005585E+03
     0.500000000000E+01    0.571500000000E+00    0.293000003762E+03
     0.500000000000E+01    0.698500000000E+00    0.293000002133E+03
     0.500000000000E+01    0.825500000000E+00    0.293000000861E+03
     0.500000000000E+01    0.952500000000E+00    0.292999999980E+03
     0.500000000000E+01    0.107950000000E+01    0.292999999440E+03
     0.500000000000E+01    0.120650000000E+01    0.292999999152E+03
     0.700000000000E+01    0.635000000000E-01    0.293000000025E+03
     0.700000000000E+01    0.190500000000E+00    0.293000000023E+03
     0.700000000000E+01    0.317500000000E+00    0.293000000016E+03
     0.700000000000E+01    0.444500000000E+00    0.293000000006E+03
     0.700000000000E+01    0.571500000000E+00    0.292999999996E+03
     0.700000000000E+01    0.698500000000E+00    0.292999999988E+03
     0.700000000000E+01    0.825500000000E+00    0.292999999982E+03
     0.700000000000E+01    0.952500000000E+00    0.292999999980E+03
     0.700000000000E+01    0.107950000000E+01    0.292999999979E+03
     0.700000000000E+01    0.120650000000E+01    0.292999999979E+03
     0.900000000000E+01    0.635000000000E-01    0.293000000000E+03
     0.900000000000E+01    0.190500000000E+00    0.293000000000E+03
     0.900000000000E+01    0.317500000000E+00    0.293000000000E+03
     0.900000000000E+01    0.444500000000E+00    0.293000000000E+03
     0.900000000000E+01    0.571500000000E+00    0.293000000000E+03
```

```
0.900000000000E+01   0.698500000000E+00   0.293000000000E+03
0.900000000000E+01   0.825500000000E+00   0.293000000000E+03
0.900000000000E+01   0.952500000000E+00   0.293000000000E+03
0.900000000000E+01   0.107950000000E+01   0.293000000000E+03
0.900000000000E+01   0.120650000000E+01   0.293000000000E+03
0.110000000000E+02   0.635000000000E-01   0.293000000000E+03
0.110000000000E+02   0.190500000000E+00   0.293000000000E+03
0.110000000000E+02   0.317500000000E+00   0.293000000000E+03
0.110000000000E+02   0.444500000000E+00   0.293000000000E+03
0.110000000000E+02   0.571500000000E+00   0.293000000000E+03
0.110000000000E+02   0.698500000000E+00   0.293000000000E+03
0.110000000000E+02   0.825500000000E+00   0.293000000000E+03
0.110000000000E+02   0.952500000000E+00   0.293000000000E+03
0.110000000000E+02   0.107950000000E+01   0.293000000000E+03
0.110000000000E+02   0.120650000000E+01   0.293000000000E+03
0.130000000000E+02   0.635000000000E-01   0.293000000000E+03
0.130000000000E+02   0.190500000000E+00   0.293000000000E+03
0.130000000000E+02   0.317500000000E+00   0.293000000000E+03
0.130000000000E+02   0.444500000000E+00   0.293000000000E+03
0.130000000000E+02   0.571500000000E+00   0.293000000000E+03
0.130000000000E+02   0.698500000000E+00   0.293000000000E+03
0.130000000000E+02   0.825500000000E+00   0.293000000000E+03
0.130000000000E+02   0.952500000000E+00   0.293000000000E+03
0.130000000000E+02   0.107950000000E+01   0.293000000000E+03
0.130000000000E+02   0.120650000000E+01   0.293000000000E+03
0.150000000000E+02   0.635000000000E-01   0.293000000000E+03
0.150000000000E+02   0.190500000000E+00   0.293000000000E+03
0.150000000000E+02   0.317500000000E+00   0.293000000000E+03
0.150000000000E+02   0.444500000000E+00   0.293000000000E+03
0.150000000000E+02   0.571500000000E+00   0.293000000000E+03
0.150000000000E+02   0.698500000000E+00   0.293000000000E+03
0.150000000000E+02   0.825500000000E+00   0.293000000000E+03
0.150000000000E+02   0.952500000000E+00   0.293000000000E+03
0.150000000000E+02   0.107950000000E+01   0.293000000000E+03
0.150000000000E+02   0.120650000000E+01   0.293000000000E+03
0.170000000000E+02   0.635000000000E-01   0.293000000000E+03
0.170000000000E+02   0.190500000000E+00   0.293000000000E+03
0.170000000000E+02   0.317500000000E+00   0.293000000000E+03
0.170000000000E+02   0.444500000000E+00   0.293000000000E+03
0.170000000000E+02   0.571500000000E+00   0.293000000000E+03
0.170000000000E+02   0.698500000000E+00   0.293000000000E+03
0.170000000000E+02   0.825500000000E+00   0.293000000000E+03
0.170000000000E+02   0.952500000000E+00   0.293000000000E+03
0.170000000000E+02   0.107950000000E+01   0.293000000000E+03
0.170000000000E+02   0.120650000000E+01   0.293000000000E+03
0.190000000000E+02   0.635000000000E-01   0.293000000000E+03
0.190000000000E+02   0.190500000000E+00   0.293000000000E+03
0.190000000000E+02   0.317500000000E+00   0.293000000000E+03
0.190000000000E+02   0.444500000000E+00   0.293000000000E+03
0.190000000000E+02   0.571500000000E+00   0.293000000000E+03
0.190000000000E+02   0.698500000000E+00   0.293000000000E+03
0.190000000000E+02   0.825500000000E+00   0.293000000000E+03
0.190000000000E+02   0.952500000000E+00   0.293000000000E+03
0.190000000000E+02   0.107950000000E+01   0.293000000000E+03
0.190000000000E+02   0.120650000000E+01   0.293000000000E+03
0.210000000000E+02   0.635000000000E-01   0.293000000000E+03
0.210000000000E+02   0.190500000000E+00   0.293000000000E+03
0.210000000000E+02   0.317500000000E+00   0.293000000000E+03
0.210000000000E+02   0.444500000000E+00   0.293000000000E+03
0.210000000000E+02   0.571500000000E+00   0.293000000000E+03
0.210000000000E+02   0.698500000000E+00   0.293000000000E+03
0.210000000000E+02   0.825500000000E+00   0.293000000000E+03
0.210000000000E+02   0.952500000000E+00   0.293000000000E+03
0.210000000000E+02   0.107950000000E+01   0.293000000000E+03
0.210000000000E+02   0.120650000000E+01   0.293000000000E+03
0.230000000000E+02   0.635000000000E-01   0.293000000000E+03
0.230000000000E+02   0.190500000000E+00   0.293000000000E+03
0.230000000000E+02   0.317500000000E+00   0.293000000000E+03
```

```
0.230000000000E+02   0.444500000000E+00   0.293000000000E+03
0.230000000000E+02   0.571500000000E+00   0.293000000000E+03
0.230000000000E+02   0.698500000000E+00   0.293000000000E+03
0.230000000000E+02   0.825500000000E+00   0.293000000000E+03
0.230000000000E+02   0.952500000000E+00   0.293000000000E+03
0.230000000000E+02   0.107950000000E+01   0.293000000000E+03
0.230000000000E+02   0.120650000000E+01   0.293000000000E+03
0.250000000000E+02   0.635000000000E-01   0.293000000000E+03
0.250000000000E+02   0.190500000000E+00   0.293000000000E+03
0.250000000000E+02   0.317500000000E+00   0.293000000000E+03
0.250000000000E+02   0.444500000000E+00   0.293000000000E+03
0.250000000000E+02   0.571500000000E+00   0.293000000000E+03
0.250000000000E+02   0.698500000000E+00   0.293000000000E+03
0.250000000000E+02   0.825500000000E+00   0.293000000000E+03
0.250000000000E+02   0.952500000000E+00   0.293000000000E+03
0.250000000000E+02   0.107950000000E+01   0.293000000000E+03
0.250000000000E+02   0.120650000000E+01   0.293000000000E+03
0.270000000000E+02   0.635000000000E-01   0.293000000000E+03
0.270000000000E+02   0.190500000000E+00   0.293000000000E+03
0.270000000000E+02   0.317500000000E+00   0.293000000000E+03
0.270000000000E+02   0.444500000000E+00   0.293000000000E+03
0.270000000000E+02   0.571500000000E+00   0.293000000000E+03
0.270000000000E+02   0.698500000000E+00   0.293000000000E+03
0.270000000000E+02   0.825500000000E+00   0.293000000000E+03
0.270000000000E+02   0.952500000000E+00   0.293000000000E+03
0.270000000000E+02   0.107950000000E+01   0.293000000000E+03
0.270000000000E+02   0.120650000000E+01   0.293000000000E+03
0.290000000000E+02   0.635000000000E-01   0.293000000000E+03
0.290000000000E+02   0.190500000000E+00   0.293000000000E+03
0.290000000000E+02   0.317500000000E+00   0.293000000000E+03
0.290000000000E+02   0.444500000000E+00   0.293000000000E+03
0.290000000000E+02   0.571500000000E+00   0.293000000000E+03
0.290000000000E+02   0.698500000000E+00   0.293000000000E+03
0.290000000000E+02   0.825500000000E+00   0.293000000000E+03
0.290000000000E+02   0.952500000000E+00   0.293000000000E+03
0.290000000000E+02   0.107950000000E+01   0.293000000000E+03
0.290000000000E+02   0.120650000000E+01   0.293000000000E+03
```

Appendix B

The ordered triple (z, r, T) showing the temperature of the specimen

```
0.150000000000E+00   0.111729853032E+00   0.293066691365E+03
0.150000000000E+00   0.204289893928E+00   0.293066918118E+03
0.150000000000E+00   0.264545599562E+00   0.293067137112E+03
0.150000000000E+00   0.313273477736E+00   0.293067355377E+03
0.150000000000E+00   0.355338908223E+00   0.293067573406E+03
0.450000000000E+00   0.111729853032E+00   0.293044360472E+03
0.450000000000E+00   0.204289893928E+00   0.293044587225E+03
0.450000000000E+00   0.264545599562E+00   0.293044806219E+03
0.450000000000E+00   0.313273477736E+00   0.293045024484E+03
0.450000000000E+00   0.355338908223E+00   0.293045242513E+03
0.750000000000E+00   0.111729853032E+00   0.293027612303E+03
0.750000000000E+00   0.204289893928E+00   0.293027839055E+03
0.750000000000E+00   0.264545599562E+00   0.293028058049E+03
0.750000000000E+00   0.313273477736E+00   0.293028276314E+03
0.750000000000E+00   0.355338908223E+00   0.293028494343E+03
0.105000000000E+01   0.111729853032E+00   0.293016446856E+03
0.105000000000E+01   0.204289893928E+00   0.293016673609E+03
0.105000000000E+01   0.264545599562E+00   0.293016892603E+03
0.105000000000E+01   0.313273477736E+00   0.293017110868E+03
0.105000000000E+01   0.355338908223E+00   0.293017328897E+03
0.135000000000E+01   0.111729853032E+00   0.293010864133E+03
0.135000000000E+01   0.204289893928E+00   0.293011090885E+03
0.135000000000E+01   0.264545599562E+00   0.293011309880E+03
0.135000000000E+C1   0.313273477736E+00   0.293011528145E+03
0.135000000000E+01   0.355338908223E+00   0.293011746173E+03
0.149905633063E+00   0.111725948199E+00   0.293049313670E+03
0.149905633063E+00   0.204282754224E+00   0.293049439559E+03
0.149905633063E+00   0.264536353988E+00   0.293049561141E+03
0.149905633063E+00   0.313262529177E+00   0.293049682317E+03
0.149905633063E+00   0.355326489525E+00   0.293049803363E+03
0.449926603494E+00   0.111725948199E+00   0.293036914091E+03
0.449926603494E+00   0.204282754224E+00   0.293037039980E+03
0.449926603494E+00   0.264536353988E+00   0.293037161561E+03
0.449926603494E+00   0.313262529177E+00   0.293037282738E+03
0.449926603494E+00   0.355326489525E+00   0.293037403783E+03
0.749947573924E+00   0.111725948199E+00   0.293027614416E+03
0.749947573924E+00   0.204282754224E+00   0.293027740305E+03
0.749947573924E+00   0.264536353988E+00   0.293027861886E+03
0.749947573924E+00   0.313262529177E+00   0.293027983063E+03
0.749947573924E+00   0.355326489525E+00   0.293028104108E+03
0.104996854435E+01   0.111725948199E+00   0.293021414634E+03
0.104996854435E+01   0.204282754224E+00   0.293021540522E+03
0.104996854435E+01   0.264536353988E+00   0.293021662104E+03
0.104996854435E+01   0.313262529177E+00   0.293021783280E+03
0.104996854435E+01   0.355326489525E+00   0.293021904326E+03
```

```
 0.134998951478E+01   0.111725948199E+00   0.293018314743E+03
 0.134998951478E+01   0.204282754224E+00   0.293018440632E+03
 0.134998951478E+01   0.264536353988E+00   0.293018562213E+03
 0.134998951478E+01   0.313262529177E+00   0.293018683390E+03
 0.134998951478E+01   0.355326489525E+00   0.293018804418E+03
 0.149794173870E+00   0.111721336627E+00   0.293008312268E+03
 0.149794173870E+00   0.204274322302E+00   0.293008319273E+03
 0.149794173870E+00   0.264525435054E+00   0.293008326039E+03
 0.149794173870E+00   0.313249599034E+00   0.293008332782E+03
 0.149794173870E+00   0.355311823160E+00   0.293008339518E+03
 0.449839913010E+00   0.111721336627E+00   0.293007622140E+03
 0.449839913010E+00   0.204274322302E+00   0.293007629145E+03
 0.449839913010E+00   0.264525435054E+00   0.293007635911E+03
 0.449839913010E+00   0.313249599034E+00   0.293007642654E+03
 0.449839913010E+00   0.355311823160E+00   0.293007649390E+03
 0.749885652150E+00   0.111721336627E+00   0.293007104579E+03
 0.749885652150E+00   0.204274322302E+00   0.293007111584E+03
 0.749885652150E+00   0.264525435054E+00   0.293007118350E+03
 0.749885652150E+00   0.313249599034E+00   0.293007125093E+03
 0.749885652150E+00   0.355311823160E+00   0.293007131829E+03
 0.104993139129E+01   0.111721336627E+00   0.293006759535E+03
 0.104993139129E+01   0.204274322302E+00   0.293006766540E+03
 0.104993139129E+01   0.264525435054E+00   0.293006773306E+03
 0.104993139129E+01   0.313249599034E+00   0.293006780049E+03
 0.104993139129E+01   0.355311823160E+00   0.293006786785E+03
 0.134997713043E+01   0.111721336627E+00   0.293006587012E+03
 0.134997713043E+01   0.204274322302E+00   0.293006594018E+03
 0.134997713043E+01   0.264525435054E+00   0.293006600783E+03
 0.134997713043E+01   0.313249599034E+00   0.293006607526E+03
 0.134997713043E+01   0.355311823160E+00   0.293006614210E+03
 0.149752259567E+00   0.111719602591E+00   0.293011224385E+03
 0.149752259567E+00   0.204271151742E+00   0.293011273020E+03
 0.149752259567E+00   0.264521329332E+00   0.293011319991E+03
 0.149752259567E+00   0.313244737060E+00   0.293011366806E+03
 0.149752259567E+00   0.355306308334E+00   0.293011413570E+03
 0.449807312997E+00   0.111719602591E+00   0.293006432664E+03
 0.449807312997E+00   0.204271151742E+00   0.293006481299E+03
 0.449807312997E+00   0.264521329332E+00   0.293006528270E+03
 0.449807312997E+00   0.313244737060E+00   0.293006575085E+03
 0.449807312997E+00   0.355306308334E+00   0.293006621849E+03
 0.749862366426E+00   0.111719602591E+00   0.293002838873E+03
 0.749862366426E+00   0.204271151742E+00   0.293002887509E+03
 0.749862366426E+00   0.264521329332E+00   0.293002934480E+03
 0.749862366426E+00   0.313244737060E+00   0.293002981295E+03
 0.749862366426E+00   0.355306308334E+00   0.293003028059E+03
```

Appendix C

Program listing

```
        program energy

******************* Declaration of Variables ************************
c
c units will be cm, g, microsec, and Mbars.
c
double precision con0,con1,con2,con3,con4,con5,con6

integer znumpts, rnumpts, arcs, i, j, flag_def
integer rbnumpts, zbnumpts,t ,tnumpts, timestep
        integer con1flag, strnumpts

character fileenergy*16

        parameter (rnumpts=5, znumpts=5, tnumpts=399, arcs=100)
        parameter (rbnumpts = 10, zbnumpts =15)

double precision  deltaz, deltal, deltat
        double precision  rpos(0:rnumpts,0:tnumpts)
        double precision  zpos(0:znumpts,0:tnumpts)
double precision  length_s, zold, znew, radius_s, rold, rnew
c
c gh, gt and gu are used by the ripapart subroutine to
c construct filenames for the various arrays that a
c saved during a program execution
c
integer ch(0:9), gh, gt, gu
c
c fileenergy is the filename used for files created by the
c    energy program
c
c
c Note, in the spline subroutine, n = rnumpts - 2.  This must be set
c   before the program will work properly
c
c Since the radius and length arrays begin at 0,
c    their dimension is numpts - 1.  These arrays keep
c    track of the centroid position.
c
        double precision  r(0:rnumpts-1,0:tnumpts)
        double precision  z(0:znumpts-1,0:tnumpts)
        double precision  rb(0:rbnumpts-1)
        double precision  zb(0:zbnumpts-1)

c
c velocity in both radial and longitudinal directions
c
double precision  uvol, density, mass, t0
```

```
c double precision  tke
c double precision  area_s

        double precision  vz(0:znumpts-1,0:tnumpts)
        double precision  vr(0:rnumpts-1,0:tnumpts)
double precision  vres(0:znumpts-1,0:rnumpts-1,0:tnumpts)
double precision  ke(0:znumpts-1,0:rnumpts-1,0:tnumpts)
double precision  radius_bar, length_bar, E
double precision  workin(1:tnumpts)
double precision  inteng(0:znumpts-1,0:rnumpts-1,0:tnumpts)

c double precision  workin_s(0:znumpts-1,0:rnumpts-1,0:tnumpts)

double precision  deltie(0:znumpts-1,0:rnumpts-1,0:tnumpts)

c Temperature buildup in the specimen at the specimen/incident bar
c interface.

double precision  us(0:znumpts-1,0:rnumpts-1,0:tnumpts)

c Temperature profile in the incident bar as a result of the heat
c transfer from the specimen.

double precision  ub(0:zbnumpts,0:rbnumpts,0:tnumpts)
double precision  usi(0:rbnumpts,0:tnumpts)
        double precision  r_spline(0:rnumpts-1), u_spline(0:rnumpts-1)
c
c ac, bc, cc, dc, are the cubic polynomial coefficients ie
c  f_i(x) = ac_i + bc_i*(x-x_i) + cc_i*(x-x_i)**2 + dc_i*(x-x_i)**3
c  and act,bct, etc. are two dim arrays keeping track of the
c  coefficients of the cubic splines as time changes
c
        double precision act(0:rnumpts-2, 1:tnumpts)
        double precision bct(0:rnumpts-2, 1:tnumpts)
        double precision cct(0:rnumpts-2, 1:tnumpts)
        double precision dct(0:rnumpts-2, 1:tnumpts)
        double precision ac(0:rnumpts-2), bc(0:rnumpts-2)
        double precision cc(0:rnumpts-2), dc(0:rnumpts-2)
        double precision strdiff(1:tnumpts), strsum(1:tnumpts)
c
        double precision xj, xi, xt, pi
c
c floating point values of rnumpts, znumpts and arcs.
c
        double precision xrnumpts, xznumpts, xarcs

common gh,gt,gu

        integer  ii, jj, jb, ib, jbfix, rb_counter

double precision diffcons, deltarb, deltazb, diffconb
double precision deltar_s, deltaz_s, avg_ub, sum
c
        double precision s1, s2, s_s1, s_s2, root, f, dfdx
        double precision rupper, rlower, Tinitial
double precision tapp3, tapp4, tapp5, tapp6, tapp8

        common/coef/con1,con2,con3,con4,con5,con6
        external f, dfdx

        pi = acos(-1.)
************************ read in the strain data ********************

strnumpts = 0
j = 1

        open (unit=22, file='strain.txt',status='old')
```

```
read(22,*)k

do 3 i = 1, k
  read(22,*)straini, strainr, straint
c   print*, straini, strainr, straint
c
c the counter strnumpts records the number of time steps that
c specimen strain is positive.
c strain values that do not reflect work being done on the specimen
c that is strain_i>(strain_e+strain_t) are filtered out
c
              if(strainr.lt.0)then
                 strainr=-1.*strainr
              endif

              if(straint.lt.0)then
                 straint=-1.*straint
              endif

c     print*, straini, strainr, straint

            if (
c     1          straini.gt.(strainr+straint).and.
     1          straini.gt.0 )then
c              print *, i
      strdiff(j)=(straini-strainr-straint)/1e6
      strsum(j) =(straini+strainr+straint)/1e6

c       print*,'strdiff(',j,')=',strdiff(j)
c       print*,'strsum(',j,')=',strsum(j)
c              read*
c              strdiff(j)=1800.99*float(j)**10.04723 /
c     1                   ( (1+float(j))**9.77690*1e6 )

      strnumpts = strnumpts + 1
              j = j+1

    endif


  3 continue
close(22)
print*,'strnumpts =', strnumpts
c read*
c
*********** Read in dimensions and materaial properties ************
        open (unit=22, file='dimen.txt',status='old')
c
c all measurements are given in centimeters
c
read(22,*)radius_bar, length_bar, radius_s, length_s
c
c Modulus of elasticity is measured in Mbars
c density of tantalum in gm/cm^3
c time is in microseconds
c
read(22,*)E, density, deltat
c
c thermal diffusion constants for both bar and specimen
c
read(22,*)diffconb, diffcons
close(22)
*********** Intialization of parameters and arrays ******************
        rold       = radius_s
c
c Since we are using the full strain data we need to work with the
c entire length of the specimen.
c
```

```
        zold       = length_s
        xrnumpts   = float(rnumpts)
        xznumpts   = float(znumpts)
        deltaz     = (length_s)/(xznumpts)
        xarcs      = float(arcs)
c
c calculation of one 'unit' of volume
c
        uvol = ( (pi*rold**2*deltaz)/xarcs )/xrnumpts
mass   = uvol*density
c
c This character table is used in the construction of the files names
c for the various arrays at each time step.  This keeps each file
c down to a workable size and helps to identify just what it is
c you're looking at
c
        ch(0) = ichar('0')
        ch(1) = ichar('1')
        ch(2) = ichar('2')
        ch(3) = ichar('3')
        ch(4) = ichar('4')
        ch(5) = ichar('5')
        ch(6) = ichar('6')
        ch(7) = ichar('7')
        ch(8) = ichar('8')
        ch(9) = ichar('9')
c
*************** Initiallizing the zpos array **********************
c
c Since there is equal spacing along z, zpos can be determined
c by simply multiplying the z counter, i, by the spacing deltaz
c
      do 5 i = 0,znumpts
         xi=float(i)
         zpos(i,0)=deltaz*xi
   5    continue
c
***************  Initiallizing the rpos array *********************
c
c the volume of each element = uvol, therefore since we are
c dealing with disks
c          2*uvol = pi * rpos**2 * deltaz,
c   we can solve for rpos by simplifying this calculation
c
        do 6 j = 0,rnumpts
         xj = float(j)
         rpos(j,0) = ((xj/(xrnumpts))**.5)*rold
   6    continue
c

        deltarb = radius_bar / float(rbnumpts)
        deltazb = length_bar / float(zbnumpts)

*************** Initiallizing the zb array **********************
c
c Since there is equal spacing along z, zb can be determined
c by simply multiplying the z counter, i, by the spacing deltazb
c
      do 13 i = 0,zbnumpts-1
         xi=float(i)
         zb(i)=(deltazb*xi+deltazb*(xi+1.0))/2.
  13    continue
c
***************  Initiallizing the rb array *********************
c
c Since there is equal spacing along the bar radius, rb can be determined
c by simply multiplying the rb counter, i, by the spacing deltarb
c
```

```
      do 14 j = 0,rbnumpts-1
         xj=float(j)
         rb(j)=( deltarb*xj+deltarb*(xj+1) )/2.
  14    continue
c
***************** Initiallizing the rest of the arrays **************
c
c Initial temperature is 293 K
c Temperature is measured in Kelvin
c
       do 10 i = 0, znumpts-1
        do 11 j = 0, rnumpts-1
             vz(i,0)       = 0
             vr(j,0)       = 0
             vres(i,j,0)   = 0
     inteng(i,j,0) = 0
             ke(i,j,0)     = 0
  11    continue
  10   continue
c
*************  Begin Calculations ***********************************
c
c Solve for the position arrays first, since these are used
c to solve for all other arrays
c
*** The zpos array is determined for all t ******
c
       do 7 t = 1, strnumpts
c
c using the strain data we solve for the change in specimen length
c
       deltal = strdiff(t)*zold

c
c The new zpos at the interface (i = 0) will be zpos at t-1 plus
c the change in specimen length brought about by specimen strain
c
            zpos(0,t) = zpos(0,t-1) + deltal
c             print*,'zpos(0,',t,')=',zpos(0,t)
            znew = zold - deltal
            deltaz = znew/(xznumpts)
c      print*,'deltaz =',deltaz
c
c the other end of the specimen doesn't move
c
            zpos(znumpts,t) = length_s

          do 15 i = 1, znumpts - 1

c
c once a new deltaz has been calculated, zpos is completed
c
              zpos(i,t) = zpos(0,t) + dfloat(i)*deltaz
c              print*,'zpos(',i,',',t,')=',zpos(i,t)

  15      continue
c
c reset zold to its new value before going through the loop again
c
       zold = znew
   7   continue
c
**** The rpos array is determined for all t *****
c
      do 12 t = 1, strnumpts
       do 16 j = 0, rnumpts
         xj = float(j)
```

```
c
c Since we have conservation of volume, knowing the change in
c length we can determine the change in radius
c
           zold = zpos(znumpts,t-1) - zpos(0,t-1)
           znew = zpos(znumpts,t) - zpos(0,t)
           rnew = (((rold**2)*zold)/znew)**.5
           rpos(j,t) = ((xj/(xrnumpts))**.5)*rnew
c            print*,'rpos(',j,',',t,')=',rpos(j,t)
   16    continue
         rold = rnew
c
c If the specimen radius exceeds the incident bar radius the
c program prints out a warning
c
         if(flag_def.ne.1 .and. rold.ge.(radius_bar))then
           print*,'Specimen deformation exceeds incident bar radius'
   print*,'at t = ',t
   flag_def = 1
         endif
   12   continue
c
**************** The z and r arrays *************
c
c We can solve for the z and r arrays using rpos and zpos
c z and r are the coordinates of the centroid of an element
c
        do 22 t = 0,strnumpts
         do 25 i = 0,znumpts - 1
             z(i,t) = (zpos(i,t) + zpos(i + 1,t))/2.0
c print*,'z(',i,',',t,')=',z(i,t)
           do 26 j = 0,rnumpts - 1
             r(j,t) = (1./3.)*(xarcs/pi)*SIN(2.*pi/xarcs)*
     1     (( rpos(j,t)**2 + rpos(j,t)*rpos(j+1,t) + rpos(j+1,t)**2 )/
     2                           (rpos(j,t) + rpos(j + 1,t)) )
c print*,'r(',j,',',t,')=',r(j,t)
  26       continue
  25     continue
  22    continue
c
************ Velocity, Kinetic and Internal Energy ********
c
        do 32 t = 1, strnumpts - 1

c here we calculate work_in as outlined in the thesis
c based on Pat's paper. uniformly distributed.  If the
c term xznumpts does not appear in the denominator
c we are solving for the total work_in

workin(t)=E*strsum(t)/2*pi*radius_bar**2*strdiff(t)*
     1             ( zpos(znumpts,t)-zpos(0,t) ) /
     2             ( xarcs*xrnumpts*xznumpts )


c here we calculate work_in as it relates to KE
c this is the total work to one sector of the specimen
c

c workin(t)=E*strsum(t)/2*pi*radius_bar**2*strdiff(t)*
c     1             ( zpos(znumpts,t)-zpos(0,t) ) /
c     2             ( xarcs )

c          tke = 0

         do 40 i = 0, znumpts - 1
           vz(i,t) = (z(i,t+1)-z(i,t-1))/(2.0*deltat)
             do 41 j = 0,rnumpts - 1
               vr(j,t) = (r(j,t+1)-r(j,t-1))/(2.0*deltat)
```

```
       vres(i,j,t) = ( vr(j,t)**2 + vz(i,t)**2 )**0.5
c        print*,'vres(',i,',',j,',',t,') =',vres(i,j,t)
       ke(i,j,t) = 0.5*mass*vres(i,j,t)**2
c               tke = tke + ke(i,j,t)

c               if( tke.lt.0 )then
c                 print*,'total ke is zero'
c                 read*
c               endif

c here we calculate work_in as also being dependent
c on the specimen radius but still uniformly distributed

c  rnew = rpos(rnumpts,t)-rpos(0,t)
c  area_s=( rpos(j+1,t)**2-rpos(j,t)**2 )/( rnew**2*xarcs )

c  workin(t)=E*strsum(t)/2*area_s*strdiff(t)*
c     1              ( zpos(znumpts,t)-zpos(0,t) ) /
c     2              (xrnumpts*xarcs*xznumpts)


       inteng(i,j,t) = workin(t) - ke(i,j,t)

c the max function, returns a 0 is inteng is less than
c workin(t) - ke(i,j,t).

       if ( inteng(i,j,t).lt.0)then
inteng(i,j,t)=-inteng(i,j,t)
c inteng(i,j,t)=0
       endif

  41      continue
  40  continue

c here we calculate work as it relates to KE, that is those
c elements of the specimen undergoing the greatest KE
c should be having the greatest amount of work done on them.
c In this case we total up the KE for all i and j and designate
c this total as tke.  The proportion of work assigned to each
c element then is a proportion ke(i,j,t)/tke


c         do 42 i = 0, znumpts - 1
c          do 43 j = 0,rnumpts - 1
c          workin_e(i,j,t) = workin(t)*ke(i,j,t)/tke
c   inteng(i,j,t) = workin_e(i,j,t) - ke(i,j,t)

c          if( inteng(i,j,t).lt.0 ) then
c             print*,'Delta_IE is negative at t=',t
c          endif

c the max function, returns a 0 is inteng is less than
c workin(t) - ke(i,j,t).

c        if ( inteng(i,j,t).lt.0)then
c inteng(i,j,t)=0
c        endif

c  43     continue
c  42    continue

  32 continue
c
c Since we are using a second order approximation for velocity,
c the last element in the array must be calculated separately
c as a first order approximation.
c
c        tke = 0
```

```
        t = strnumpts

 workin(t)=E*strsum(t)/2*pi*radius_bar**2*strdiff(t)*
      1             ( zpos(znumpts,t)-zpos(0,t) )/
      2             ( xarcs*xrnumpts*xznumpts )

 c workin(t)=E*strsum(t)/2*pi*radius_bar**2*strdiff(t)*
 c    1             ( zpos(znumpts,t)-zpos(0,t) )/
 c    2             ( xarcs )


         do 45 i = 0, znumpts - 1
           vz(i,t) = (z(i,t)-z(i,t-1))/(deltat)
             do 56 j = 0,rnumpts - 1
               vr(j,t) = (r(j,t)-r(j,t-1))/(deltat)
       vres(i,j,t) = ( vr(j,t)**2 + vz(i,t)**2 )**0.5
 c       print*,'vres(',i,',',j,',',t,') =',vres(i,j,t)
       ke(i,j,t) = 0.5*mass*vres(i,j,t)**2
 c              tke = tke + ke(i,j,t)

 c              if( tke.lt.0 )then
 c                print*,'total ke is zero'
 c                read*
 c              endif

 c here we calculate work_in as also being dependent
 c on the specimen radius

 c  rnew = rpos(rnumpts,t)-rpos(0,t)
 c  area_s=( rpos(j+1,t)**2-rpos(j,t)**2 )/( rnew**2*xarcs )

 c  workin(t)=E*strsum(t)/2*area_s*strdiff(t)*
 c    1             ( zpos(znumpts,t)-zpos(0,t) ) /
 c    2             (xrnumpts*xarcs*xznumpts)

       inteng(i,j,t) = workin(t) - ke(i,j,t)

 c the max function, returns a 0 is inteng is less than
 c workin(t) - ke(i,j,t).

       if ( inteng(i,j,t).lt.0)then
 inteng(i,j,t)=-inteng(i,j,t)
 c inteng(i,j,t)=0
       endif

   56      continue
   45  continue

 c here we calculate work as it relates to KE, that is those
 c elements of the specimen undergoing the greatest KE
 c should be having the greatest amount of work done on them.
 c In this case we total up the KE for all i and j and designate
 c this total as tke.  The proportion of work assigned to each
 c element then is a proportion ke(i,j,t)/tke


 c         do 46 i = 0, znumpts - 1
 c          do 47 j = 0,rnumpts - 1
 c          workin_e(i,j,t) = workin(t)*ke(i,j,t)/tke
 c   inteng(i,j,t) = workin_e(i,j,t) - ke(i,j,t)

 c          if( inteng(i,j,t).lt.0 ) then
 c             print*,'Delta_IE is negative at t=',t
 c          endif

 c the max function, returns a 0 is inteng is less than
 c workin(t) - ke(i,j,t).
```

```
c        if ( integ(i,j,t).lt.0)then
c integ(i,j,t)=0
c        endif

c  47      continue
c  46    continue

t0 = 293
do 115 t = 0,strnumpts
  do 116 i = 0, znumpts -1
    do 117 j = 0,rnumpts -1
      us(i,j,t) = t0
  117     continue
  116   continue
  do 118 ib = 0, zbnumpts
    do 119 jb = 0,rbnumpts
      ub(ib,jb,t) = t0
  119     continue
  118   continue
  115 continue
c
******* Calculating the temperature rise in the specimen ************
c
c Using specific volume to solve for temperature
c we are given 180.9479 grams/mole as the atomic weight of Tantalum
c From TAPP the specific heat of solids are expressed as a function
c of temperature with the equation:
c C_v(T)=-C_3 - 2(C_4E-3)*T - 2(C_5E6)T^-2 - 6(C_6E-6)*T^2
c             + .25(C_8)*T^-0.5 + C_p,mag
c where
tapp3     =   -47.6677*1e-5
       tapp4     =     5.45091*1e-5
       tapp5     =    -0.21647*1e-5
       tapp6     =    -0.59727*1e-5
       tapp8     = -1669.643*1e-5
c
c As a check we will also solve for Temperature based on a
c constant C_v.  Further more, if we shut diffusion off we should
c be able to sit back and watch the temperature rise in the
c specimen.  In this way we can compare the temperature rise in
c the specimen with and without diffusion.
c We can then determine how much bleeds out as a temperature
c loss due to diffusion.
c
con0 = density*uvol*5.526453e-3
c print*,'con0=',con0
c coeff of T term
con1 = -1.0*con0*tapp3
c print*,'con1=',con1
c coeff of T^2 term
con2 = -1.0*con0*tapp4*1e-3
c print*,'con2=',con2
c coeff of T^-1
con3 = 2.0*con0*tapp5*1e6
c print*,'con3=',con3
c coeff of T^3
con4 = -2.0*con0*tapp6*1e-6
c print*,'con4=',con4
c coeff of T^.5
con5 = .5*con0*tapp8
c print*,'con5=',con5

do 700 t = 1, strnumpts
c  print*,'beginning of t loop t=',t
 do 701 i = 0, znumpts-1
  do 702 j = 0, rnumpts-1
```

```
c     Tinitial = con1*us(i,j,t-1)+con2*us(i,j,t-1)**2 +
c     1 con4*us(i,j,t-1)**3+con3*1.0/us(i,j,t-1) +
c     2          con5*us(i,j,t-1)**.5

    Tinitial = con1*t0+con2*t0**2 +
     1 con4*t0**3+con3*1.0/t0 +
     2          con5*t0**.5



           deltIE(i,j,t) = inteng(i,j,t)
c
c ************ Constant C_v calculations *********************

c The integral to solve for T_f is easily done leaving us with
c          us(i,j,t) = deltIE(i,j,t)/(con0*1.)+us(i,j,t-1)

c ************************************************************
c
    con6 = deltIE(i,j,t)+Tinitial
c            print*,'con6=',con6
c     print*,'initial us(',i,',',j,',',t-1,') =',us(i,j,t-1)
    call newton( us(i,j,t-1) ,0.0001,10,215.0,1,root,f,dfdx,n)
c     print*, 'root is ',ROOT,' in ',N,' iterations'
           us(i,j,t) = root
c     print*,'us(',i,',',j,',',t,') =',us(i,j,t)
 702   continue
c
**** Calculating the arguements for the spline routine ****
c
       nspline = rnumpts
       if(i.eq.0) then
         do 148 j = 0, rnumpts - 1
c
c Rspline and qspline are used so that only a one-dimensional
c array needs to be passed to the spline routine.
c
           r_spline(j) = r(j,t)
           u_spline(j) = us(0,j,t)
 148      continue
c
c The spline routine returns the coefficients of the cubic spline
c polynomials needed to interface between the speciman and the
c incident bar.
c
        call spline(nspline,r_spline,u_spline,ac,bc,cc,dc)
        do 51 j = 0, rnumpts - 2
            act(j,t) = ac(j)
            bct(j,t) = bc(j)
            cct(j,t) = cc(j)
            dct(j,t) = dc(j)
c           print*,'act(', j,',',t,') =',act(j,t)
c           print*,'bct(', j,',',t,') =',bct(j,t)
c           print*,'cct(', j,',',t,') =',cct(j,t)
c           print*,'dct(', j,',',t,') =',dct(j,t)
c           read*
 51     continue
       endif
 701  continue
c
c **************** Incident bar diffusion ********************
c
       s1=diffconb*deltat/deltarb**2
       s2=diffconb*deltat/deltazb**2
c      print *,'s1 =',s1
c      print *,'s2 =',s2
         j = rnumpts - 1
c         print*,'j =',j
```

```
   counter = 1
   sum = 0
   con1flag = 0
           rupper = r(j,t)
c   print*,'rupper=',r(j,t)
           rlower = r(j - 1,t)
           do  86 ib = 0, zbnumpts
             do 65 jb = rbnumpts, 0 ,-1
               xj = float(jb)
               xi = float(ib)
               xt = float(t)
c ------------------------------------------------------------------
c Condition 1
if (xj*deltarb.le.rupper .and. ib.eq.0 .and. con1flag.ne.1)then
rb_counter = jb
c print*,'rb_counter =',rb_counter
do 48 jj=jb, 0, -1
  xj=float(jj)
    if (xj*deltarb.le.rupper .and. xj*deltarb.gt.rlower)then
             usi(jj,t) = act(j-counter,t) +
     1                   bct(j-counter,t)*(deltarb*xj - rlower)+
     2                   cct(j-counter,t)*(deltarb*xj - rlower)**2 +
     3                   dct(j-counter,t)*(deltarb*xj - rlower)**3

c print*,'usi11(',jj,',',t,')=',usi(jj,t)
c these conditions allow for a value in the specimen array to
c be used in calculating a corresponding value in the incident
c bar.
if( jj.ne.rbnumpts .and. jj.ne.0 ) then
           ub(ib,jj,t)=s1/(2.*xj)*(
     1              (2.*xj+1.)*ub(ib,jj+1,t-1)-4.*xj*ub(ib,jj,t-1)
     2              +(2.*xj-1.)*ub(ib,jj-1,t-1) )
     3        +s2*( ub(ib+1,jj,t-1)-2.*ub(ib,jj,t-1)+usi(jj,t-1) )
     4        +ub(ib,jj,t-1)
endif

if( jj.eq.rbnumpts ) then
           ub(ib,jj,t)=s1/(2.*xj)*(
     1              (2.*xj+1.)*ub(ib,jj,t-1)-4.*xj*ub(ib,jj,t-1)
     2              +(2.*xj-1.)*ub(ib,jj-1,t-1) )
     3         +s2*( ub(ib+1,jj,t-1)-2.*ub(ib,jj,t-1)+usi(jj,t-1) )
     4         +ub(ib,jj,t-1)
endif

        if(jj.eq.0)then
         ub(ib,jj,t)=(4.*s1)*( ub(ib,jj+1,t-1)-ub(ib,jj,t-1) )
     1     +s2*( ub(ib+1,jj,t-1)-2.*ub(ib,jj,t-1)+usi(jj,t-1) )
     2     +ub(ib,jj,t-1)
endif

c print*,'usi12(',jj,',',t,')=',usi(jj,t)

sum = ub(ib,jj,t)+sum
     else
     jj = jj + 1
     rupper = rlower
     counter = counter + 1
c      print*,'counter is now',counter,'at t=',t
     rlower = r(j-counter,t)
               if(counter.eq.rnumpts)then

                jbfix = jj
                 do 320 ii = jbfix - 1,0,-1
                   usi(ii,t) = usi(jbfix,t)

c these conditions allow for a value in the specimen array to
c be used in calculating a corresponding value in the incident
c bar.
```

```
if( ii.ne.rbnumpts .and. ii.ne.0 ) then
          ub(ib,ii,t)=s1/(2.*xj)*(
     1             (2.*xj+1.)*ub(ib,ii+1,t-1)-4.*xj*ub(ib,ii,t-1)
     2             +(2.*xj-1.)*ub(ib,ii-1,t-1) )
     3        +s2*( ub(ib+1,ii,t-1)-2.*ub(ib,ii,t-1)+usi(ii,t-1) )
     4        +ub(ib,ii,t-1)
endif

if( ii.eq.rbnumpts ) then
          ub(ib,ii,t)=s1/(2.*xj)*(
     1             (2.*xj+1.)*ub(ib,ii,t-1)-4.*xj*ub(ib,ii,t-1)
     2             +(2.*xj-1.)*ub(ib,ii-1,t-1) )
     3         +s2*( ub(ib+1,ii,t-1)-2.*ub(ib,ii,t-1)+usi(ii,t-1) )
     4         +ub(ib,ii,t-1)
endif

       if(ii.eq.0)then
        ub(ib,ii,t)=(4.*s1)*( ub(ib,ii+1,t-1)-ub(ib,ii,t-1) )
     1    +s2*( ub(ib+1,ii,t-1)-2.*ub(ib,ii,t-1)+usi(ii,t-1) )
     2    +ub(ib,ii,t-1)
endif
sum = ub(ib,ii,t) + sum
  320           continue
       jj = -1
              coniflag = 1
             endif
          endif
  48 continue
endif
c -------------------------------------------------------------------
c Condition 2

       if(jb.eq.0 .and. ib.ne.0 .and. ib.ne.zbnumpts)then
        ub(ib,jb,t)=(4.*s1)*( ub(ib,jb+1,t-1)-ub(ib,jb,t-1) )
     1    +s2*( ub(ib+1,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     2    +ub(ib,jb,t-1)

c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c              endif
c  print*,'ub2(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c  read*

        endif
c -------------------------------------------------------------------
c Condition 3
       if((xj*deltarb).gt.r(rnumpts-1,t) .and. ib.eq.0
     1  .and. coniflag.ne.1 .and. jb.ne.rbnumpts)then

c print*,'loop 3 calculations'
c print*,'xj*deltarb=',xj*deltarb,'r max at',t,'is',r(rnumpts-1,t)
ub(ib,jb,t)=s1/(2.*xj)*(
     1             (2.*xj+1.)*ub(ib,jb+1,t-1)-4.*xj*ub(ib,jb,t-1)
     2             +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib+1,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib,jb,t-1) )
     4        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c              endif
c print*,'ub3(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

       endif
c -------------------------------------------------------------------
```

```
c Condition 4

        if(ib.eq.0 .and. jb.eq.rbnumpts)then
           ub(ib,jb,t)=s1/(2.*xj)*(
     1             (2.*xj+1.)*ub(ib,jb,t-1)-4.*xj*ub(ib,jb,t-1)
     2             +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib+1,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib,jb,t-1) )
     4        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c                endif
c print*,'ub4(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

        endif
c -------------------------------------------------------------------
c Condition 5
        if(jb.ge.1 .and. ib.ge.1 .and. ib.ne.zbnumpts
     1                      .and. jb.ne.rbnumpts)then
           ub(ib,jb,t)=s1/(2.*xj)*(
     1             (2.*xj+1.)*ub(ib,jb+1,t-1)-4.*xj*ub(ib,jb,t-1)
     2             +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib+1,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     4        +ub(ib,jb,t-1)


c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c                endif
c print*,'ub5(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*
        endif
c -------------------------------------------------------------------
c Condition 6
        if(jb.eq.0 .and. ib.eq.zbnumpts)then
           ub(ib,jb,t)=(4.*s1)*( ub(ib,jb+1,t-1)-ub(ib,jb,t-1) )
     1        +s2*( ub(ib,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     2        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c                endif
c print*,'ub6(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

        endif
c -------------------------------------------------------------------
c Condition 7
        if(jb.ne.0 .and. ib.eq.zbnumpts .and. jb.ne.rbnumpts)then
           ub(ib,jb,t)=s1/(2.0*xj)*(
     1             (2.*xj+1.)*ub(ib,jb+1,t-1)-4.*xj*ub(ib,jb,t-1)
     2             +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     4        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c                endif
c print*,'ub7(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

        endif
```

```
c -------------------------------------------------------------------
c Condition 8
        if(jb.eq.rbnumpts .and. ib.ne.zbnumpts .and. ib.ne.0)then
           ub(ib,jb,t)=s1/(2.*xj)*(
     1              (2.*xj+1.)*ub(ib,jb,t-1)-4.*xj*ub(ib,jb,t-1)
     2              +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib+1,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     4        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c              endif
c print*,'ub8(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

        endif
c -------------------------------------------------------------------
c Condition 9
        if(jb.eq.rbnumpts .and. ib.eq.zbnumpts)then
           ub(ib,jb,t)=s1/(2.*xj)*(
     1              (2.*xj+1.)*ub(ib,jb,t-1)-4.*xj*ub(ib,jb,t-1)
     2              +(2.*xj-1.)*ub(ib,jb-1,t-1) )
     3        +s2*( ub(ib,jb,t-1)-2.*ub(ib,jb,t-1)+ub(ib-1,jb,t-1) )
     4        +ub(ib,jb,t-1)
c If the heat in the specimen is currently less than that of the bar
c there will be no heat conduction.
c       if( ub(ib,jb,t).lt.ub(ib,jb,t-1) )then
c ub(ib,jb,t)=ub(ib,jb,t-1)
c              endif
c print*,'ub9(',ib,',',jb,',',t,')=',ub(ib,jb,t)
c read*

        endif
  65    continue
avg_ub = sum / rb_counter
  86    continue

*********************************************************************
*                                                                   *
*     Calculating diffusion within the specimen                     *
*                                                                   *
*********************************************************************
c
c The specimen is always changing size thus we need a new
c deltar and deltaz each at each t.

        deltar_s = rpos(rnumpts,t) / float(rnumpts)
        deltaz_s = ( length_s - zpos(0,t) )/float(znumpts)
        s_s1=diffcons*deltat/deltar_s**2
        s_s2=diffcons*deltat/deltaz_s**2

c        print *,'s_s1 =',s_s1
c       print *,'s_s2 =',s_s2
c read*

do 52 i = 0, znumpts-1
  do 53 j = 0, rnumpts-1

          xj = float(j)
          xi = float(i)
          xt = float(t)

c -----------------------------------------------------------------
c Condition 1

        if(i.eq.0 .and. j.eq.0 )then
         us(i,j,t)=(4.*s_s1)*( us(i,j+1,t-1)-us(i,j,t-1) )
     1    +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+avg_ub )
     2    +us(i,j,t-1)

       endif
c ------------------------------------------------------------------
c Condition 2

       if(j.eq.0 .and. i.ne.0 .and. i.ne.znumpts)then

        us(i,j,t)=(4.*s_s1)*( us(i,j+1,t-1)-us(i,j,t-1) )
     1    +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     2    +us(i,j,t-1)

        endif
c ------------------------------------------------------------------
c Condition 3
       if( i.eq.0 .and. j.ne.rnumpts .and. j.ne.0 )then

us(i,j,t)=s_s1/(2.*xj)*(
     1             (2.*xj+1.)*us(i,j+1,t-1)-4.*xj*us(i,j,t-1)
     2             +(2.*xj-1.)*us(i,j-1,t-1) )
     3        +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+avg_ub )
     4        +us(i,j,t-1)

       endif
c ------------------------------------------------------------------
c Condition 4

       if(i.eq.0 .and. j.eq.rnumpts)then

          us(i,j,t)=s_s1/(2.*xj)*(
     1             (2.*xj+1.)*us(i,j,t-1)-4.*xj*us(i,j,t-1)
     2             +(2.*xj-1.)*us(i,j-1,t-1) )
     3        +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+avg_ub )
     4        +us(i,j,t-1)

       endif
c ------------------------------------------------------------------
c Condition 5
       if(j.ge.1 .and. i.ge.1 .and. i.ne.znumpts
     1                  .and. j.ne.rnumpts)then

          us(i,j,t)=s_s1/(2.*xj)*(
     1             (2.*xj+1.)*us(i,j+1,t-1)-4.*xj*us(i,j,t-1)
     2             +(2.*xj-1.)*us(i,j-1,t-1) )
     3      +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     4      +us(i,j,t-1)

       endif
c ------------------------------------------------------------------
c Condition 6
       if(j.eq.0 .and. i.eq.znumpts)then

          us(i,j,t)=(4.*s_s1)*( us(i,j+1,t-1)-us(i,j,t-1) )
     1        +s_s2*( us(i,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     2        +us(i,j,t-1)

       endif
c ------------------------------------------------------------------
c Condition 7
       if(j.ne.0 .and. i.eq.znumpts .and. j.ne.rnumpts)then

          us(i,j,t)=s_s1/(2.0*xj)*(
     1             (2.*xj+1.)*us(i,j+1,t-1)-4.*xj*us(i,j,t-1)
     2             +(2.*xj-1.)*us(i,j-1,t-1) )
     3       +s_s2*( us(i,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     4       +us(i,j,t-1)
```

```
      endif
c ---------------------------------------------------------------------
c Condition 8
      if(j.eq.rnumpts .and. i.ne.znumpts .and. i.ne.0)then

         us(i,j,t)=s_s1/(2.*xj)*(
     1            (2.*xj+1.)*us(i,j,t-1)-4.*xj*us(i,j,t-1)
     2            +(2.*xj-1.)*us(i,j-1,t-1) )
     3      +s_s2*( us(i+1,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     4      +us(i,j,t-1)

      endif
c ---------------------------------------------------------------------
c Condition 9
      if(j.eq.rnumpts .and. i.eq.znumpts)then

         us(i,j,t)=s_s1/(2.*xj)*(
     1            (2.*xj+1.)*us(i,j,t-1)-4.*xj*us(i,j,t-1)
     2            +(2.*xj-1.)*us(i,j-1,t-1) )
     3       +s_s2*( us(i,j,t-1)-2.*us(i,j,t-1)+us(i-1,j,t-1) )
     4       +us(i,j,t-1)

      endif

c print*,'diffusion us(',i,',',j,',',t,') =',us(i,j,t)

 53       continue
 52      continue

700   continue

c Print the temperatures at each time step in a separate file
c These files take on names in the form T00###.dat where ### is
c the value of t for  0<=t<=399 .

c       filetb = 'TB___.dat'
c  888 do 43 t = 2, strnumpts
c
c        call ripapart(t)
c        filetb(3:3) = char(ch(gh))
c        filetb(4:4) = char(ch(gt))
c        filetb(5:5) = char(ch(gu))
c        print*,'ripapart loop for t=',t
c        open(unit = 22,file=filetb,status='new')
c        write(22,*)zbnumpts,rbnumpts
c
c         do 91 jj=rbnumpts,0,-1
c          write(22,99) (ub(ii,jj,t), ii=0,zbnumpts)
c  91      continue
c        close(22)
c
c  43 continue
c
c       filets = 'TS___.dat'
c  do 44 t = 2, strnumpts

c       call ripapart(t)
c        filets(3:3) = char(ch(gh))
c        filets(4:4) = char(ch(gt))
c        filets(5:5) = char(ch(gu))
c        print*,'ripapart loop for t=',t
c        open(unit = 22,file=filets,status='new')
c        write(22,*)znumpts,rnumpts
c
c         do 92 jj=rnumpts,0,-1
c          write(22,99) (us(ii,jj,t), ii=0,znumpts)
c  92      continue
c        close(22)
```

```
c
c  44 continue

  99    format(11(2x, f7.2))

c The print loop
c To print every XX number of steps use the timestep variable

        timestep = 1

        fileenergy = '______art000.dat'

c        call ripapart(t)
c          fileenergy(10:10) = char(ch(gh))
c          fileenergy(11:11) = char(ch(gt))
c          fileenergy(12:12) = char(ch(gu))

c        fileenergy(1:1) = 'r'
c        fileenergy(2:2) = '_'
c        fileenergy(3:3) = '_'
c       fileenergy(4:4) = '_'
c        fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'

        open(22, file='tem_spec.dat',status='new')
        do 55 t =0,strnumpts,timestep
           do 96 i=0,znumpts-1
     do 57 j=0,rnumpts-1
             write(22,90) z(i,t),r(j,t),us(i,j,t)
 57     continue
 96        continue
 55     continue
        close(22)

        open(22, file='tem_bar.dat',status='new')
        do 62 t =0,strnumpts,timestep
           do 68 i=0,zbnumpts-1
     do 72 j=0,rbnumpts-1
c      write(22,*) (i,j,t)
             write(22,90) zb(i),rb(j),ub(i,j,t)
 72     continue
 68        continue
 62     continue
        close(22)




c        fileenergy(1:1) = 'z'
c        fileenergy(2:2) = '_'
c        fileenergy(3:3) = '_'
c        fileenergy(4:4) = '_'
c        fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c        open(22, file=fileenergy,status='new')
c        do 57 t = 0, strnumpts, timestep
c          write(22,*)'z array at time',t
c              write(22,90) (z(i,t), i=0,znumpts-1)
c 57     continue
c        close(22)

c        fileenergy(1:1) = 'w'
c        fileenergy(2:2) = 'o'
c        fileenergy(3:3) = 'r'
c        fileenergy(4:4) = 'k'
c        fileenergy(5:5) = '_'
c fileenergy(6:6) = 'e'
```

```
c       open(22, file=fileenergy, status='new')
c       do 162 t = 1, strnumpts, timestep
c         write(22,*)'work_in for each element at t =',t
c           do 106 j=rnumpts-1,0,-1
c             write(22,90) (workin_e(i,j,t), i=0,znumpts-1)
c  106      continue
c  162   continue
c       close(22)

c       fileenergy(1:1) = 'r'
c       fileenergy(2:2) = 'p'
c       fileenergy(3:3) = '_'
c       fileenergy(4:4) = '_'
c       fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c       open(22, file=fileenergy,status='new')
c       do 68 t = 0, strnumpts, timestep
c         write(22,*)'rp array at time',t
c           do 111 j=rnumpts,0,-1
c             write(22,90)(rpos(j,t))
c  111      continue
c  68   continue
c       close(22)
c
c       fileenergy(1:1) = 'z'
c       fileenergy(2:2) = 'p'
c       fileenergy(3:3) = '_'
c       fileenergy(4:4) = '_'
c       fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c       open(22, file=fileenergy,status='new')
c       do 72 t = 0, strnumpts, timestep
c         write(22,*)'zp array at time',t
c         write(22,90)(zpos(i,t), i = 0,znumpts)
c  72   continue
c       close(22)
c
c       fileenergy(1:1) = 'v'
c       fileenergy(2:2) = 'z'
c       fileenergy(3:3) = '_'
c       fileenergy(4:4) = '_'
c       fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c       open(22, file=fileenergy,status='new')
c       do 77 t = 0, strnumpts, timestep
c         write(22,*)'vz array at time',t
c             write(22,90) (vz(i,t), i=0,znumpts-1)
c  77   continue
c       close(22)
c
c      fileenergy(1:1) = 'v'
c       fileenergy(2:2) = 'r'
c       fileenergy(3:3) = '_'
c       fileenergy(4:4) = '_'
c       fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c       open(22, file=fileenergy,status='new')
c       do 82 t = 0, strnumpts, timestep
c         write(22,*)'vr array at time',t
c           do 121 j=rnumpts-1,0,-1
c             write(22,90) (vr(j,t))
c  121      continue
c  82   continue
c       close(22)
c
        fileenergy(1:1) = 'i'
        fileenergy(2:2) = 'n'
        fileenergy(3:3) = 't'
        fileenergy(4:4) = 'e'
        fileenergy(5:5) = 'n'
       fileenergy(6:6) = 'g'
        open(22, file=fileenergy,status='new')
        do 97 t = 0, strnumpts, timestep
          write(22,*)'inteng array at time',t
            do 136 j=rnumpts-1,0,-1
               write(22,90) (inteng(i,j,t), i=0,znumpts-1)
  136       continue
  97    continue
        close(22)

        fileenergy(1:1) = 't'
        fileenergy(2:2) = 'e'
        fileenergy(3:3) = 's'
        fileenergy(4:4) = 'p'
        fileenergy(5:5) = 'e'
fileenergy(6:6) = 'c'
        open(22, file=fileenergy,status='new')
        do 102 t = 0, strnumpts, timestep
          write(22,*)'spec temp array at time',t
            do 141 j=rnumpts-1,0,-1
              write(22,90) (us(i,j,t), i=0,znumpts-1)
  141       continue
  102   continue
        close(22)
c
c        fileenergy(1:1) = 'u'
c        fileenergy(2:2) = 's'
c        fileenergy(3:3) = '_'
c        fileenergy(4:4) = '_'
c        fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c        open(22, file=fileenergy,status='new')
c        do 202 t = 0, strnumpts, timestep
c          write(22,*)'Specimen temperature array at time',t
c            do 241 j=rnumpts-1,0,-1
c              write(22,90) (us(i,j,t), i=0,znumpts-1)
c  241       continue
c  202   continue
c        close(22)
c
c        fileenergy(1:1) = 'u'
c        fileenergy(2:2) = 'b'
c        fileenergy(3:3) = '_'
c        fileenergy(4:4) = '_'
c        fileenergy(5:5) = '_'
c fileenergy(6:6) = '_'
c        open(22, file=fileenergy,status='new')
c        do 93 t = 0, strnumpts, timestep
c          write(22,*)'Incident bar temperature array at time',t
c            do 131 jb=rbnumpts,0,-1
c              write(22,90) (ub(ib,jb,t), ib=0,zbnumpts)
c  131       continue
c   93   continue
c        close(22)
c
c
       fileenergy(1:1) = 'w'
        fileenergy(2:2) = 'o'
        fileenergy(3:3) = 'r'
        fileenergy(4:4) = 'k'
        fileenergy(5:5) = 'i'
fileenergy(6:6) = 'n'
        open(22, file=fileenergy,status='new')
        do 107 t = 0, strnumpts, timestep
          write(22,*)'workin array at time',t
          write(22,90) (workin(t))
  107   continue
        close(22)
c
  113   stop
  95 format(10(2x,f6.3))
  90    format(5(2x, e20.12))
  999   format(5(2x,f7.1))
        end
* -------------- functions and subroutines follow ---------

       FUNCTION f(x)
  double precision con1,con2,con3,con4,con5,con6
  double precision f, x
        COMMON/coef/con1,con2,con3,con4,con5,con6
c   print*,'in the f function routine x=',x

          f = con1*x+con2*x**2+con3*(1/x)+con4*x**3
     1 +con5*x**.5-con6

c   print*,'at x=',x,'the value of f is',f
          RETURN
       END

function dfdx(x)
  double precision con1,con2,con3,con4,con5
  double precision x, dfdx
  common/coef/con1,con2,con3,con4,con5
c   print*,'in dfdx function routine x=',x
  dfdx = con1+2*con2*x-con3*(1/x**2)+
     1  3*con4*x**2+.5*con5*(1/x**.5)
c   print*,'con1 =',con1
c   print*,'con2 =',con2
c   print*,'con3 =',con3
c   print*,'con4 =',con4
c   print*,'con5 =',con5
c   print*,'at x=',x,'the value of dfdx is',dfdx
  return
end

        subroutine ripapart(m)
c BIG ASSUMPTION: m is an integer in the range 0..999
        integer m
        common ih, it, iu
c
c CAUTION: INTEGER mode arithmetic
c
        ih = m / 100
        it = (m - ih * 100) / 10
        iu = m - ih * 100 - it * 10
        return
        end

        subroutine spline(k,x,y,ac,bc,cc,dc)

        integer k
        parameter(n = 3)

        double precision  x(0:n+1), y(0:n+1)
        double precision  deltax(0:n), deltay(0:n)
        double precision  s(0:n+1), a(2:n), b(1:n), c(1:n-1)
        double precision  u(1:n), gam(2:n)
        double precision  ac(0:n), bc(0:n), cc(0:n), dc(0:n)
        double precision  bet

c       data (x(i), i=0, n+1) / 1.0,2.25,3.5,4.0,5.25,6.75,8.0 /
c       data (y(i), i=0, n+1) / 1.0,2.75,4.25,6.0,4.5,1.75,2.0 /

        do 70 i = 0,n
        deltax(i) = x(i+1)-x(i)
        deltay(i) = y(i+1)-y(i)
c         print*, 'deltax(',i,') =',deltax(i)
c         print*, 'deltay(',i,') =',deltay(i)
  70    continue

        do 75 i = 1,n
         b(i)=2.0*( deltax(i-1)+deltax(i) )
c         print*,'b(',i,') =',b(i)
         u(i) = 6.0*( deltay(i)/deltax(i) - deltay(i-1)/deltax(i-1) )
c         print*,'u(',i,') =',u(i)
  75    continue

        do 80 i = 2, n
         a(i) = deltax(i-1)
c         print*,'a(',i,') =',a(i)
  80    continue

        do 85 i = 1, n-1
         c(i) = deltax(i)
c         print*,'c(',i,') =',c(i)
  85    continue

        if (b(1).eq.0) pause
        bet = b(1)
        s(1) = u(1)/bet

        do 95 i  = 2, n
         gam(i) = c(i-1)/bet
c         print*,'gam(',i,') =',gam(i)
         bet = b(i) - a(i)*gam(i)
         if (b(i) .eq. 0) pause
         s(i) = ( u(i) - a(i)*s(i-1) )/bet
c         print*,'s(',i,') =',s(i)
  95    continue

        do 100 i = n - 1, 1, -1
         s(i) = s(i) - gam(i+1)*s(i+1)
c         print*, s(i)
 100     continue

        s(0)   = 0.0
        s(n+1) = 0.0

        do 105 i = 0, n
         ac(i) = y(i)
                bc(i) = ( y(i+1) - y(i) )/deltax(i)
     1               - (1.0/6.0)*( s(i+1) + 2.0*s(i) )*deltax(i)
         cc(i) = s(i)/2.0
         dc(i) = ( s(i+1) - s(i) )/(6.0*deltax(i))
 105    continue

        do 110 i = 0, n
c         print*, 'ac(',i,') =', ac(i)
c         print*, 'bc(',i,') =', bc(i)
c         print*, 'cc(',i,') =', cc(i)
c         print*, 'dc(',i,') =', dc(i)

 110    continue
        return
        end
* ----------------------------------------------------------

      SUBROUTINE NEWTON(X,EPS,IMAX,DXMAX,MULT,ROOT,F,DFDX,N)
*
*  Newton's algorithm , Xnew = Xold + dx, where
*
*                  dx = -f(Xold)/dfdx(Xold)
```

```
*
*  is used to find a root of the function f(x). Additionally,
*  the derivative of the function, called dfdx() is required
*  as a used coded function.  The initial guess for the root
*  is X and convergence is attained when |dx| < eps.
*------------------------------------------------------------
* Variables
*
double precision X, ROOT, F, DFDX, DF, DX
real eps, dxmax
       INTEGER N, IMAX, MULT
c print*,' I am here in the newton subroutine'
c print*,'x=',x
c print*,'eps=',eps
c print*,'dxmax=',dxmax
c print*,'imax=',imax
c print*,'mult=',mult

*     IMAX  -- The maximum no. of iterations.
*     DXMAX -- A limit on the size of computed dx's
*     N     -- The actual number of iterations
*     MULT  -- The assumed multiplicity of the root.
*              (Usually mult = 1)
*     ROOT  -- The value computed for the root of f(x)
*------------------------------------------------------------
* Iterations
*
      DO 15, I = 1,IMAX
c   print*,'i=',i,'x=',x
         DF = DFDX(X)
c         print*,'at i=',i,'the value of dfdx at x=',x,'is',df
  IF(DF .EQ. 0.)THEN
             WRITE(*,12)I,X,F(X)
             PAUSE 'Fatal Error in NEWTON'
             RETURN
         ENDIF
         DX = -MULT*F(X)/DFDX(X)
c    print*,'at x=',x,'dx =',dx
         IF(ABS(DX) .LT. EPS)THEN
             ROOT = X + DX
             N    = I
             RETURN
         ELSEIF(ABS(DX) .GT. DXMAX)THEN
             WRITE(*,10)I,DX
             PAUSE 'Fatal Error in NEWTON, method diverging'
             RETURN
         ENDIF
         X = X + DX
c   print*,'the new x+dx becomes ',x
 15  CONTINUE
      WRITE(*,11)X,F(X),DX
      PAUSE 'ERROR in NEWTON, method not converging'
*------------------------------------------------------------
* Formats
*
 10   FORMAT(//,
     + T5,'=================ERROR IN NEWTON==============',/,
     + T5,'|  In iteration No.-',I3,'                  |',/,
     + T5,'|  The current value of dx = ',E8.1,'       |',/,
     + T5,'|  is larger than the prescribed limit      |',/,
     + T5,'===============Program Terminated============',/)
*
 11   FORMAT(//,
     + T5,'=================ERROR IN NEWTON==============',/,
     + T5,'| Newton fails due to excessive iterations   |',/,
     + T5,'| After executing the maximum number of steps |',/,
     + T5,'|   f(',E12.5,   ') = ',E8.1,'              |',/,
     + T5,'| and the latest dx = ',E8.1,'              |',/,
     + T5,'================Program Terminated=============',/)
*
 12   FORMAT(//,
     + T5,'=================ERROR IN NEWTON===============',/,
     + T5,'|  In iteration No.-',I3,'                 |',/,
     + T5,'|  The current value of dF is EXACTLY zero    |',/,
     + T5,'|  The problem is likely in the code for DFDX |',/,
     + T5,'================Program Terminated=============',/)
      END
```