12-1991

Investigations of the Mechanical Behavior of Graphite/PEEK (APC-2) Composits

Kyun Lee
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Kyun Lee entitled "Investigations of the Mechanical Behavior of Graphite/PEEK (APC-2) Composits." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Engineering Science.

Y. Jack Weitsman, Major Professor

We have read this dissertation and recommend its acceptance:

John Stoneking, John Landis, Raymond Krieg, Gerald Fitzpatrick

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Kyun Lee entitled "Investigations of the Mechanical Behavior of Graphite/PEEK (APC-2) Composites." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Engineering Science.

Dr. Y. Jack Weitsman , Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

INVESTIGATIONS OF THE MECHANICAL BEHAVIOR OF GRAPHITE/PEEK (APC-2) COMPOSITES

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Kyun Lee
December 1991

DEDICATION

I dedicate this dissertation to my mother.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to the my advisory committee, Dr. Stoneking, Dr. Fitzpatrick, Dr. Krieg, Dr. Landes and especially to my chairman, Dr. Weitsman for their invaluable guidance and assistance in preparing the dissertation.

I also thank Alice Beauchene and Loretta Haack of The University of Tennessee Computing Center for their helpful technical advice in performing the computer works.

I wish to express deep appreciation to my wife, Hae-Ok, to my mother and to my son, Jung-Woo for their understanding, patience and support.

This research was sponsored by the office of Naval Research. The Part I of this dissertation was performed under Contract N00014-90-J-1556 and the Part II was conducted under Contract N00014-82-K-0562. Special thanks to the program manager, Dr. Y. Rajapakse in the Mechanics Division, Engineering Sciences Directorate, ONR. The support of ONR is gratefully acknowledged.

ABSTRACT

The purpose of this research is to investigate the effects of two phenomena that are specific to thermoplastic resin composites. These phenomena are:

(a) the significant nonlinear creep, that occurs especially at high temperatures, which affects the residual thermal stresses in a geometrically constrained structure.

(b) the spatially nonuniform transverse crystallinity which develops in some fiber-reinforced thermoplastic composites, which introduces spatial variability and nonlinearity in the stiffness of the composite.

The first part of the dissertation is related to the former phenomenon. It presents a computational scheme for the the optimal temperature path for *APC-2* composite laminate, modeled as nonlinear, thermorheologically complex, viscoelastic material. The laminate is assumed to be sufficiently thin to obviate the accounting for temperature nonuniformities across the thickness. An iterative, computational scheme is developed for the particular case of a cross-ply lay-ups laminate, which can be generalized to other symmetric lay-up with minor modifications. As prototype cases which can provide the guidance, as well as verification checks, for the iterative numerical scheme, analytic solutions are developed for two idealized relaxation models.

The second part of the dissertation, which concerns the latter phenomenon, presents analytical/computational results for the inplane stresses in a moderately thick, cross-ply, graphite/PEEK (*APC-2*) laminate accounting for effects of nonuniform PEEK crystallinity across the plate's thickness. These effects are incorporated as crystallinity-induced variations in the longitudinal modulus E_L, including its nonlinear dependence on strain.

Contents

List of Figures

List of Tables

Part I

Residual Thermal Stresses in Graphite/PEEK (APC-2) Laminates

Chapter 1

General

1.1 Introduction

Structural composite systems, with large volume fractions of continuous fiber reinforcement, are being increasingly utilized in aerospace structures, as well as industrial applications, due to their high stiffness and strength to weight ratios. In the past, crosslinked thermoset resin systems such as epoxy resins were primarily used as the matrix in which glass, boron or carbon fibers have been embedded. This was done in spite of the well-established brittleness of those resins. More recently, thermoplastic polymers such as Polyetheretherketone (PEEK) have been introduced as alternatives to epoxy materials for binding fiber reinforced composites. Thermoplastic resins possess improved features such as increased fracture toughness, stronger resistance to the crack propagation, and higher imperviousness to attack by hostile environments. Thermoplastic resin composites are gaining growing acceptance in applications for high performance engineering structures, not only as a result of improvements in matrix dominated mechanical response but also because of their potential for cheaper and more efficient mass production techniques. In addition, thermoplastics have the important advantage in their potential for recycling, which is becoming an increasingly important environmental issue. The translation to thermoplastic resin system from thermoset matrix system is mostly made by Aromatic Polymer Composites (APC), which are based on 60% by volume of continuous , collimated, high strength, carbon fibers in a matrix of polyetheretherketone (PEEK).

The current study is concerned with carbon/PEEK composite, also denoted as *APC-2*. PEEK is a semicrystalline polymer and, like most thermoplastic resins, it exhibits substantial time-dependent stress-strain response,especially at high temperatures [1]. In particular, since graphite fibers possess a null coefficient of the thermal expansion , the shrinkage of the relatively soft PEEK resin is severely inhibited during the post-cure cool-down stage. Furthermore, the processing of *APC-2* involves cool-down from $T_M \approx 400°C$ to $T_R \approx 20°C$ – which is about twice the temperature excursion sustained by graphite/epoxy composites. The above factors produce significant residual thermal stresses in multi-directionally reinforced laminates, which exceed the range of linear behavior. In view of the time-dependent response of PEEK, these stresses are sensitive to temperature history. Consequently, it is desirable to determine an optimal temperature path, which minimizes the residual thermal stresses.

It seems that the most complete characterization to date of the time-dependent response of PEEK and unidirectionally reinforced *APC-2* coupons was obtained by Xinran [1] [2]. A substantial amount of experimental creep data and other supplementary experimental results enabled Xinran to represent the time-dependent response of PEEK and *APC-2* by means of the nonlinear viscoelastic model of Schapery [2]. However, it is important to note that creep data in Reference [1] were collected under isothermal conditions and are limited to the temperature range of $20^0C < T < 200^0C$. However, it was shown in Reference [3] that the full implementation of Schapery's model requires additional creep data under transient temperature conditions. Consequently, Xinran's characterization contains an uncertain component. This issue is clarified in Sections 1.2.3 and 1.2.4 of the current chapter.

[1] Note that the properties of unreinforced PEEK do not carry over to fiber reinforced layers since in the presence of fibers the semicrystalline microstructure of PEEK changes to a transversely crystalline array. Consequently, it is incorrect to employ micromechanics concepts to predict the properties of *APC-2* from bulk properties of unreinforced PEEK.

[2] Numbers in brackets indicate Reference listed at the end of this work. The reference list is representative only and is not intended to be comprehensive.

The time-dependent response of *APC-2* laminates can be predicted from the behavior of a single ply by means of classical laminate theory [4].

When linearly viscoelastic materials undergo a prescribed geometrically constrained temperature drop over a finite, predetermined, time interval it is possible to find an optimal cool-down path $T_\Omega(t)$ which results in minimal residual stresses. Such paths were found for either thermorheologically simple [5] - [7] or complex [8], linear viscoelastic responses. However, those results do not apply to *APC-2* composites because of the substantial nonlinearity which occurs during their cool-down. Consequently, the extension of the previous optimization schemes to circumstances which account for nonlinear behavior forms the subject of this part of the dissertation.

This part of the dissertation presents a computational scheme for the the optimal temperature path for *APC-2* composite laminate, modeled as nonlinear, thermorheologically complex, viscoelastic material. The laminate is assumed to be sufficiently thin to obviate the accounting for temperature nonuniformities across the thickness. An iterative numerical scheme is developed for the particular case of a cross-ply laminate, but can be generalized to other symmetric lay-up with minor modifications. As prototype cases, analytic solutions are developed for two idealized relaxation models. These cases provide guidance, as well as verification checks, for the iterative numerical scheme. Some computational results for the *APC-2* are presented and discussed for those two specific cases.

The brief outline of the Part I is as follows. Analytic solutions, developed for the optimal cool-down in the linear case, are briefly reviewed in Chapter 2. Chapter 3 presents analytic solutions and numerical iterative schemes for the optimal cool-down paths which correspond to two idealized nonlinear viscoelastic responses. Also, comparisons are provided between the analytical results and the results of the iterative schemes. Finally, the nonlinear iterative scheme was applied to the *APC-2*, employing the nonlinear viscoelastic data available for this material. This scheme, together with computational results, is presented and discussed in Chapter 4. Also, computer code developed for the computations is listed in Appendix A.

The remainder of this chapter provides some background information related to the current work. This includes an overview of viscoelastic constitutive equations in Section 2, including Schapery's nonlinear viscoelastic constitutive model. Subsequently, some discussion is provided concerning the experimental viscoelastic characterization schemes associated with Schapery's model and thermorheologically complex response. Section 3 presents the experimental results [1] which were adapted to the computations in this study.

1.2 Viscoelastic Constitutive Equations

1.2.1 General

The term "viscoelastic response" refers to a material behavior which combines both the viscous and elastic phenomena. Many materials exhibit a characteristic viscoelastic behavior. Prominent among them are polymers, especially at elevated temperatures and/or in the presense of moisture.

The stress-strain relations of viscoelastic materials may be either linear or nonlinear. A behavior is defined to be linear if, and only if, it satisfies the following two properties: *homogeneity* and *superposition*. The first property states that a proportional change in input causes the same proportional change in response (e.g., if the input is doubled, the response doubles). The second property states that the response due to the simultaneous or sequential application of multiple inputs is equal to the sum of the responses due to each input acting separately. When the mechanical behavior of a material is linear, as defined above, and when strains are sufficiently small to preclude geometrical nonlinearity, the mathematical theory of viscoelasticity yields explicit single integral expressions relating any viscoelastic response to any input.

Several mathematical formalisms were employed to express nonlinear viscoelastic response. In most cases these formalisms employ multiple integral representations, with multiple-parameter kernels, to represent the nonlinear behavior. In addition to the excessive computational work associated with multiple integrations, the main drawback in the

above representation is due to the impractical amount of experiments that are necessary to characterize the multi-parameter kernels. In contrast, Schapery's nonlinear viscoelasticity model, which is based upon fundamental concepts of continuum mechanics and irreversible thermodynamics, employs a single integal representation, which is easier to compute, and involves characteristic nonlinear material functions which can be characterized experimentally by means of a manageable test program.

One purpose of this chapter is to give a brief overview of viscoelastic constitutive modeling. For simplicity and clarity, the next section is concerned with one-dimensional linear and nonlinear viscoelastic constitutive models.

1.2.2 Linear Constitutive Equation

One dimensional response

Consider a one-dimensional quasi-static case in which the stress and strain depend on a single material coordinate X and on time t. If the material is linear-elastic solid, we have

$$\sigma(X,t) = E\epsilon(X,t) \tag{1.1}$$

in which E is Young's modulus. Similarly, if the material is linear-viscous liquid, the constitutive equation which relates shear stress and shear strain rate reads

$$\tau(X,t) = \mu\frac{\partial\gamma(X,t)}{\partial t} \tag{1.2}$$

where μ denotes viscosity. Thus a viscous fluid "remembers" the past history of deformation only to a extent which can be expressed in terms of the current strain rate. The two simple cases described in Equations 1.1 and 1.2 provide the basis for the classical mathematical theories of viscoelasticity, and also suggest that viscoelastic materials remember in a more general manner previous deformations to which they have been subjected. Thus it can be postulated that the response of viscoelastic materials, $R(X,t)$, depends on the input $I(X,\tau)$ at all time τ up to and including the current time t. Formally

$$R(X,t) = F[I(X,\tau)], \quad -\infty < \tau < t \tag{1.3}$$

where F is a functional of I. The brackets [ ] are used to denote the fact that the current value of R depends on the history of I instead of just its current value. Equation 1.3 is the starting point for the mathematical theory of one-dimensional viscoelasticity.

Hereditary integrals

In the case of linear viscoelasticity [17], the functional F is expressible in the form of superposition integral such that

$$R(t) = \int_{-\infty}^{t} \phi(t,\tau)\frac{dI(\tau)}{d\tau}d\tau \tag{1.4}$$

The type of relation connecting input and response in Equation 1.4 is called a hereditary integral or hereditary law. Equation 1.4 allows for the possiblity of material *aging* since the response R to a given input I depends separately upon both t, the time of measurement, and τ, the time of input. If, instead of t and τ separately, R depends on elapsed time, $t-\tau$, since application of the input I, the viscoelastic behavior is called *non-aging*. For a non-aging material, the hereditary integral Equation 1.4 becomes

$$R(t) = \int_{-\infty}^{t} \psi(t-\tau)\frac{dI(\tau)}{d\tau}d\tau \tag{1.5}$$

In the sequel non-aging material behavior will be assumed.

One-dimensional viscoelastic response functions

The linear isothermal viscoelastic constitutive relations relating current response, stress $\sigma(t)$, and intput history of strain $\epsilon(\tau)$, for a nonaging material follow immediately from Equation 1.5, namely

$$\sigma(t) = \int_{-\infty}^{t} E(t-\tau)\frac{d\epsilon(\tau)}{d\tau}d\tau \tag{1.6}$$

The function E is known as the relaxation modulus function.

Equivalently, the current strain can also be related to the past history of stress. The explicit linear form of this relation for a non-aging viscoelastic material is

$$\epsilon(t) = \int_{-\infty}^{t} D(t-\tau)\frac{d\sigma(\tau)}{d\tau}d\tau \tag{1.7}$$

The function D is known as the creep compliance of the material. The relaxation modulus or creep compliance must be obtained from experiment or from the physics of the material structure.

The linear one-dimensional constitutive relation (Eq. 1.6) can be rewritten in an alternate form as

$$\sigma(t) = E_\infty \epsilon(t) + \int_{-\infty}^{t} \Delta E(t-\tau)\frac{d\epsilon(\tau)}{d\tau}d\tau \tag{1.8}$$

where $E_\infty = E(\infty)$, and

$$\Delta E(t) = E(t) - E_\infty \tag{1.9}$$

denotes the transient component of the stress-relaxation function. An analogous alternate expression can be written in place of Equation 1.7.

1.2.3 Nonlinear Consitutive Equation

A formulation which extends Equation 1.8 to nonlinear response has been developed by Schapery [9] on the basis of considerations of the thermodynamics of irreversible processes ([10] - [12]). Introducing special forms of entropy production and the Gibbs free energy, he derived the following consititutive equation

$$\sigma(t) = v_\infty E_\infty \epsilon(t) + v_1 \int_{-\infty}^{t} \Delta E(\rho - \rho')\frac{\partial}{\partial \tau}[v_2 \epsilon(\tau)]d\tau \tag{1.10}$$

where the reduced time ρ is defined by

$$\rho \equiv \rho(t) = \int_{t_0}^{t} \frac{dt'}{a_\epsilon[\epsilon(t')]}, \quad a_\epsilon > 0 \tag{1.11}$$

where t_0 is an arbitrary time prior to the application of strain and $\rho' = \rho(\tau)$. In Equation 1.10 v_∞ and v_1 are functions of $\epsilon(t)$, while v_2 and a_ϵ are functions of $\epsilon(\tau)$ during

$-\infty < \tau < t$. The linear form Equation 1.6 is recovered when $v_\infty = v_1 = v_2 = a_\epsilon = 1$, which are the limiting values of these functions as strains become sufficiently small.

Schapery [2] provided the stress formulation analog of Equation 1.10, which reads as follows:

$$\epsilon(t) = g_0 D_0 \sigma(t) + g_1 \int_{-\infty}^{t} \Delta D(\xi - \xi') \frac{\partial}{\partial \tau}[g_2 \sigma(\tau)] d\tau \tag{1.12}$$

where the reduced time ξ is defined by

$$\xi \equiv \xi(t) = \int_{t_0}^{t} \frac{dt'}{a_\sigma[\sigma(t')]}, \quad a_\sigma > 0 \tag{1.13}$$

Also, t_0 is an arbitrary time prior to the first application of stress and $\xi' = \xi(\tau)$. In Equation 1.12 D_0 and ΔD are given by

$$D_0 = D(0), \quad \Delta D(t) = D(t) - D_0 \tag{1.14}$$

where $D(t)$ is the creep function for stress levels which are sufficiently small to assume linear behavior, g_0 and g_1 are functions of $\sigma(t)$, and g_2 and a_σ are functions of $\sigma(\tau)$. The functions g_0, g_1, g_2 and a_σ tend to unity as stresses tend to zero, in which case Equation 1.12 reduces to the linear form given in Equation 1.7.

Thermoviscoelasticity The thermoviscoelastic behavior of polymeric materials has been characterized by assuming either a thermorheologically simple or a thermorheologically complex response. In thermorheologically simple material (TSM) elevated temperatures reduce the viscosity of the polymeric material, resulting in an acceleration of the creep process. Experimentally this phenomenon is observed by rigid horizontal shifts of isothermal data parallel to the $\log t$ scale. However, in thermorheologically complex material (TCM) modulus softening as well as the reduction of the viscosity complicate the creep process so that both the vertical and horizontal shifts of isothermal data are required parallel to both the $\log t$ and, say, the $\log D$ axes.

It is instructive to note that both TSM and TCM responses can be obtained from Schapery's nonlinear viscoelastic model by considering that g_0, g_1, g_2, and the reduced time γ are exclusively functions of temperature. In this case Equation 1.12 becomes the general linear nonisothermal constitutive equation,

$$\epsilon(t) = g_0(T)D_0\sigma(t) + g_1(T)\int_{-\infty}^{t} \Delta D(\gamma - \gamma')\frac{\partial}{\partial\tau}[g_2(T(\tau))\sigma(\tau)]d\tau, \qquad (1.15)$$

involving temperature-dependent material properties. In Equation 1.15 the reduced time γ is defined by

$$\gamma \equiv \gamma(t) = \int_{t_0}^{t} \frac{dt'}{a_T(T)}, \quad a_T > 0 \qquad (1.16)$$

Materials for which a smooth master compliance curve may be obtained by horizontal and vertical shiftings of isothermal creep data will be referred to as a thermorheologically complex material (TCM). If all curves are shifted towards the reference curve [3], which is obtained at certain reference temperature T_{ref}, the amounts of vertical and horizontal shift along two coordinate directions determine $\log g_1 g_2$ and $\log a_T$ respectively.

When $g_0 = g_1 = g_2 = 1$, Equation 1.15 reduces to the linear nonisothermal constitutive equation for a thermorheologically simple material (TSM), for which a master compliance curve may be obtained by horizontal shifting parallel to the $\log t$ axis alone. In the context of Schapery's model, nonlinear thermoviscoelastic response is expressed by g_0, g_1, and g_2 as well as a_T, which depend on both T and σ. However, in many circumstances it suffices to consider only $a_T = a_T(T, \sigma)$ to account for nonlinearity. In particular this is the case for *APC-2* material.

1.2.4 A Simple Illustration of Thermorheologically Simple and Complex Responses

A simple explanation of the essential features of thermorheologically simple and complex responses is provided by considering the creep of a Kelvin model subjected to a unit step stress input, shown in Firgure 1.1.

[3] The temperature dependent parameters are *normalized* at a certain reference temperature T_{ref}, namely $g_1(T_{ref}) = g_2(T_{ref}) = g_0(T_{ref}) = a_T(T_{ref}) = 1$.

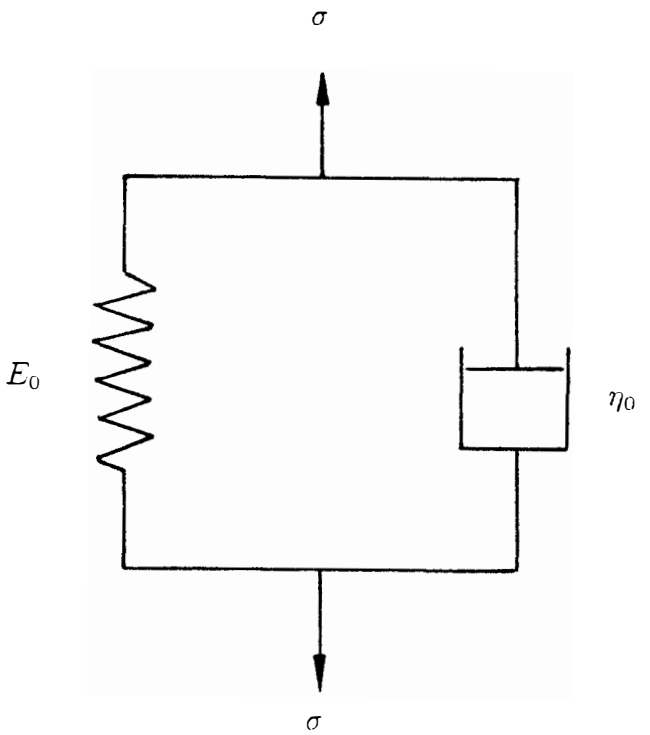

Figure 1.1: The Kelvin Model

Basic considerations give

$$\epsilon(t) = \frac{1 - \epsilon^{-(E_0/\eta_0)t}}{E_0} = D(t) \tag{1.17}$$

where $D(t)$ denotes creep compliance.

A schematic plot of $\log D(t)$ vs. $\log t$ is shown in Figure 1.2.

Consider now temperature effects. If all temperature effects are confined to the viscous element, whereby reducing the viscosity η_0 to a value $\eta(T) = a(T)\eta_0$ (where $a(T)$ decreases with rising temperature) then $D(t)$ in Equation 1.17 becomes

$$D(t;T) = \frac{1}{E_0}\left[1 - \exp\left(-(E_0/\eta_0)t/a(T)\right)\right] \tag{1.18}$$

Therefore, the plot $\log D(t;T)$ vs. $\log t$ will appear the same as $logD(t)$ vs. $logt$ except for a horizontal shift parallel to the $\log t$ axis by an amount of $\log a(T)$. The shifted plot is shown by the dashed line in Figure 1.2 (b).

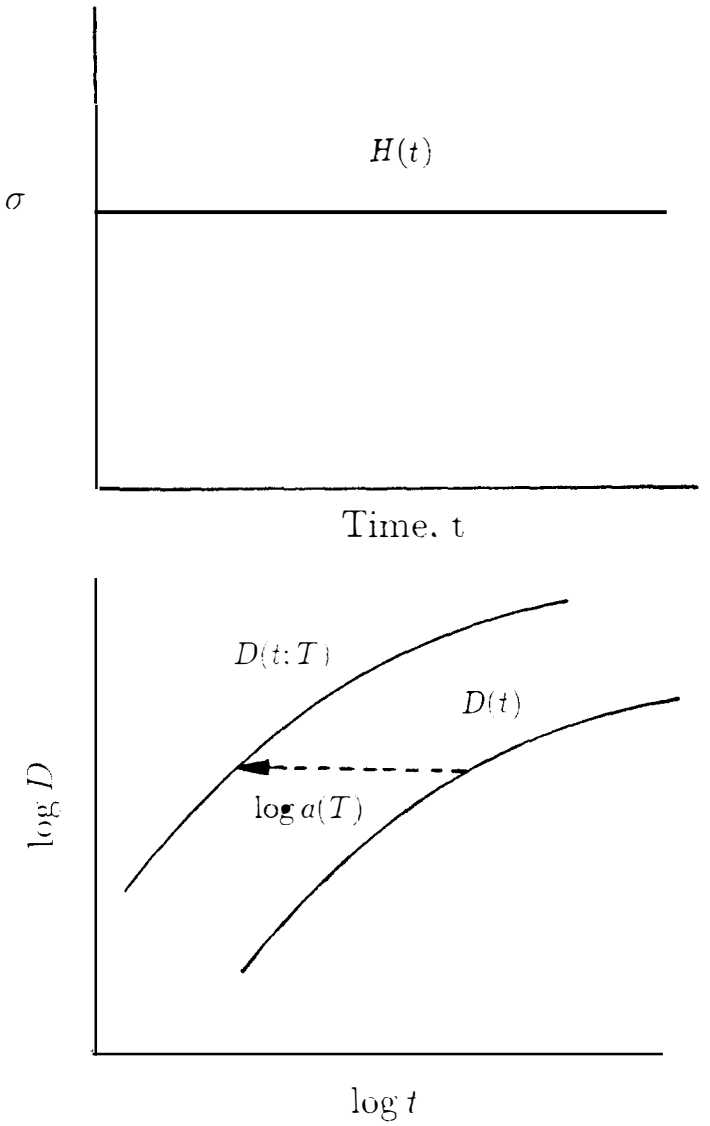

Figure 1.2: Unit step stress input and the creep compliance $D(t;T)$ for a thermorheologically simple response. Creep compliances at distinct temperatures are related through a horizontal shift function only.

In such circumstances the viscoelastic response is said to be thermorheologically simple.

If temperature affects both spring and dash-pot elements of the Kelvin model, namely when $E = E(T) = E_0 v(T)$ and $\eta(T) = \eta_0 a(T)$, the creep compliance in Equation 1.18 reads

$$\begin{aligned} D(t;T) &= \frac{1}{E_0 v(T)} \left[1 - \exp\left(-\frac{E_0}{\eta_0}\frac{v(T)}{a(T)}t\right)\right] \\ &= \frac{1}{E_0 v(T)} \left[1 - \exp\left(-\frac{E_0}{\eta_0}\frac{t}{\tilde{a}(T)}\right)\right] \end{aligned} \tag{1.19}$$

In this case the plot of $\log D(t;T)$ vs. $\log t$ will appear the same as $\log D(t)$ vs. $\log t$ except for a horizontal shift by an amount of $\log \tilde{a}(T)$ parallel to the $\log t$ axis and a vertical shift by an amount of $\log v(T)$ parallel to the $\log D$ axis (see Figure 1.3).

In such circumstances the viscoelastic response is said to be thermorheologically complex.

1.2.5 Creep and Recovery tests

In a creep test, for which $\sigma(t) = \sigma_0 H(t)$, with $H(t)$ denoting the Heaviside unit step function Equation 1.12 simplifies to

$$\epsilon_c(\sigma_0, t) = g_0^0 D_0 \sigma_0 + g_1^0 g_2^0 \sigma_0 \Delta D(t/a_\sigma^0) \tag{1.20}$$

where the superscript denotes that σ_0 is the argument of the specified function, for example, $g_0^0 = g_0(\sigma_0)$. By definition, Equation 1.20 shows that the nonlinear creep compliance D_n is given by

$$D_n^0 \equiv \epsilon_c(\sigma_0, t)/\sigma_0 = g_0^0 D_0 + g_1^0 g_2^0 \Delta D(t/a_\sigma^0) \tag{1.21}$$

which depends on σ_0 in the nonlinear range.

Schapery [9] proposed to determine the material functions by the following graphical method. From Equation 1.21, we have

$$\log(D_n^0 - g_0^0 D_0) = \log g_1^0 g_2^0 + \log \Delta D(t/a_\sigma^0). \tag{1.22}$$

Hence, plots of $\log(D_n^0 - g_0^0 D_0)$ against $\log t$ for different stress levels σ_0 should yield a set of curves which can be superposed by suitable translations in the coordinate directions. If

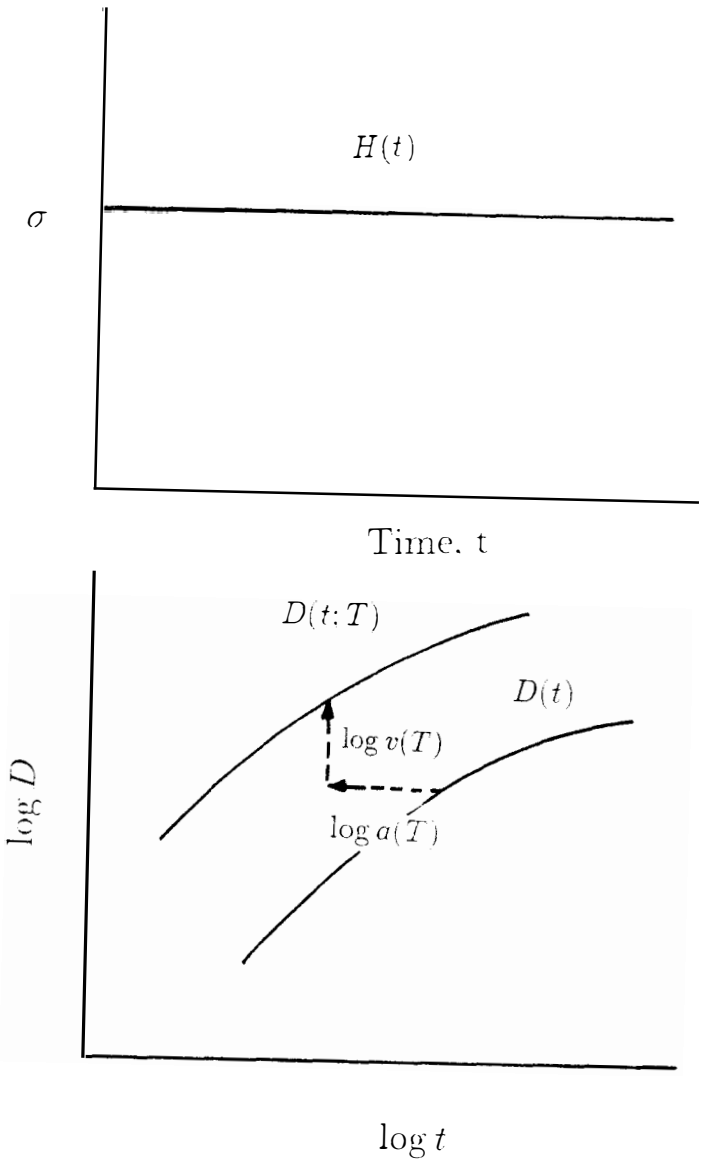

Figure 1.3: Unit step stress input and the creep compliance $D(t;T)$ for a thermorheologically complex response. Creep compliances at distinct temperatures are related through both horizontal and vertical shift functions.

all curves are shifted towards the creep curve which applies in the linear range (which, strictly speaking, is valid for $\sigma_0 = 0$) so as to form a smooth and continuous "master curve", then the amounts of these shifts are $\log g_1^0 g_2^0$ and $\log a_\sigma^0$, respectively, which enables the determination of $g_1^0 g_2^0$ and a_σ^0. To determine the individual functions g_1^0 and g_2^0 it is necessary to conduct creep and recovery tests.

Peretz and Weitsman ([3],[13]) characterized the nonlinear viscoelastic response of a technical adhesive by means of creep and recovery tests employing the nonlinear viscoelastic constitutive model of Schapery. The data base for a material characterization required the imposition of transient temperature conditions to separate nonlinear functions. The characterization scheme is briefly outlined below.

Assuming a power law creep response, the Schapery's nonlinear viscoelastic constitutive equation (Eq. 1.12) becomes

$$\epsilon = g_0 A_0 \sigma + C g_1 \int_0^t (\xi(t) - \xi(t'))^n \frac{d(g_2 \sigma(t'))}{dt'} dt' + \alpha \Delta T \tag{1.23}$$

For the special purpose of separation of g_1 and g_2, the following load and temperature histories were considered:

$$\begin{array}{rcll} \sigma & = & \sigma_0[H(t) - H(t-t_1)], & \sigma_0/\sigma_{ult} \ll 1 \\ T & = & T_2 - (T_2 - T_1)H(t-t_1), & T_2 > T_1 \end{array} \tag{1.24}$$

During the loading stage Equation 1.23, therefore, reads

$$\frac{1}{\sigma_0}(\epsilon_t - \alpha(T_2 - T_R)) = g_0(T_2)A_0 + C g_1(T_2) g_2(T_2) \left(\frac{t}{a(T_2)}\right)^n, \quad 0 < t < t_1 \tag{1.25}$$

while, after unloading, we have

$$\frac{1}{\sigma_0}(\epsilon_t - \alpha(T_1 - T_R)) = C g_1(T_1) g_2(T_2) \left[\left(\frac{t_1}{a(T_2)} + \frac{t-t_1}{a(T_1)}\right)^n - \left(\frac{t-t_1}{a(T_1)}\right)^n\right], \quad t > t_1 \tag{1.26}$$

The primary significance of Equations 1.25 and 1.26 lies in the fact that they relate the recorded creep data to distinct values of $g_1(T_1)$ and $g_1(T_2)$, whereby providing a way to determine $g_1(T)$. More complete details are given in the works by Peretz and Weitsman ([3],[13]).

1.3 Characterization of *APC-2*

1.3.1 General

Xinran [1] characterized the viscoelastic response of the *APC-2* composites in the context of Schapery's nonlinear viscoelastic constitutive model. The linear thermoviscoelastic response was investigated by conducting isothermal creep tests at various levels of temperature under sufficiently low stress levels. Creep response, at reference temperature T_R and within linear range, was modeled by a general power law, with the attendant material constants determined by curve fitting techniques. Nonlinear effects were determined by conducting isothermal creep tests at various levels of stress and temperature. Master compliance curves were constructed from the experimental data by horizontal and vertical shifts using a graphical method. The nonlinear parameter a_σ was obtained by numerical fitting of the creep data.

1.3.2 Experimental Procedure

Specimens

As recommended by the ICI corporation, *APC-2* laminates were fabricated in an autoclave. The prepreg roll was 14 cm wide with nominal thickness of 0.125 mm. All specimens consisted of uni-directionally reinforced flat coupons. These included 90^0 and 15^0 (off-axis) coupons of nominal dimensions of $2.1 \times 14 \times 130$ mm and $1.6 \times 14 \times 340$ mm, respectively. The 90^0 coupons were cut from a 16 ply panel of dimensions of $2.1 \times 130 \times 3400$ mm, and off-axis coupons from a 12 ply panel of dimensions of $1.6 \times 2600 \times 3400$ mm.

Creep-recovery tests

The thermoviscoelastic characterization was cast in the context of an assumed power-law creep response and Schapery's nonlinear viscoelastic constitutive model. Therefore, it was necessary to determine the constants which prescribe the general power law as well as the

stress and temperature dependence of the functions, g_0, g_1, g_2 and $a_{\sigma T}$, listed in Equation 1.12.

The linear thermoviscoelastic behavior was characterized by conducting creep and recovery test under sufficiently low applied stresses at various constant levels of temperature. These data determined the values of A_0, C, n and the functions $g_0(T)$ and $a_T(T)$ as well as the product of $g_1(T)$ and $g_2(T)$ for describing the transverse compliance of *APC-2* in the form of Equation 1.23. As noted previously these isothermal creep data did not suffice to obtain $g_1(T)$ and $g_2(T)$ individually. These isothermal creep tests were conducted on creep frames at fixed temperatures which ranged from 24^0C to 180^0C. Creep time was 20 min and recovery was measured over a period of 200 min. The values of the above-mentioned parameters and functions were determined by curve fitting techniques.

The nonlinear aspects of the response of *APC-2* were evaluted from isothermal creep and recovery tests which were conducted at several stress levels. These stresses ranged between 10 and 40 MPa and the temperatures ranged between 24^0 and 180^0C. Creep time was 60 min for 90^0 coupons and 50 min for 15^0 off-axis coupons. Recovery strains were recorded over periods of 600 min. The nonlinear parameters of Schapery's constitutive model were subsequently determined from creep-recovery data by means of curve fitting techniques.

1.3.3 Results of the Characterization

Master Curve and Temperature-dependent Shift Functions

Creep data within the linear range, plotted at various levels of temperature, are shown in Figure 1.4. These experimental creep data demonstrate a three-fold increase in compliance over the range of time and temperature recorded therein. This increase is an order of magnitude larger than the comparable rise in the compliance of epoxy matrix composites, demonstrating the importance of time dependence in the response of thermoplastic composites. The transverse creep compliance master curve in Figure 1.5 was constructed by

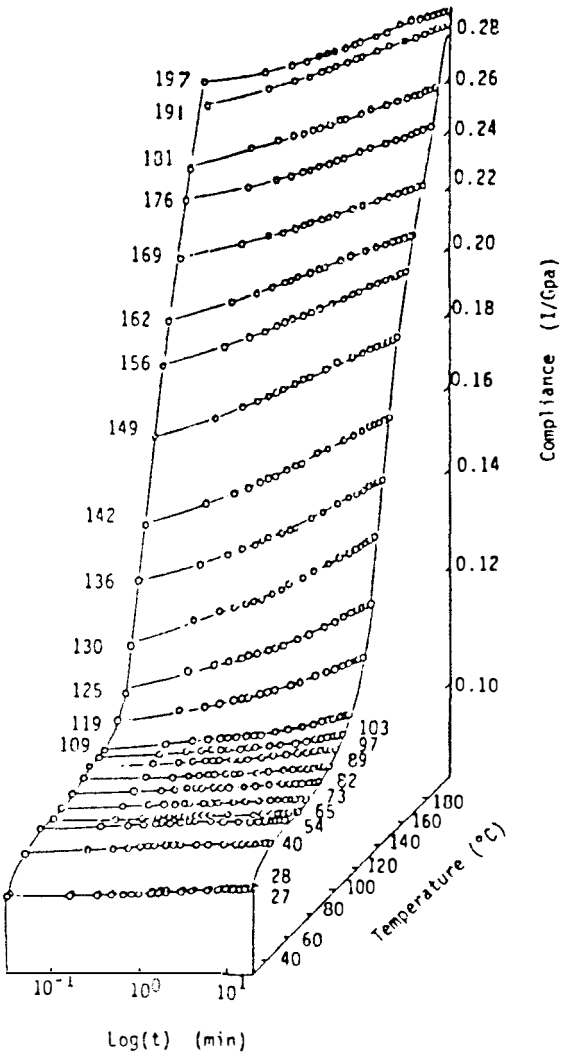

Figure 1.4: Temperature dependence of transverse creep compliance of *APC-2* unidirectional laminate, pp 109 Ref. [1]

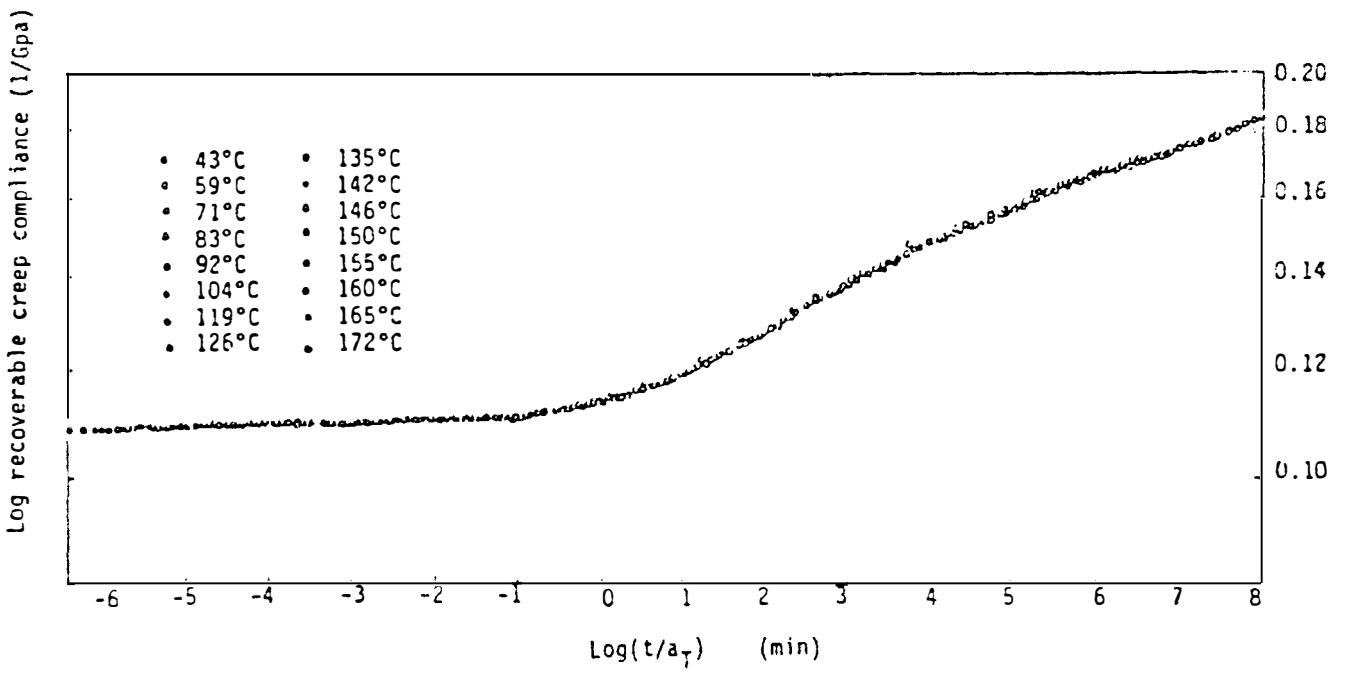

Figure 1.5: Master transeverse creep compliance curve of *APC-2* laminate ($T_{ref} = 119^0 C$). pp 116 Ref. [1]

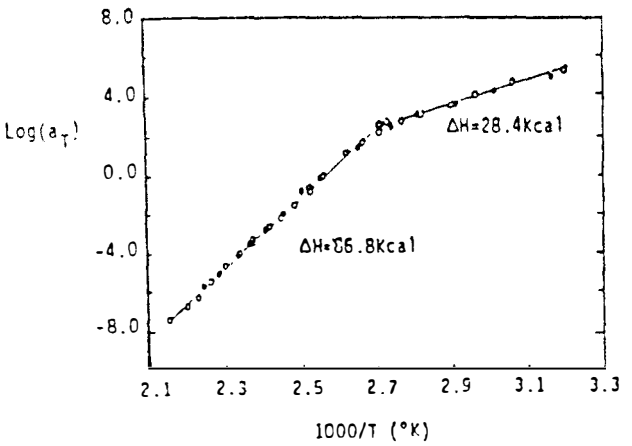

Figure 1.6: Horizontal shift versus temperature for creep compliance master curve. pp 117 Ref. [1]

horizontal shifting of the data in Figure 1.4. The corresponding horizontal shift factor a_T is related to the temperature T in Figure 1.6 by means of the familiar *Arrhenius* relation:

$$\log a_T = \frac{\Delta H}{2.303R}\left(\frac{1}{T} - \frac{1}{T_M}\right) \tag{1.27}$$

where R is the universal gas constant and ΔH is the activation energy per gram mole with $R = 1.987\ (cal/g - mole^0k)$.

The plot of $\log a_T$ against the reciprocal of absolute temperature consisted of two straight lines above and below 97^0C. The activation energy, which was estimated from the slope of line, was about $86.8kcal/mol$ at above T_g. Note that the compliance in Figure 1.5 ranges between value of 0.11 and 0.18 $(GPa)^{-1}$ in contrast with the range of 0.1 through 0.29 $(GPa)^{-1}$ in Figure 1.4. The discrepency is taken up by the vertical shift factor, which is drawn vs. temperature in Figure 1.7

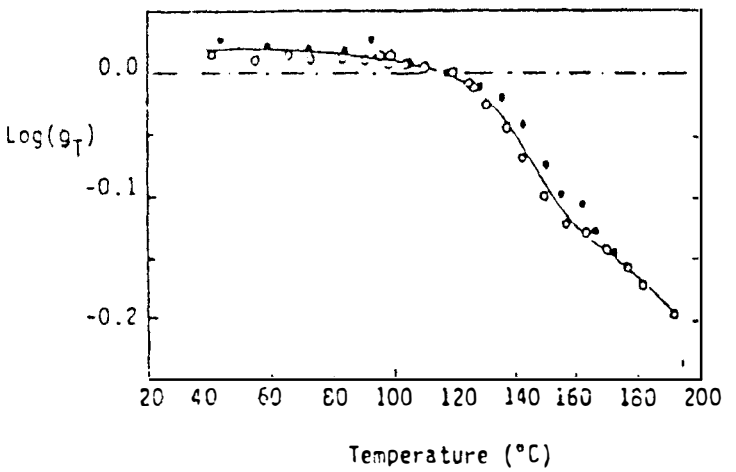

Figure 1.7: Vertical shift versus temperature for creep compliance master curve. pp 117 Ref. [1] (where $g = g_1 g_2$)

It is noted that the slope changes of the vertical shift factor, shown in Figure 1.7, occurred at about 129^0C and 160^0C. The former could be attributed to the glass transition, and the latter to the further crystallization. The vertical shift factor, provided in Reference [1], consists of the product $g_1(T)g_2(T)$ of the individual functions expressed in Equation 1.12.

Nonlinear Response and the parameter a_σ

Creep data which were recorded at various levels of stress and temperature demonstrated that both factors accelerate and accentuate the deformation of *APC-2* composites. The general trend is shown in Figure 1.8.

The nonlinear parameter a_σ, obtained by fitting the above-mentioned creep data through a least square numerical method, was characterized in terms of the octahedral shear stress

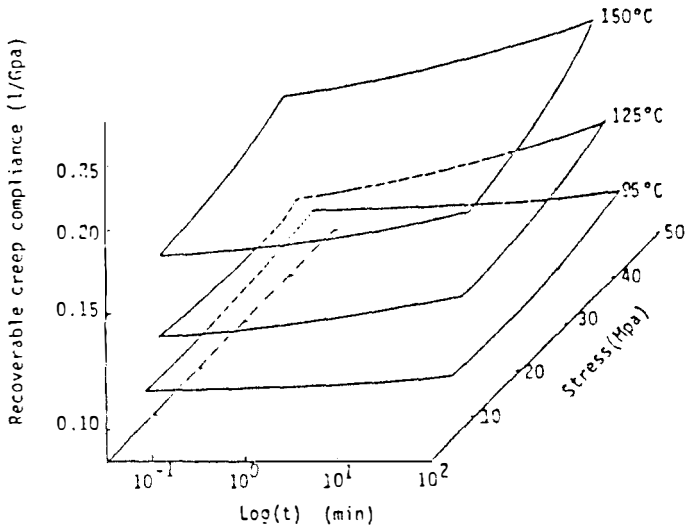

Figure 1.8: Transverse tensile compliance-stress-time relationship of typical *APC-2* composite laminate. pp 124 Ref. [1]

τ in the matrix according to

$$\begin{aligned} a_\sigma(\sigma) &= \exp(-\tilde{\alpha}(\tau - \tilde{\tau}_0)) \quad for \quad \tau > \tilde{\tau}_0 \\ a_\sigma(\sigma) &= 1 \qquad\qquad\qquad\qquad\quad \tau \leq \tilde{\tau}_0 \end{aligned} \tag{1.28}$$

For the case of uniaxially applied loads, τ in Equation 1.28 is equivalent to $\sqrt{2}\sigma/3$ where σ is uniaxial stress. In addition, $\tilde{\tau}_0$ is an experimentally determined threshold value , and $\tilde{\alpha}$ an experimental factor. In Figure 1.9, the open circles give discrete values of a_σ obtained by numerical fitting of tensile creep data for 15^0 off-axis coupons. The solid lines are then drawn to obtain the best fit with Equation 1.28.

The temperature dependence of the parameters $\tilde{\alpha}$ and $\tilde{\tau}_0$ is shown in Figure 1.10 and 1.11 respectively. It can be seen that $\tilde{\alpha}$ is temperature-independent below 91^0C. The temperature dependence of $\tilde{\alpha}$ above 91^0C is pronounced, but due to an insufficient number of data points, it was not possible to express $\tilde{\alpha}$ mathematically for this range. The parameter $\tilde{\tau}_0$, the threshold value of τ, falls on two straight lines below and above 91^0C. (Figure 1.11)

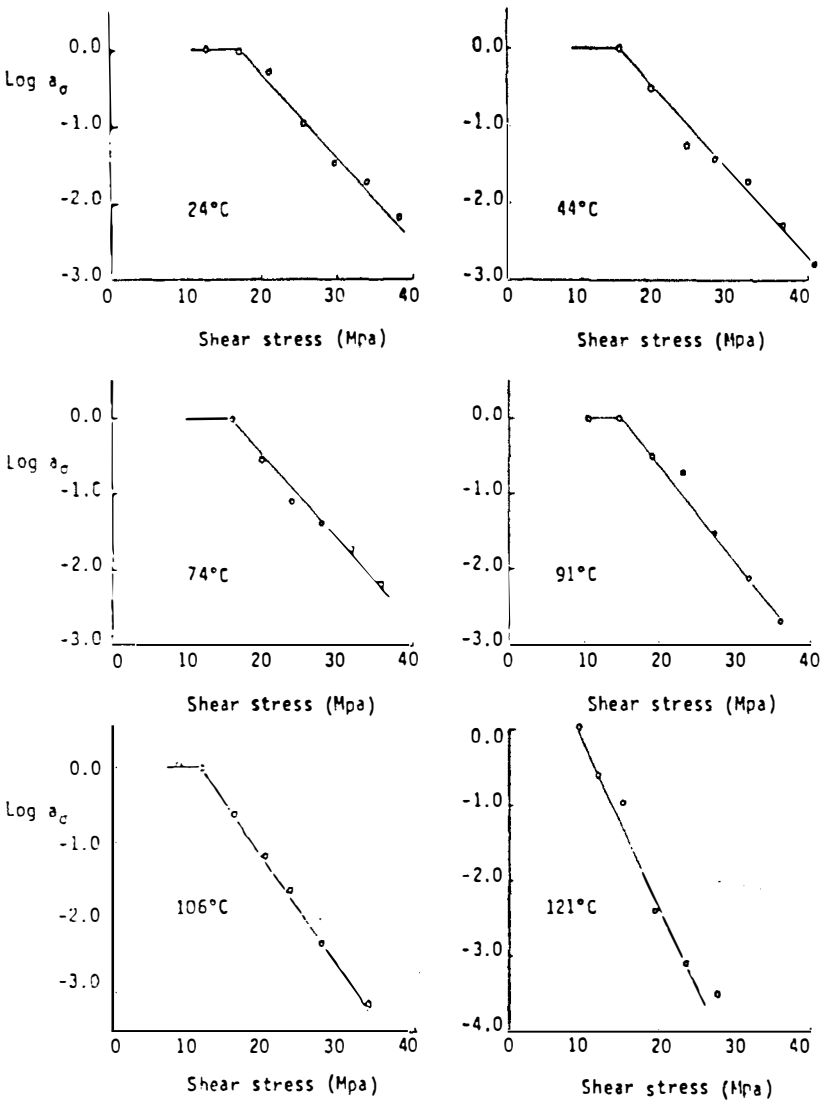

Figure 1.9: Time scale shift factor a_σ versus shear stress obtained by numerical method. pp 138 Ref. [1]

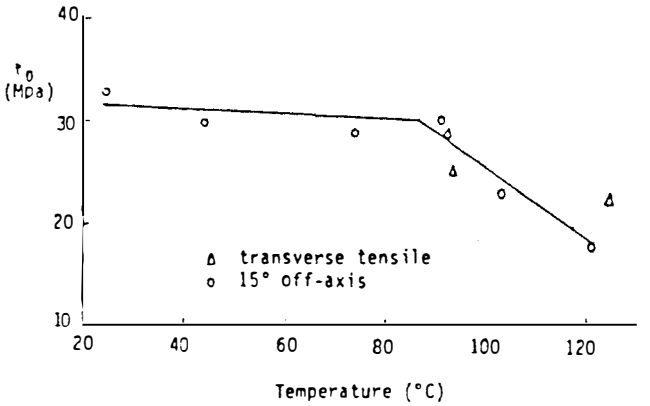

Figure 1.10: Temperature dependence of the parameter $\bar{\alpha}$. pp 139 Ref. [1]

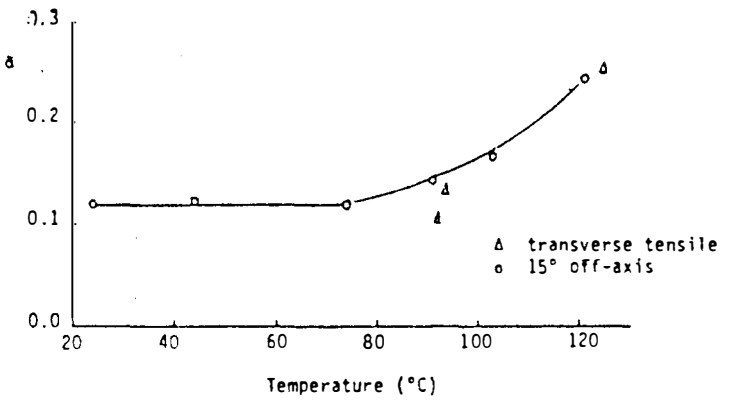

Figure 1.11: Temperature dependence of the parameter $\tilde{\tau}_0$. pp 139 Ref. [1]

Chapter 2

Linear Optimization Schemes

2.1 Introduction

The issue of optimal cool-down can be stated as follows: given an initial temperature, T_I, where the material is stress-free, a final temperature, $T_F (T_F < T_I)$, a finite time interval for cooling, t_f, and a thermoviscoelastic stress-strain relation (e.g. Equation 2.3), find the "best" time-temperature history, $T = T_\Omega(t)$, which minimizes the residual thermal stresses $\sigma(t_f)$ at time $t = t_f$.

During a finite cool down time interval, the time and the temperature introduce two contradictory effects on the relaxation process. Let us consider two extreme cases to illustrate these two effects. In the first scenario let the temperature drop immediately from T_I to T_F at $t = 0$. In this case the largest possible residual stresses are established instantaneously and the largest amount of real time, namely t_f, is available for their relaxation. However, relaxation takes place entirely at the low temperature level of T_F and thereby to the smallest effect. On the other hand, consider the second scenario where one allows only a small temperature drop prior to $t = t_f$. In this case the relaxation takes place at a temperature near T_I - whereby at a most significant amount. However, the small temperature drop activates only small residual stresses. Since, by hypothesis, it is necessary to reach the temperature T_F at time t_f, this state of affairs forces a large temperature change to occur at $t = t_f$, yielding large residual stresses without any spare time for relaxation. In the

second scenario too much time is "wasted" to relax too small stresses and not much time remains to relax large stresses. The actual optimal time-temperature path achieves the best interplay between these two competing effects so as to minimize the residual thermal stress at the termination of the cool down phase. In the past, analytic solutions were developed for optimal cool-down in the viscoelastic linear case [5] - [8].

The next section provides a brief overview for these analytic solutions. These concern three specific material responses; a Thermorheologically Simple Material with temperature-independent coefficients of thermal expansion, a Thermorheologically Simple Material with temperature-dependent coefficients of thermal expansion, and Thermorheologically Complex Material.

2.2 Formulation

2.2.1 Basic Equations

When elastic materials undergo cool-down against geometric constraints, the residual thermal stresses are given by

$$\sigma(t) = -E\alpha(T_I - T_F) \tag{2.1}$$

where E and α denote Young's modulus and the coefficient of thermal expansion respectively, and T_I and T_F are the initial and final temperatures (for cool-down, $T_I > T_F$).

Denote $T_I - T_F = \Delta T$ and assume an elastic geometric constraint, related by a factor r, then

$$\sigma(t) = -rE\alpha\Delta T \tag{2.2}$$

In the thermorheologically simple viscoelastic case, Equation 2.2 is modified to read as follows [7]:

$$\sigma(t) = -r\alpha \int_{0^-}^{t} E(\rho(t) - \rho(\tau)) \frac{d\Delta T}{d\tau} d\tau \tag{2.3}$$

where

$$\rho(u) = \int_0^u dp / \alpha(T(p)) \tag{2.4}$$

In Equation 2.4 $a(T)$ is the horizontal shift factor required to coalesce isothermal data (say creep) to form a "master curve."

For the thermorheologically complex response, employment of Schapery's viscoelastic constitutive model modifies Equation 2.2 to yield

$$\sigma(t) = -r\alpha v_1(T) \int_0^t E(\rho(t) - \rho(\tau)) \frac{d(v_2(T)\Delta T)}{d\tau} d\tau \tag{2.5}$$

where the product $v(T) = v_1(T)v_2(T)$ is the vertical shift which is required to coalesce isothermal data (say creep) to form a "master curve"[1] in addition to $a(T)$.

2.2.2 Case of Thermorheologically Simple Material

Weitsman[5] and Gurtin and Murphy [6] have proposed a variational method to determine optimal cool down paths that minimize the thermal residual stress in linearly viscoelastic materials.

In all previous studies [5] - [8], which concerned linear viscoelastic behavior, namely $a = a(T)$ but not $a = a(T, \sigma)$ in Equation 2.4, it was found that $T_\Omega(t)$ undergoes sharp drops at $t = 0$, from T_I to T_0 and at t_f, from $T_\Omega(t_f)$ to T_F, and it decreases monotonically and smoothly over the range $0 < t < t_f$.

Let $T(t)$ be the optimal path (unknown yet) and $\tilde{T}(t)$ be an adjacent path which is defined by

$$\tilde{T}(t) = T(t) + \epsilon\eta(t) \tag{2.6}$$

Substitution of Equation 2.6 into Equation 2.3 yields

$$\tilde{\sigma}(t) = -r\alpha \int_{0^-}^t E(\tilde{\rho}(t) - \tilde{\rho}(\tau)) \frac{d\Delta\tilde{T}}{d\tau} d\tau \tag{2.7}$$

where

$$\tilde{\rho}(u) = \int_0^u dp/a(\tilde{T}(p))$$

[1] It is obvious from Equation 4.3 that for $T = constant$, $\sigma(t)$ depends only on $v(T)$ and not on $v_1(T)$ and $v_2(T)$ separately. This no longer holds for fluctuating temperatures.

Applying the fundamental principles of variational calculus that the Euler's equation can be obtained from the stationary value

$$\frac{d}{d\epsilon}\tilde{\sigma}(t_f)\Big|_{\epsilon=0} = 0 \tag{2.8}$$

to Equation 2.7, an optimal time-temperature path was derived.

The derivation showed that the optimal cool-down path contains an instantaneous temperature drop from T_I to T_0 at $t = 0$. The magnitude of temperature drop $(T_0 - T_I)$ is expressed by

$$T_0 - T_I = \frac{a(T_0)}{a'(T_0)} \tag{2.9}$$

which is a transcendental equation in the unknown T_0. Beyond the initial drop, the optimal path $T_\Omega(t)$ is governed by the integro-differential equation

$$\frac{dT_\Omega(t)}{dt} = -\frac{E''(t,t_f)}{E'(t,t_f)}\frac{a'(T(t))}{a(T(t))a''(T(t))} \tag{2.10}$$

where

$$E(t,t_f) = E\left(\int_t^{t_f}\frac{ds}{a(T(s))}\right)$$

In Equation 2.10 primes denote derivatives with respect to the argument. Note that Equation 2.10 must be solved "backwards" from $t = t_f$, towards $t = 0$ since $E(t,t_f)$ cannot be evaluated without a-priori knowledge of the solution, with exception of $E(t_f,t_f) = E(0)$ - which is known. The procedure is to guess $T_\Omega(t_f^-)$, solve Equation 2.10 numerically and evaluate $T(0)$, then adjust the guess value of $T_\Omega(t_f^-)$ iteratively until $T(0) = T_0$ which matches the root of Equation 2.9.

These analytic methods have been applied to obtain optimal cooling paths for graphite/epoxy cross-ply composites and adhesive joints[7]. Weitsman [5] also obtained analytic solutions for optimal cooling in the special cases of a three element model and a "power law" response. The latter case corresponds to a wide range of real materials.

The Three Element Model

Consider a three element model for which the relaxation function is given by

$$E(t) = C_0 + D_0 \exp(\frac{-t}{\lambda}) \tag{2.11}$$

where $C_0 = E_\infty$, $D_0 = E_0 - E_\infty$. Assume now

$$a(T) = \exp\left(-\frac{T}{A} + B\right) \tag{2.12}$$

and $v_1 = v_2 = 1$. Straightforward manipulations yield [5];

$$T_\Omega(t) = A(B - \ln \phi(t)) \tag{2.13}$$

where $\phi(t) = \frac{t}{\lambda} + e^C$, $C = B + 1 - \frac{T_I}{A}$.

The initial and the final temperature drops in T_Ω are given by

$$\begin{aligned} T_0 &= T_I - A \\ T(t_f^-) - T(t_f^+) &= A[B - \ln \phi(t_f)] - T_F \end{aligned}$$

The residual thermal stresses during optimal cool-down are given by

$$\frac{(1-\nu)\sigma(t)}{\alpha} = A\left(C_0\left(\ln \phi(t) + \frac{T_I}{A} - B\right) + D_0\right) \tag{2.14}$$

$$\frac{(1-\nu)\sigma(t_f^+)}{\alpha} = C_0(T_I - T_F) + D_0\left(A(B + 1 - \ln \phi(t_f)) - T_F\right) \tag{2.15}$$

"Power Law" response

A power law representation of relaxation and creep data is frequently adapted for a broad range of materials from metals to polymeric composites, since it enables the fitting of data over a large time scale. For relaxation, the general power law has the form

$$E(t) = E_\infty + \frac{E_0 - E_\infty}{(1 + \frac{t}{\tau_0})^n} \tag{2.16}$$

where τ_0, n, E_0 and E_∞ are non-negative constants. A simple form of power law, which is a special case of the general power law is expressed by

$$E(t) = E_\infty + E_1 t^{-n} (0 \leq n < 1) \tag{2.17}$$

For simplicity, re-write Equation 2.16 in the following form:

$$E(t) = A_1 + A_2 (t + t_0)^{-n} \tag{2.18}$$

Assuming that $a(T)$ is given by Equation 2.12, straightforward manipulations yield [5];

$$\begin{aligned} T_0 &= \frac{1}{C_1} \ln \frac{L}{K} \\ T_\Omega(t) &= \frac{1}{C_1} \ln \frac{1}{K + C_1 t}, \quad L, K > 0 \end{aligned} \tag{2.19}$$

where $C_1 = \frac{n}{n+1}$. K and L are constants, which can be determined by solving the following simultaneous transcendental equations

$$\begin{aligned} L &= \left[\frac{t_o (K + C_1 t_f)^{\frac{1}{n}} \exp B}{(n+1)A} \right]^{\frac{n}{n+1}} \\ L &= K \exp(C_1 T_0) \end{aligned} \tag{2.20}$$

The residual thermal stresses during optimal cool-down are obtained by direct substitution of Equation 2.19 into Equation 2.3. After several manipulations we get

$$\begin{aligned} \frac{1-\nu}{\alpha} \sigma(t) &= A \left[A_1 + A_2 \left(Q \left(K^{-\frac{1}{n}} - (K + C_1 t)^{-\frac{1}{n}} \right) + t_0 \right)^{-n} \right] - \\ & \frac{A_1}{C_1} \ln \frac{K}{K + C_1 t} + A_2 I_2(t) \end{aligned} \tag{2.21}$$

where

$$\begin{aligned} I_2(t) &= Q^{-n} \int_0^t \left[(K + C_1 \tau)^{-\frac{1}{n}} - (K + C_1 t)^{-\frac{1}{n}} + \frac{t_0}{Q} \right]^{-n} \frac{d\tau}{(K + C_1 \tau)} \\ Q &= \frac{n \epsilon^{-B} L^{\left(\frac{n+1}{n}\right)}}{C_1} \end{aligned} \tag{2.22}$$

For the case of $n = \frac{1}{2}$, Equation 2.22 can be reduced to

$$I_2 = \frac{b}{C_1} Q^{-n} \left(\sin^{-1} \left(\frac{K + C_1 t}{b} \right) - \sin^{-1} \left(\frac{K}{b} \right) \right) \tag{2.23}$$

where

$$b = \frac{1}{\left((K + C_1 t)^{-\frac{1}{n}} - \frac{t_0}{Q}\right)^n}$$

2.2.3 Case of Temperature-dependent Coefficients of Thermal Expansion

Weitsman and Harper [7] modified expressions 2.9 and 2.10 to account for the effects of the temperature-dependence of the thermal expansion coefficicents. Such temperature dependence, namely $\alpha = \alpha(T)$, was observed in the the data of the graphite/epoxy composites. For this specific case the optimal path contains an initial drop which is given by

$$T_0 - T_I = \frac{\alpha(T_0)a(T_0)}{\alpha(T_0)a'(T_0) - \alpha'(T_0)a(T_0)} \tag{2.24}$$

while the Euler's equation of optimal path $T_\Omega(t)$ is given by

$$\frac{dT_\Omega(t)}{dt} = -\frac{E''(t,t_f)}{E'(t,t_f)} \frac{a'}{a\left(a'' - a'\frac{2\alpha' + \alpha''(T-T_I)}{\alpha + \alpha'(T-T_I)}\right)} \tag{2.25}$$

2.2.4 Case of Thermorheologically Complex Material

Harper [8] extended the previous works and derived a scheme to predict optimal temperature paths for a class of thermorheologically complex materials, where effects of temperature may be accounted for by both vertical and horizontal shift factors. For this class of thermorheologically complex viscoelastic response, the initial drop is given by

$$T_0 - T_I = \frac{v_2(T_0)a(T_0)}{a'(T_0)v_2(T_0) - v_2'(T_0)a(T_0)} \tag{2.26}$$

The Euler's equation of optimal path $T_\Omega(t)$ is given by

$$\frac{dT_\Omega(t)}{dt} = -\frac{E''(t,t_f)}{E'(t,t_f)} \frac{a'(v_2'(T-T_I) + v_2)}{a(a''(v_2'(T-T_I) + v_2) - a'(v_2'(T-T_I) + v_2)')} \tag{2.27}$$

Chapter 3

Nonlinear Optimization

3.1 Introduction

Problems of finding the optimal cool-down temperature path are directly related to the minimization of the final residual stress $\sigma(t_f)$. In general, when dealing with optimization problems which have no known analytic solutions, it is necessary to resort to approximation procedures by means of the iterative methods. In this chapter such an iterative convergent numerical scheme is developed to obtain the optimal cool down-path for simple, hypothetical nonlinear viscoelastic responses. Two prototype cases, the three element model and the "power law" response, are considered. Besides their relative simplicity, these prototypes have the distinct advantage of possessing complete or partial analytic solutions. In this manner some basis was made for comparisons between optimal paths which could be obtained numerically from the iterative scheme and corresponding paths derived analytically.

3.2 Formulation

Problems of finding zeros of a function are closely associated with minimization of a function

$$f(X) \tag{3.1}$$

where f is a real-valued differentiable function and X is a n-dimensional vector.

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3.2}$$

For if f has a local minimum at $X = X^*$, then we must have

$$g \equiv \nabla f(X^*) = 0 \tag{3.3}$$

where g is the gradient of f.

Let us consider the iterative schemes to find the *zeroes* of a given function Γ such that

$$\Gamma\,(T_\Omega^*) = 0$$

where

$$T_\Omega^* = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_n \end{bmatrix}.$$

In the above expression T_i represents $T(t_i)$, where $T(t)$ can be obtained by interpolation.

Beginning with an initial set of values $(T_\Omega)_0$, successive approximates $(T_\Omega)_i$ $(i = 0, 1, 2, \ldots)$ to $(T_\Omega)^*$ are computed with the aid of a *suitable* iteration function Φ:

$$(T_\Omega)_{i+1} := \Phi\,[(T_\Omega)_i] \quad i = 0, 1, 2, \ldots \tag{3.4}$$

If $(T_\Omega)^*$ is a fixed point of Φ; that is $(T_\Omega)^* = \Phi[(T_\Omega)^*]$, and if all fixed points of Φ are also zeroes of Γ and if Φ^* is continuous in the neighborhood of each of its fixed points, then $(T_\Omega)^*$ is a zero of the given function Γ [14].

The important key to the success of the iterative scheme is to obtain a suitable iteration function Φ. In this dissertation it is suggested that such a function for the nonlinear optimization problem at hand can be obtained by the formulation which is based upon the known explicit linear solution. In the linear viscoelastic case the final stress and the optimal path can be expressed by

$$\sigma(t_f) = \Psi(T_\Omega^L) \tag{3.5}$$

$$T_\Omega^L(t) = \Phi^*(E', E'', a', a'', \nu') \tag{3.6}$$

where Ψ and Φ^* are expressed by integral equations. In the present case these equations are given in Equations 2.5 and 2.27. In nonlinear viscoelasticity the most significant effects of nonlinearity in stress-strain behavior can be incorporated into the stress-affected horizontal shift-factor function alone such that:

$$a = a\,(T(t); \sigma(t)) \tag{3.7}$$

It is now hypothesized that the optimal nonlinear temperature path can be constructed in an iterative manner by incorporating the stresses $\sigma(t)$ which arise along the optimal path *at the very same iteration* into Equation 3.7. Furthermore, since those stresses $\sigma(t)$ correspond to the optimal temperature $T(t)$ at each given iteration, elimination of t between $\sigma(t)$ and $T(t)$ yields a relation $\sigma = \hat{\sigma}(T)$, whereby Equation 3.7 can be rewritten as

$$a = a\,(T(t); \sigma(T(t))) - \hat{a}\,(T(t)) \tag{3.8}$$

The primary significance of Equation 3.8 is that shift-factor function $\hat{a}$ no longer contains σ explicitly; therefore, the optimal temperature path $\hat{T}_\Omega(t)$ and stress history $\hat{\sigma}(t)$ along the temperature path can be obtained by using Equation 3.6. Consequently Φ^* satisfies the qualification of being a suitable iterative function if $(T_\Omega)^*$ is a fixed point of Φ^*, that is $(T_\Omega)^* = \Phi^*[(T_\Omega)^*]$, and if Φ^* is continuous in the neighborhood of each of its fixed points. These conditions are assumed to hold.

3.3 The Three Element Model as a Prototype Case

3.3.1 An Analytic Solution for the Optimal Path

In this sub-section an analytic solution is derived for the optimal cool-down path for a special form of nonlinear behavior. The analytic solution is obtained iteratively. The analytic expression for the the optimal temperature path for the linear case (Eq. 2.10) serves as an *iteration function* for the nonlinear iterative scheme. Since it is possible to obtain an analytic solution in the present case, this circumstance provides at least a partial

test for the iterative numerical scheme, which will be extended in the next chapter to the more complicated case of *APC-2*.

Consider the three element model, with response as prescribed in Equation 2.11, and a shift factor as given in Equation 2.12. The corresponding optimal cool-down path and residual stresses are listed in Equations 2.13 and 2.14

The key step in our approach calls for relating the temperature and stress along the optimal temperature path, which are expressed in the time domain. Thus, combining Equation 2.13 into Equation 2.14 leads to the following expression:

$$\sigma(t) = -kT_\Omega(t) + kT_I + mA \tag{3.9}$$

where $k = \alpha E_\infty/(1-\nu)$, and $m = \alpha E_0/(1-\nu)$. Note that Equation 3.9 is expressed in terms of temperature only.

Nonlinearity is introduced by considering

$$a = a(T,\sigma) = \exp\left(-\frac{T}{A} + B - \beta\sigma\right), \quad \beta > 0 \tag{3.10}$$

However, in view of Equation 3.9 the stress along the optimal linear path is in the form of $\sigma = \sigma(T)$.

The optimal cooling-path can be obtained by iteration, using linear results Equations 2.13 and 2.14 as initial guesses which are denoted by A^0, B^0, $T_\Omega^{(0)}$, σ^0. Namely, by inserting the linear results, Equations 2.13 and 3.9, into Equation 3.10. After straightforward manipulations we obtain

$$a = a(T,\sigma) = \exp\left(-\frac{T^{(0)}}{A^{(1)}} + B^{(1)}\right) \tag{3.11}$$

where $A^{(1)} = A^{(0)}/(1-\beta kA^{(0)})$, $B^{(1)} = B^{(0)}[1-\beta(kT_I + mA^{(0)})/B^{(0)}]$ with $A^{(0)} = A$, $B^{(0)} = B$. As intended, the Expression 3.11 no longer contains the stress σ explicitly, hence it is possible to write $a(T,\sigma) = a^{(1)}(T)$ and, ***most fortuitously indeed***, $a^{(1)}(T)$ happens to be of the very same form as $a(T)$. Consequently, $T_\Omega^{(1)}(t)$ and $\sigma^{(1)}(T)$ will retain the same

forms as Equations 2.13, and 2.14 but with $A^{(1)}$ and $B^{(1)}$ replacing A and B. At this stage, $\sigma^{(1)}(t)$ is substituted into Equation 3.10 and a revised $a(T,\sigma)$, as readily seen, is obtained in the form of $a^{(2)}(T) = \exp(-T^{(1)}/A^{(2)} + B^{(2)})$ where $A^{(2)} = A^{(1)}$ and $B^{(2)} = B^{(0)}[1 - \beta(kT_I + mA^{(1)})/B^{(0)}]$, which is still of the same form as the linear $a(T)$. An additional substitution yields $A^{(3)} = A^{(2)}$ and $B^{(3)} = B^{(2)}$ indicating convergence after three iterations. Consequently, the optimal path for the three element model with viscoelastic *nonlinearity* introduced through expression (Eq. 3.10) is

$$\begin{aligned} T_\Omega(t) &= \hat{A}(\hat{B} - \ln\hat{\phi}(t)) \\ T_\Omega(0^+) &= T_I - \hat{A} \end{aligned} \tag{3.12}$$

where $\hat{A} = A/(1 - \beta kA)$, $\hat{B} = B[1 - \beta(kT_I + m\hat{A})/B]$, and $\hat{\phi}(t) = t/\lambda + \exp(1 + \hat{B} - T_I/\hat{A})$.

3.3.2 A Numerical Solution for the Optimal Path

In the computations, expressions 2.11, 2.12, and 3.10 were employed with $A = 10.0$, $B = 1.0$, $C_0 = 0.1$, $D_0 = 1.0$, $\lambda = 1.0$, as well as $\alpha = 10^{-5}$, and $\nu = 0.25$. The initial stress-free temperature, T_I, was chosen to be $150^0 C$, the final temperature, T_F, $-50^0 C$, and a cooling time, t_f, 100 min. The control parameter of nonlinearity β was considered 1000.

In the numerical, iterative optimization scheme, the first iteration consisted of the determination of the optimal temperature path which corresponds to the linear viscoelastic case. The optimal temperature path, $T_\Omega(t)$, was obtained numerically by step-by-step integration of Equation 2.10 with the intermediate guesses of T_F until $T(0)$ was reached the prescribed value of T_0, as determined from Equation 2.9. The sequential values of T_i were computed from T_{i-1} of the previous step, backwards from $t = t_f$ towards $t = 0$, according to

$$T_i = T_{i-1} + \frac{dT}{dt}\Delta t_i \tag{3.13}$$

where dT/dt was evaluated by means of Equation 2.10 and Δt_i denotes the width of i_{th} time increment. To achieve both accuracy and efficiency in determing the optimal path,

nonuniform time intervals Δt_i were selected, according to considerations discussed later in this sub-section.

Once the optimal temperature path T_Ω was determined numerically in a fairly efficient manner, the stress history $\sigma(t)$ along the T_Ω was obtained also numerically by solving the integral Equation 2.3. For the purpose of numerical correlation $\sigma = F(T)$, values of $\sigma(t)$ were obtained so as to establish a list of numerically corresponding values of T_i and σ_i along the linear optimal path, resulting in $\sigma_i = F(T_i)$ at n discrete locations. With these n sets of data (σ_i, T_i) the interpolating function of $\sigma = F(T)$ was generated by means of the IMSL routine "CSCON". This routine created a shape-preserving cubic spline interpolant which is consistent with these n sets of data, obtained from the previous iteration. The consequent interpolating functions of derivatives of $F(T)$ were also generated by means of IMSL routine "CSDER," which evaluated the derivatives of the same cubic spline function for $\sigma = F(T)$.

Generating a relation of $\sigma = F(T)$ and, therefore, preparing for the evaluation of numerical values for a modified shift factor function $\hat{a}(T)$ is the starting point of the next iteration. A revised optimal path was to be generated with $\hat{a}(T) = a(T, F(T))$ replacing $a(T)$ in Equations 2.9 and 2.10. Upon obtaining a modified optimal path in any given iteration, the whole procedure was repeated until the following convergence criterion was satisfied:

$$\frac{1}{t_f}\sqrt{\int_0^{t_f}(T_i^{k+1}(t) - T_i^k(t))^2 dt} \;<\; \epsilon \tag{3.14}$$

In computing the intermediate temperatures T_i according to Equation 3.13, it was necessary to pay ongoing attention to the retention of numerical stability at all intermediate times t_i. In particular, the slope of the optimal path underwent extreme changes in the vicinity of $t = 0$ so that even very small variations in the guess values of the temperature T_f at $t = t_f^-$ resulted in substantial discrepancies in $T(0)$. Consequently, it was necessary to develop several specific trial-and-error tests to get a value of T_f which yielded $T(0)$ at

least in the neighborhood of T_0, as given by Equation 2.9, within the computable range[1] of temperatures. In spite of those careful improvements, the values of $T(0^+)$ and T_0, as determined by our computational scheme and Equation 2.9, respectively, could not be matched accurately in some circumstances. Fortuitously, these uncertainties had very little significance in determining the optimal paths for times beyond some fractions of a second from onset of cooling. The reason for the trivial effect of having $T(0^+) \neq T_0$ is that the optimal temperature paths contains precipitous drops from T_I to T_0 at $t = 0$ which were followed by extremely steep drops during $0 < t < \epsilon$ (with ϵ amounting to just fractions of a second). Therefore, exact matchings of $T(0^+)$ and T_0 were of little consequence for the overall dependence of T_Ω on time.

When optimal temperature paths were computed by using the numerical, iterative scheme, all optimal paths reached convergence within about four iterations in computations.

Comparison between results from the analytic solution and the predictions from the numerical, iterative method are provided in Figures 3.1 - 3.4. Figures 3.1 and 3.2 show the optimal cool-down paths, $T_\Omega(t)$ vs. t, for the linear and nonlinear cases respectively. And Figures 3.3 and 3.4 exhibit residual stress build-ups during optimal cool-downs, which correspond to the two cases shown in Figures 3.1 and 3.2. All figures show very good agreement between these two approaches. Also, a comparison between values of final stresses, provided in Table 3.1, shows very good agreement between the computational and analytic results. The discrepancies between numerical and analytical approaches were about 0.69% for the linear case, and 0.57% for the nonlinear case. It is also noted that the predicted nonlinear effect, which was about 9.3%, was almost the same for both methods.

It is observed that, for the three element model considered here, there were no noticeable differences in the optimal temperature paths between the linear and the nonlinear cases.

[1]The first jump is always directed downward. By hypothesis, the temperature T is assumed to drop monotonically over the time period $(0, t_f)$. Therefore, it is appropriate to require that none of the intermediate temperatures T_i should exceed T_I during the computation.

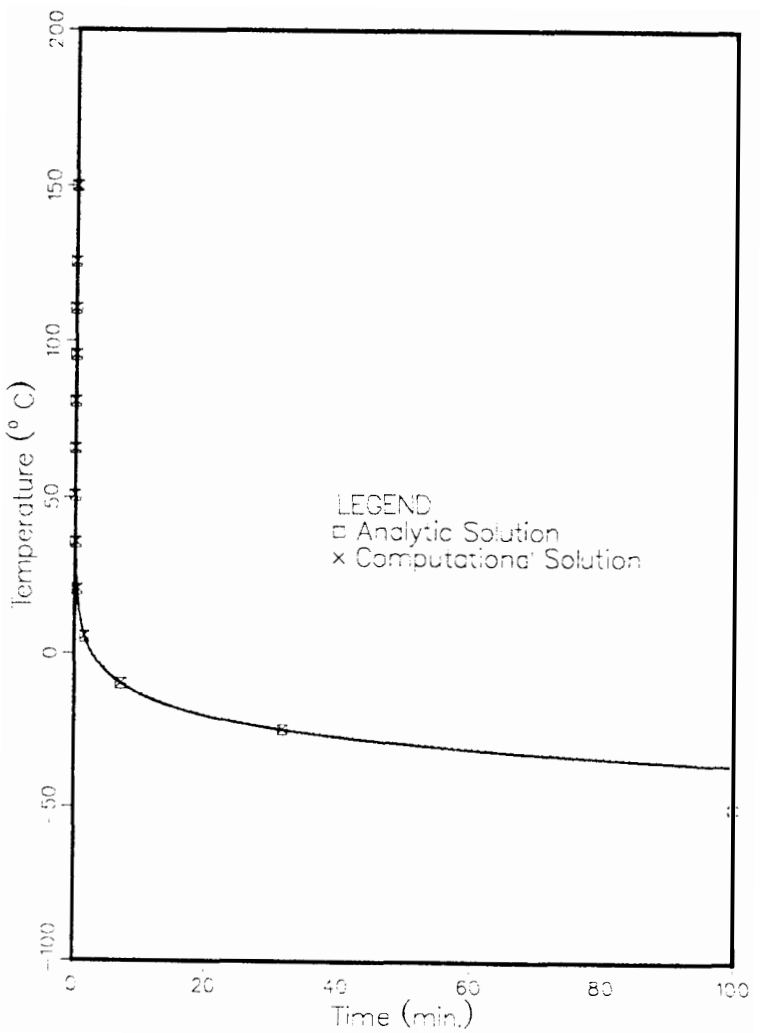

Figure 3.1: Optimal temperature paths for three element model (linear case)

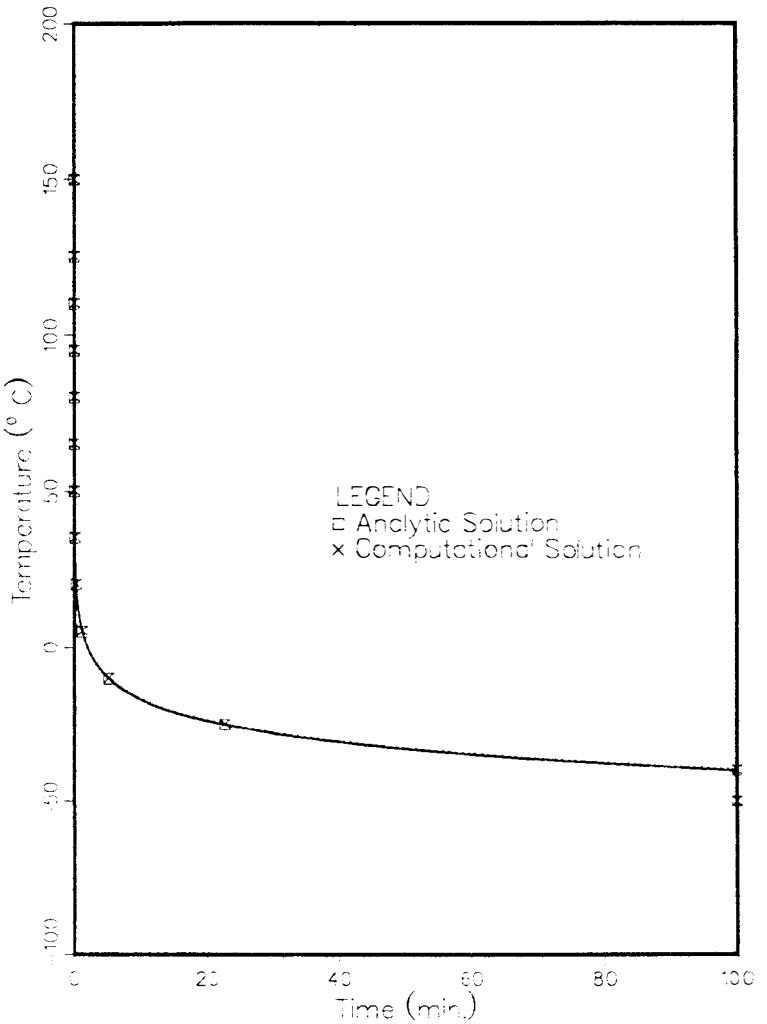

Figure 3.2: Optimal temperature paths for three element model (nonlinear case)

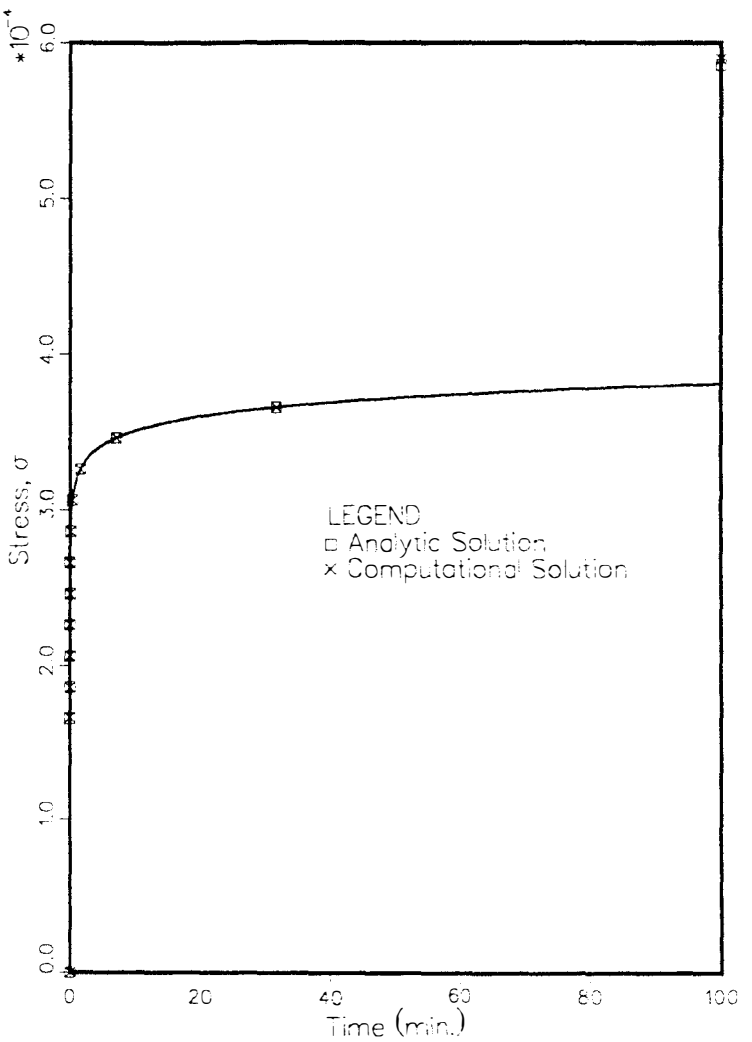

Figure 3.3: Stress histories along optimal temperature paths for three element model (linear case)

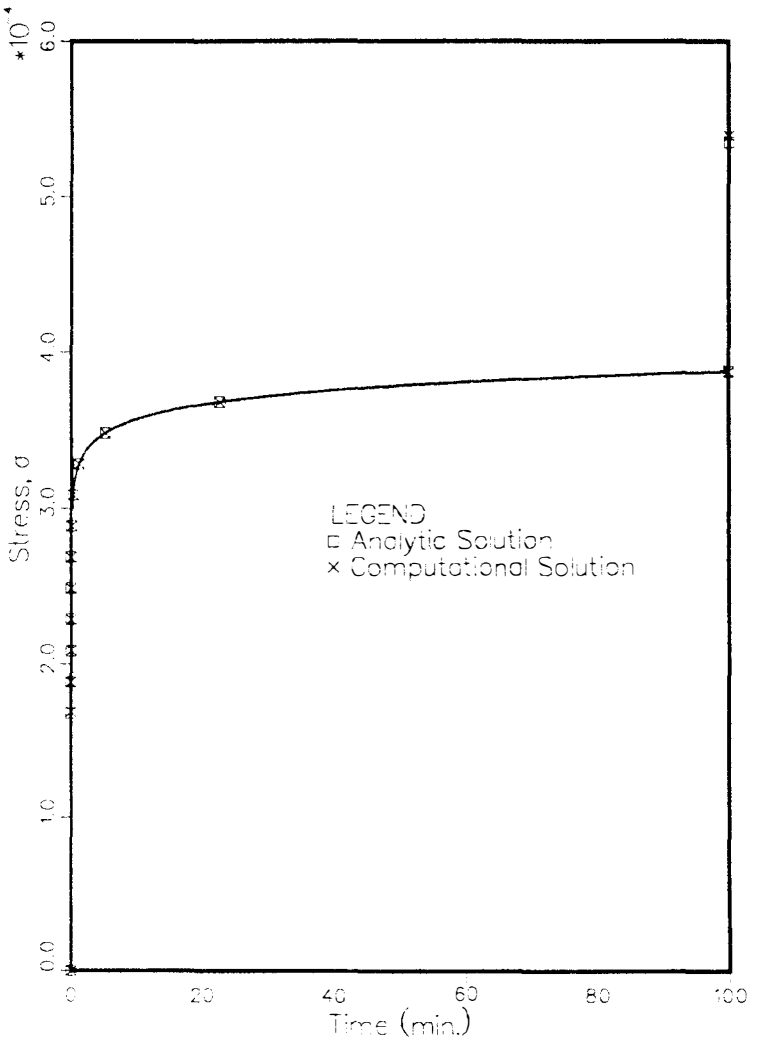

Figure 3.4: Stress histories along optimal temperature paths for three element model (nonlinear case)

Table 3.1: Comparison of final stresses for a prototype case of three element model between analytic solution and iterative scheme

Description	Final Stress	
	analytic solution	numerical scheme
Linear case	$.585977(10^{-3})$	$.589353(10^{-3})$
Nonlinear case	$.535630(10^{-3})$	$.539362(10^{-3})$

Both optimal temperature paths underwent very rapid temperature drops at the initial cool-down stage. but soon followed distinct slower temperature variation. This phenomenon suggested that subdivision of the time span t_f into equal time sub-intervals $\Delta t = t_f/n$ was unsuitable for performing the numerical integration. It was deemed advantageous to select unequal time intervals Δt_k which assured equal temperature drops ΔT. The determination of the unequal sizes of Δt_k required an iterative process. It was found that sufficient accuracy was obtained with fewer than 300 intervals.

The satisfactory agreement, exhibited in this section, between the iterative, numerical results and the analytical solutions encouraged us to employ the numerical scheme in more complicated circumstances, when analytical solutions were not available.

3.4 A Further Prototype Case; "Power Law" Response

The second case study involves the "power law" response, given in Equation 2.18. For this case analytic solutions for $T_\Omega(t)$, and for $\sigma(t)$ along the optimal path were derived for the linear case with a shift factor function given in Equation 2.12. However, for a nonlinear shift factor, such as occurs in Equation 3.10, no analytical result for $T_\Omega(t)$ is available. The

Table 3.2: Comparison of final stresses for a prototype case of power response law model between analytic solution and iterative scheme

Description	Final Stress	
	analytic solution	numerical scheme
Linear case	.001629	.0016228
Nonlinear case	not available	.0015145

"power law" case provided an opportunity to check the numerically attained $T_\Omega(t)$, with the corresponding values of $\sigma(t)$, against analytical results for the linear case only.

In the computations, Equations 2.18 and 3.10 were employed with $A = 10.0$, $B = 2.0$, $A_1 = 0.1$, $A_2 = 0.031623$, $t_o = 0.001$,$n = 0.5$ as well as $\alpha = 10^{-5}$, and $\nu = 0.25$. Also, the initial stress-free temperature, T_I, was chosen to be $150^0\,C$, the final temperature, T_F, $-150^0\,C$,and a cooling time, t_f, 100 min. The control parameter of nonlinearity, β, was considered 1500.

The computational scheme followed the previously outlined approach. Optimal temperature paths in the "power law" response, determined by means of the iterative scheme, converged within less than four iterative steps. Computational results are plotted in Figures 3.5 - 3.8. The linear and nonlinear optimal cool-down paths for the power law model are shown in the Figures 3.5 and 3.6. Figures 3.7 and 3.8 exhibit residual stress build-ups during optimal cool-downs which correspond to the paths shown in Figures 3.5 and 3.6. The results for the linear case, shown in Figures 3.5 and 3.7, afford comparisons between analytical predictions and numerical values. These results show good agreement between two approaches. Also, results of the final stresses are listed in Table 3.2. For the linear

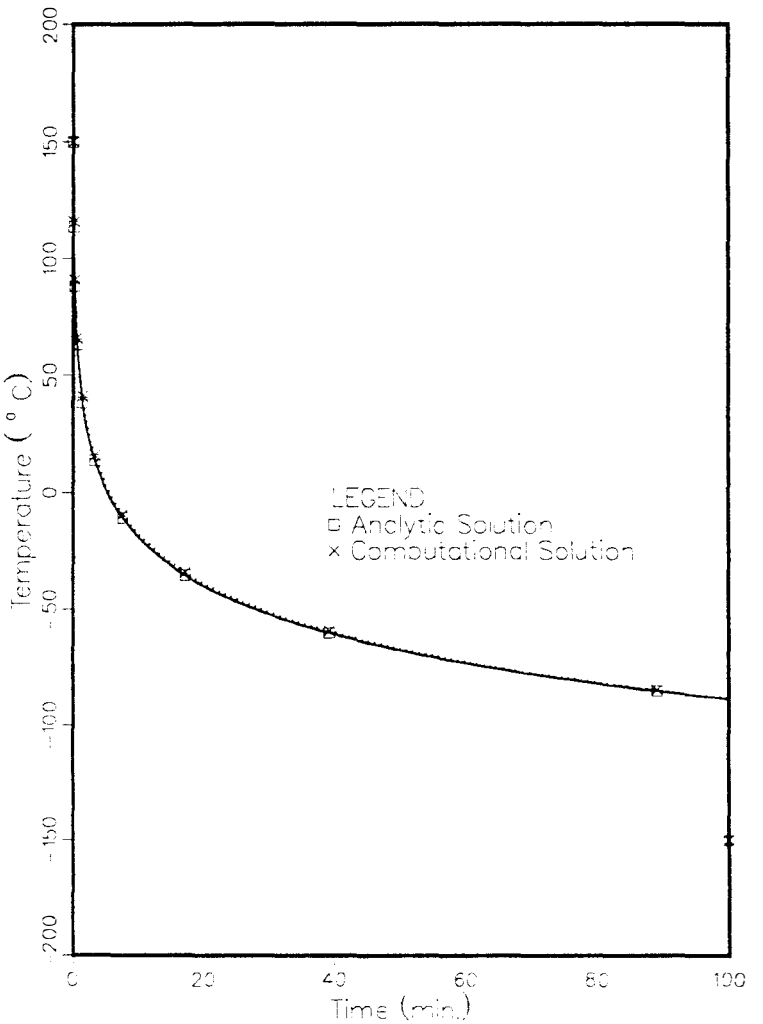

Figure 3.5: Optimal temperature paths for power law model (linear case)

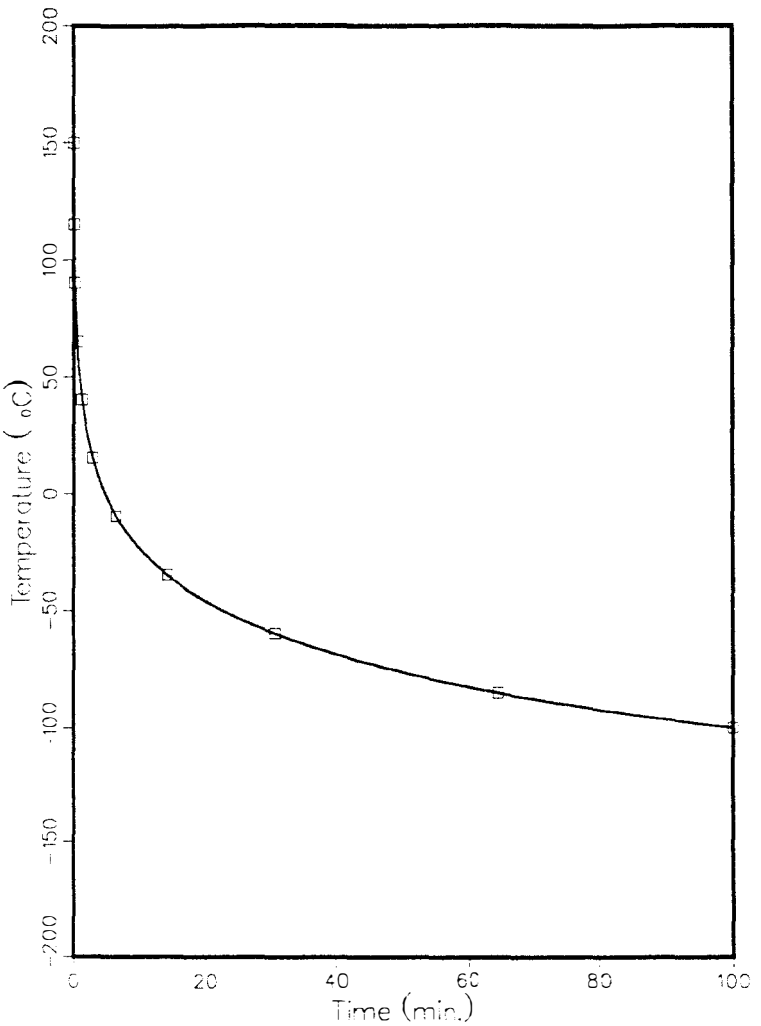

Figure 3.6: Optimal temperature path for power law model in nonlinear case (Computational results only)

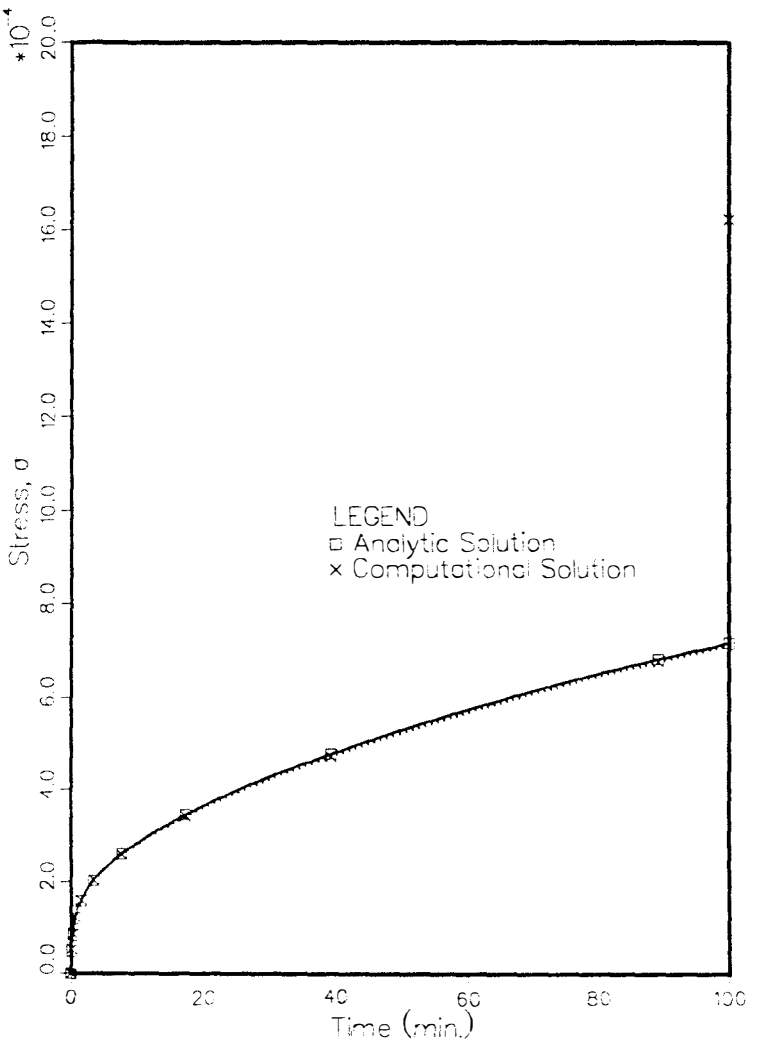

Figure 3.7: Stress histories along optimal temperature paths for power law model (linear case)

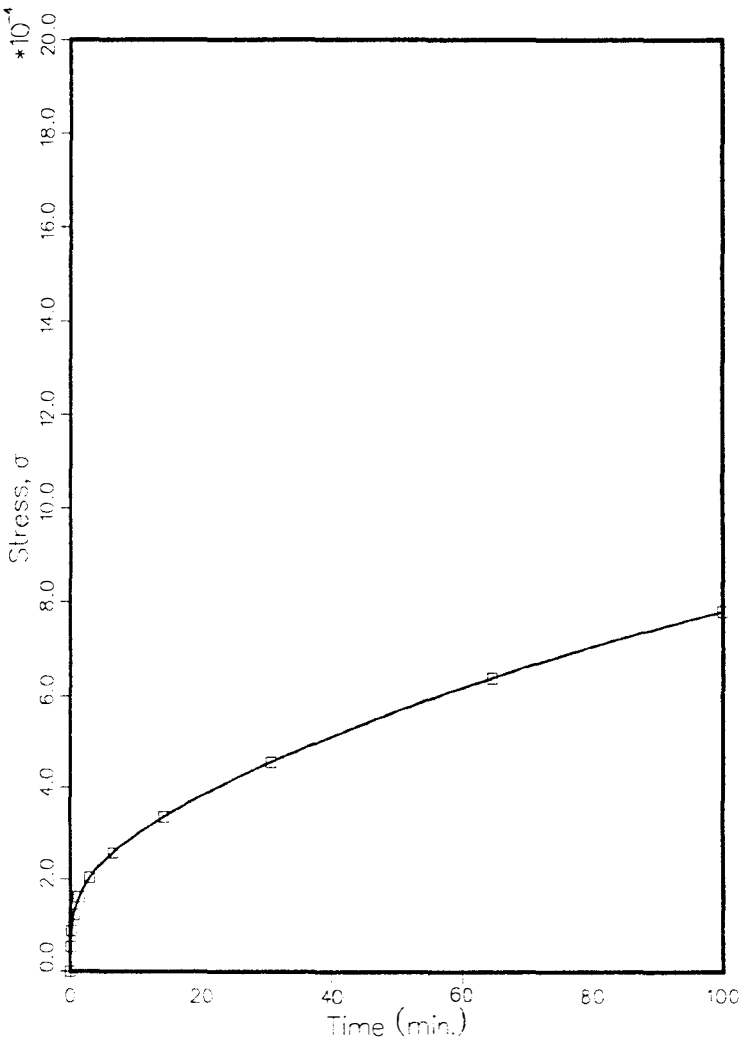

Figure 3.8: Stress history along optimal temperature path for power law model in nonlinear case (Computational results only)

case, the numerical scheme and the analytic solution yielded values for the residual thermal stresses at $t = t_f$, which are in very good agreement. The discrepancy between two approaches was about 0.35%.

The optimal temperature paths for the "power law" response possess some characteristics which resembles those for the prototype case of the three element model, namely

- no noticeable changes were observed between the optimal paths for the nonlinear and linear cases.
- the optimal temperature paths underwent very rapid drops at the initial cool-down stage, but. thereafter, followed a continuous and distinctly slower temperature variation.

It is also noted that for the present choice of material parameters the nonlinear effect on $\sigma(t_f^+)$ is about 6.67%.

Chapter 4

Optimal Cool-Down of $[0/90]_s$ APC-2 Composites.

4.1 Introduction

This chapter presents a convergent, iterative, numerical scheme to determine the optimal cool-down temperature path, which minimizes residual thermal stresses, in symmetric, balanced, cross-ply [1] *APC-2* composite laminate at the termination of cool-down. The response of the *APC-2* composite laminate was modeled as that of a nonlinear, thermorheologically complex, viscoelastic material [2].

Computations were conducted based upon the experimental data obtained by Xinran [1], as outlined in Chapter 1. These data were reduced to forms which can be fitted to the constitutive model of Schapery [2], which includes the following components:

1. a horizontal time-temperature shift-factor function
2. a vertical time-temperature shift-factor function
3. a stress-affected horizontal shift-factor function [3]. This effect accounts for the nonlinearity in the stress-strain behavior.

[1] A cross-ply laminate consists of a lay-up of 0 and 90 degree plies.

[2] It was also assumed that the laminate is sufficiently thin, so that thermal transients across its thickness may be disregarded. In other words the termperature T is spatially uniform throughout the laminate, with a temperature variation $T = T(t)$ matching the fluctuations in the ambient temperature.

[3] In principle, the vertical shift-factor can also depend on stress, but it appears that for *APC-2*, as well as many other resins, it is possible to account for nonlinear behavior by confining the stress effects to the horizontal shift function alone.

4. a non-recoverable component of strain, which depends on the stress and temperature

The latter component appears to play a minor role in the response of $APC\text{-}2$ and is neglected in the present study.

It should be noted that the data base of Reference [1] does not suffice for our present purpose, since it covers the temperature range $20^0C < T < 200^0C$, while acutal cool-down occurs between $T_M \approx 400^0C$ and $T_R \approx 20^0C$. It was, therefore, necessary to employ "reasonable" extrapolations of the data base to accomplish our computational tasks.

The brief outline of the current chapter is as follows. Section 2 lists the basic equations related to the current work and re-states the basic algorithm of the iterative scheme for obtaining the optimal cool-down path with the nonlinear viscoelastic response at hand. Section 3 presents the experimental data in forms which are reduced to suit the computational scheme. That section also explains the procedures used for the data reduction. Finally, the computational scheme and results for $APC\text{-}2$ are presented and discussed in section 4. The discussion focuses on several problems and difficulties which were encountered during the computations.

4.2 Analytic Background

The linear thermo-elastic response of a uni-directionally reinforced ply undergoing a temperature excursion $\Delta T = T - T_I$ from some stress-free reference temperature, T_I (say), is given by [4]

$$\begin{aligned} \epsilon_L - \alpha_L \Delta T &= \frac{\sigma_L}{E_L} - \frac{\nu_{TL}}{E_T}\sigma_T \\ \epsilon_T - \alpha_T \Delta T &= -\frac{\nu_{LT}}{E_L}\sigma_L + \frac{\sigma_T}{E_T} \end{aligned} \tag{4.1}$$

In Equation 4.1 ϵ and σ denote strain and stress. Also α, E, and ν denote coefficient of thermal expansion, Young's modulus, and Poisson's ratio, respectively. Subscripts L and T indicate longitudinal and transverse directions. These notations will be employed as standard symbols throughout this section.

In the particular case of a symmetric balanced cross-ply lay-up consisting of an equal number of 0 and 90 degree plies, we have following relations;

$$\sigma_L = -\sigma_T = \sigma, \epsilon_L = \epsilon_T = \epsilon$$

A straightforward employment of laminate theory yields the following expression for the laminate-level thermal stresses

$$\sigma = -rE_T\alpha\Delta T \tag{4.2}$$

where

$$\frac{1}{r} = (1 + \nu_{TL})\left(1 + \frac{1 + \nu_{LT}E_T}{1 + \nu_{TL}E_L}\right)$$

and $\alpha = \alpha_T - \alpha_L$.

Note that the form of Equation 4.2 remains valid for other symmetric lay-ups (in particular, quasi-isotropic lay-ups) but with different expressions for r. In the case of *APC-2*, the most pronounced time dependence occurs in the transverse modulus E_T, as can be seen from Figure 1.4. The remaining quantities in Equation 4.1 are [15]

$$E_L = 112GPa,\ \nu_{LT} = 0.31,\ \nu_{TL} = 0.0083$$

The values of α_T and α_L are stated in Section 4.3.4. Therefore, since $E_T/E_L \ll 1$, we may ignore the time depedence of r and for the specific Graphite/PEEK lamina under consideration let $r = 0.9125$ to within $\pm 2\,\%$ error.

For the thermorheologically complex response, employment of Schapery's nonlinear viscoelastic model modifies Equation 4.2 in a manner which resembles Equation 1.10 to yield

$$\sigma(t) = -r\alpha v_1(T)\int_0^t E(\rho(t) - \rho(\tau))\frac{d(v_2(T)\Delta T)}{d\tau}d\tau \tag{4.3}$$

with the "reduced times," $\rho(t)$ and $\rho(\tau)$, in Equation 4.3 given by

$$\rho(u) = \int_0^u dp/a(T;\sigma(p)) \tag{4.4}$$

In Equation 4.4 $a(T;\sigma)$ is the stress-affected horizontal shift factor.

Though both r and a may, in principle, depend on σ – it appears that the nonlinearity of the response can be expressed through $a(T;\sigma)$ alone. Under fluctuating temperatures, with $T = T(t)$, it is necessary to re-write Equation 4.3 with $v_1(T(t))$ and $v_2(T(\tau))$. In this case we also have $\Delta T = \Delta T(\tau)$ in Equation 4.3 and $a = a(T(p);\sigma(p))$ in Equation 4.4.

4.3 Reduction of *APC-2* Response Data

4.3.1 Relaxation Modulus

The isothermal creep compliance data shown in Figure 1.5 were reduced by Xinran [1] to expressions which fit Schapery's model. However, present analysis requires values which involve the relaxation modulus $E(t)$, rather than creep compliance $D(t)$. For this purpose $E(t)$ was derived from $D(t)$ by the well known relation [17]:

$$E(t) = \frac{1}{D(T)} \frac{\sin(\pi p(t))}{\pi p(t)} \tag{4.5}$$

where $p(t) = d\log D(t)/d\log t$.

From the experimental creep data 14 data points were selected, and then converted to the relaxation moduli using Equation 4.5. With this data set, piecewise interpolating polynomials were generated using least square error fits, and combined in a way that assured the continuity of the E, E' and E''. It is also noted that for the data at hand the cruder approximation $E(t) \approx 1/D(t)$ did not deviate by more than $2\% \sim 3\,\%$ from Equation 4.5.

In view of the fact that the actual creep data were coalesced into a master compliance curve which gave $D(t)$ vs. $\log t$, rather than vs. t, it was decided to utilize a $\log t$ scale for $E(t)$ and for the interpolating functions of E as well. This necessitated the employment of the following relations:

$$\frac{dE(t)}{dt} = \log e \frac{dE(\log t)}{d\log t} \frac{1}{t} \tag{4.6}$$

$$\frac{d^2E(t)}{dt^2} = -\log e \frac{dE(\log t)}{d\log t} \frac{1}{t^2} + (\log e)^2 \frac{d^2E(\log t)}{d\log t^2} \frac{1}{t^2} \tag{4.7}$$

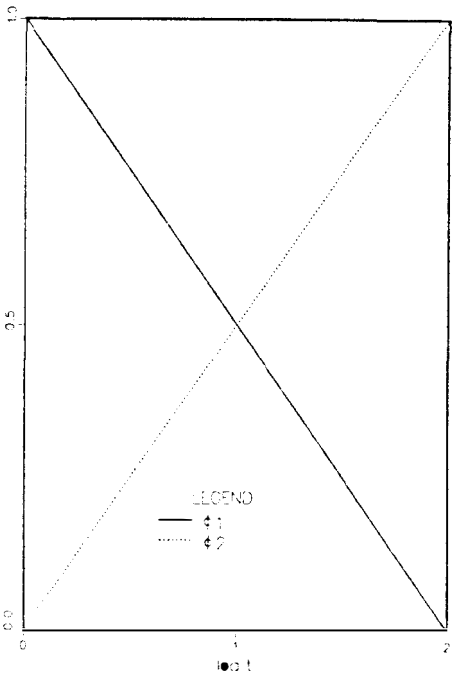

Figure 4.1: Weighting factors used to smooth out knots of piecewise polynomials, generated for fitting the curve of relaxation modulus E

In view of the noticeable variations in the curvatures of $E(t)$, it was divided into four parts. Each part was expressed separately by a third order polynomial employing a least square error fit. The distinct parts were overlapped smoothly over a range of two decades of times (namely two units along the $\log t$ scale). This smooth joining was accomplished by forming $E(t)$ within the overlapping range as a weighted average of the distinct parts as follows:

$$E(t)_{avg} = E_1(t)\phi_1(t) + E_2(t)\phi_2(t) \tag{4.8}$$

where ϕ_1 and ϕ_2 are the weighting factors, shown in Figure 4.1. Similar operations were made for E' and E'' with the derivatives of the piecewise polynomials obtained for E, resulting in the continuous and smooth functional values of E, E' and E'' through the curves. The resulting $E(t)$ vs. $\log t$, obtained from above above process, is shown in Figure 4.2.

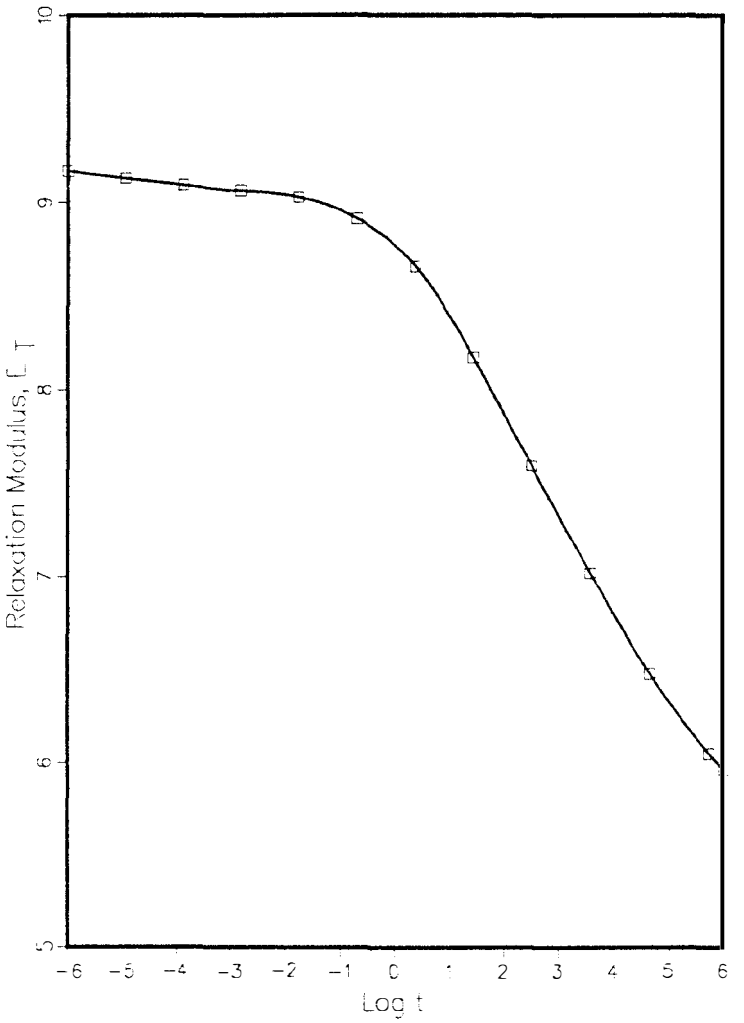

Figure 4.2: The relaxation modulus $E(t)$ vs. $\log t$, reduced from experimental data (Ref. [1]).

It turned out that for the realistic cooling times, namely $t_F \leq 1000\,min.$ all the required values of $E(t)$ were provided by the actual experimental data and there was no need for extrapolation. However, we note that while Figure 1.4 indicates a three-fold increase in compliance, the modulus in Figure 4.2 drops by only half that amount over the same range of time. This disparity indicates that in the case of *APC-2* the formation of a master curve through a horizontal shifting of isothermal data is remarkably inadequate and that factors like vertical shift and nonlinear effects must be considered in both formulation and computations.

4.3.2 Horizontal and Vertical shifts

The horizontal shift factor $a_T(T)$ was determined from a set of values which were obtained by Xinran [1] for the transverse compliance master curve. These values corresponded to the following sets of paired values: $\theta_1 = 3.3$, $\log a_{T_1} = 6.0$; $\theta_2 = 2.69$, $\log a_{T_2} = 2.5$; $\theta_3 = 2.1$, $\log a_{T_3} = -8$; and $\theta_4 = 1.6$, $\log a_{T_4} = -10$; where $\theta_i = \frac{1000}{273.1+T_i\,^0C}$. Intermediate values were obtained by linear interpolation. The results are plotted in Figure 4.3. Note that, for temperatures above 202^0C, Figure 4.3 contains values which were extrapolated beyond the experimentally reduced results. These values are shown by dashed lines.

Since the experimental data points are reduced on plots which have $\log a_T$ as an ordinate and $k = \frac{1000}{273.1+T}$ for an abscisa, it was necessary to employ the following relations to express the first and second derivatives of $a_T(T)$ with respect to T;

$$\frac{da_T}{dT} = \ln(10)\frac{\log a_T}{dK}\frac{dK}{dT}a_T \tag{4.9}$$

$$\frac{d^2a_T}{dT^2} = \ln(10)\frac{\log a_T}{dK}a_T\left(\frac{d^2K}{dT^2} + \ln(10)(\frac{dK}{dT})^2\frac{d\log a_T}{dK}\right) \tag{4.10}$$

where

$$\frac{dK}{dT} = -\frac{1000}{(273.3+T)^2}$$

and

$$\frac{d^2K}{dT^2} = \frac{2000}{(273.3+T)^3}$$

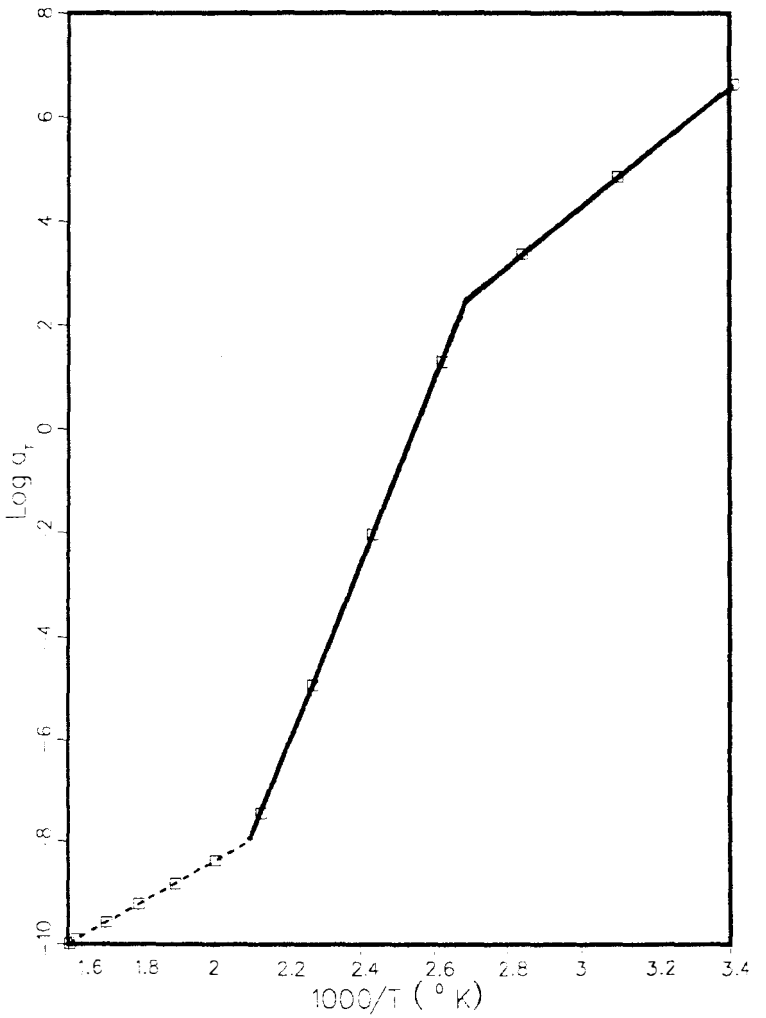

Figure 4.3: Horizontal shift factor for transverse compliance of *APC-2*, reduced from the experimental data (Ref. [1])

and $d\log a_T/dK$ is the slope of lines in Figure 4.3.

Furthermore, it can be readily shown that the components of the vertical shift factors for relaxation modulus $v_1(T)$ and $v_2(T)$ are the reciprocals of the opposite components of the vertical shift factor of the creep compliance g_1 and g_2 as follows:

$$v_1 = \frac{1}{g_2}$$
$$v_2 = \frac{1}{g_1}$$

Since Reference [1] provides only the product $g(T) = g_1(T)g_2(T)$, $v(T)$ is taken to be $g(T)^{-1}$. The vertical shift $g(T)$ shown in Figure 4.4 corresponded to $\log g(40) = 0.025$, $\log g(129.4) = 0.0031$, and $\log g(200) = -0.21$ with linear interpolation for intermediate values. For our computational purposes, the vertical shift factor at above $180\ ^0C$, shown by the dashed line in Figure 4.4, was extrapolated from the original data due to the lack of experimental data.

4.3.3 Nonlinear Parameter a_σ

Nonlinear viscoelastic effects were considered in the present calculations by means of stress-affected horizontal shift factor function ($a(T;\sigma)$) expressed in a product form as follows:

$$a(T;\sigma) = a_T(T)a_\sigma(\sigma;T) \tag{4.11}$$

where a_T is horizontal shift factor shown in Figure 4.3. The function $a_\sigma(\sigma)$ is expressed in the form given in Equation 1.28. The nonlinear function a_σ, shown in Figure 4.5, is plotted in relation to the uniaxial stress σ within the matrix. As shown in Figures 1.10 and 1.11 both the parameters $\tilde{\alpha}$ and $\tilde{\tau}$ depend upon temperature. For computational purposes, the parameter $\tilde{\alpha} = \tilde{\alpha}(T)$ was expressed by spline function fit of data points corresponding to $\tilde{\alpha}(24.41) = 0.118$, $\tilde{\alpha}(44.26) = 0.118$, $\tilde{\alpha}(74,41) = 0.118$, $\tilde{\alpha}(91.32) = 0.146$, $\tilde{\alpha}(103.1) = 0.171$ and $\tilde{\alpha}(120) = 0.239$. Xinran's experimental data, shown in Figure 1.10, indicated a substantial increase of $\tilde{\alpha}$ with temperature. However, the data are restricted to

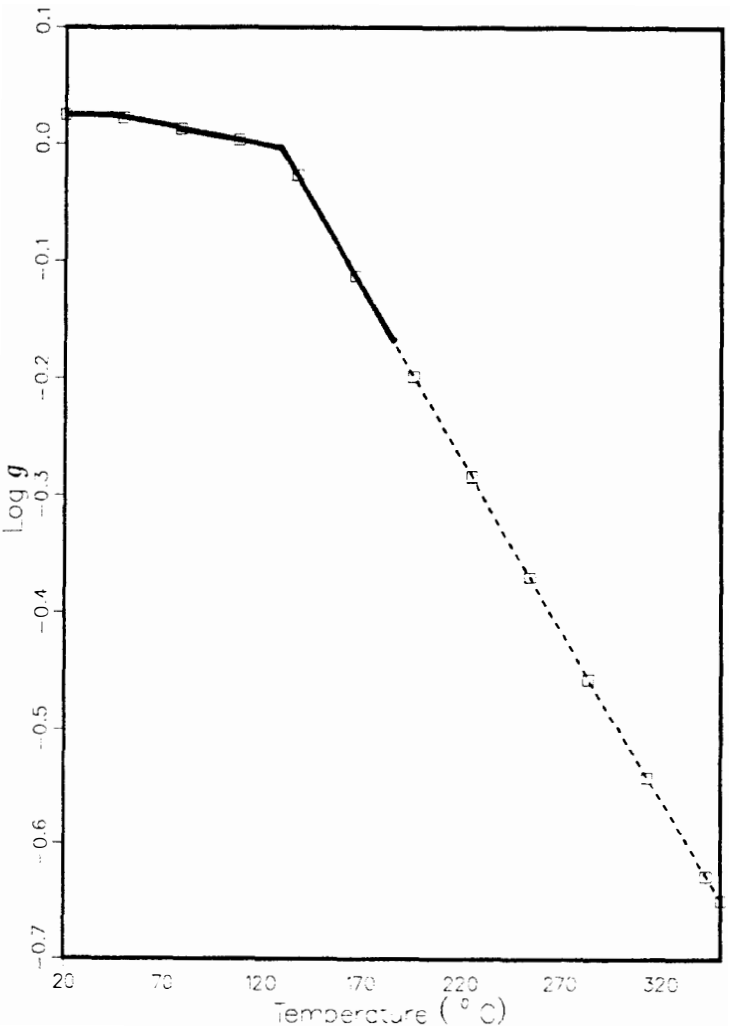

Figure 4.4: Vertical shift factor for transverse compliance of *APC-2*, reduced from the experimental data (Ref. [1]).

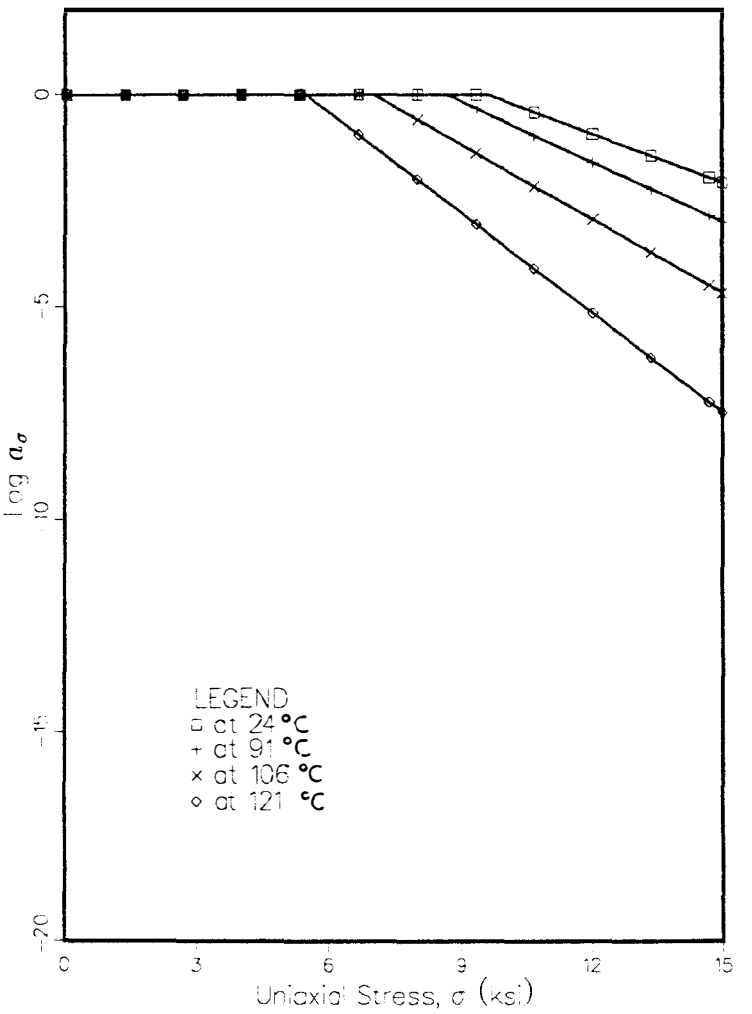

Figure 4.5: Nonlinear function a_σ vs. uniaxial stress, reduced from the experimental data (Ref. [1]).

temperatures up to 120^0C. For our computational purposes it was necessary to extrapolate $\tilde{\alpha}$ at temperatures above 120^0C. This somewhat arbitrary extrapolation is shown by the dashed line in Figure 4.6.

The parameter $\tilde{\tau}(T)$ is the nonlinear threshold of τ. It is shown in Figure 1.11 that Xinran's data for $\tilde{\tau}(T)$ followed two straight lines at below and above 91^0C. For our computational purposes $\tilde{\tau}(T)$ was expressed by a spline function fit through the data points $\tilde{\tau}(20) = 31.42$, $\tilde{\tau}(26) = 31.3$, $\tilde{\tau}(38) = 31.03$, $\tilde{\tau}(50) = 30.8$, $\tilde{\tau}(86.2) = 30$, $\tilde{\tau}(108) = 22.16$, $\tilde{\tau}(114) = 20$, and $\tilde{\tau}(120) = 17.86$. Beyond the experimentally available information, i.e. for T above 120^0C $\tilde{\tau}(T)$ was assumed to be constant up to $T = 300^0C$. This arbitrary extrapolation is shown by the dashed line in Figure 4.7.

4.3.4 Coefficients of Thermal Expansion

In accordance with Reference [16], the coefficients of thermal expansion were taken to be $\alpha_L = 0.5 \times 10^{-6}/1\ ^0C$, $\alpha_T = 30.0 \times 10^{-6}/1\ ^0C$, for $T < 125\ ^0C$ and $\alpha_L = 1.0 \times 10^{-6}/1\ ^0C$, $\alpha_T = 75 \times 10^{-6}/1\ ^0C$, for $125^0C < T < 300^0C$. To avoid non-essential computational difficulties at $T = 125\ ^0C$, smooth transitions [4] were assumed in values of α_L and α_T to occur over $124^0C < T < 126^0C$. The coefficients of thermal expansion in 0^0 direction and 90^0 direction are shown in Figure 4.8 and 4.9 respectively.

4.4 The Numerical Evaluation of Optimal Paths

4.4.1 Algorithm for the Iterative scheme

The optimal temperature path was obtained numerically and iteratively, by means of a special-purpose, custom-made computational subroutine. Guided by the considerations and results presented in Chapter 3, the steps employed in the computational procedure were restated . These are:

[4] The function $aT^3 + bT + c$ with $d\alpha/dT = 0$ at $T = 124^0C$ and 126^0C was used to provide slope continuity at the transition points.

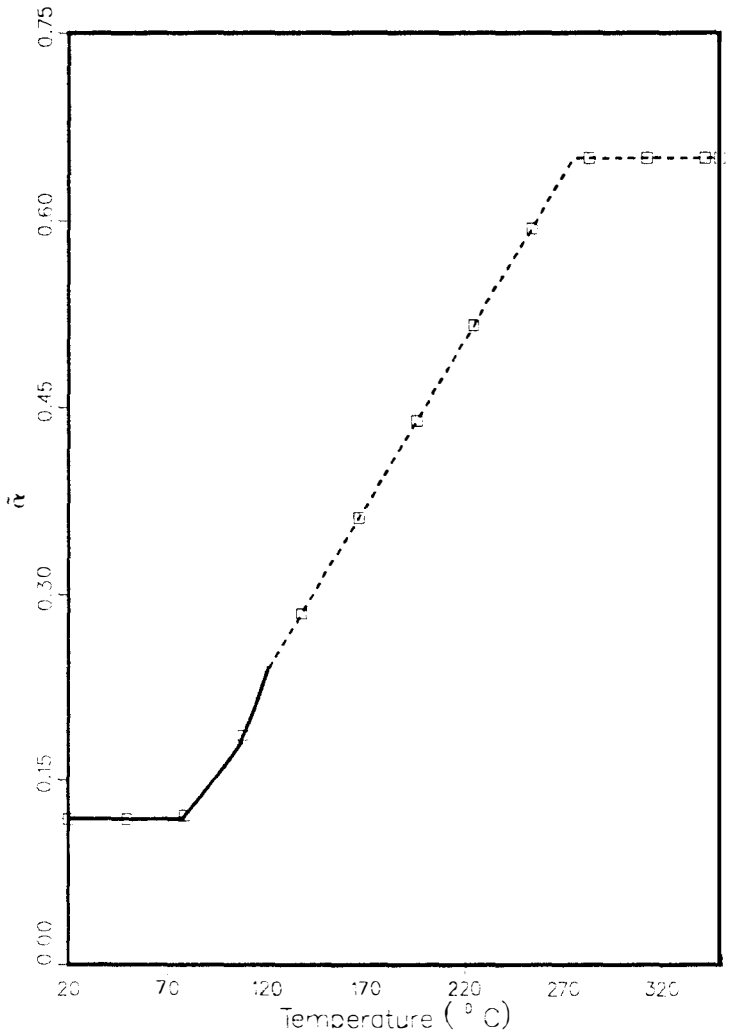

Figure 4.6: Nonlinear function $\tilde{\alpha}$, reduced from the experimental data (Ref. [1]) and extended by extrapolation (extrapolated values shown in dashed line).

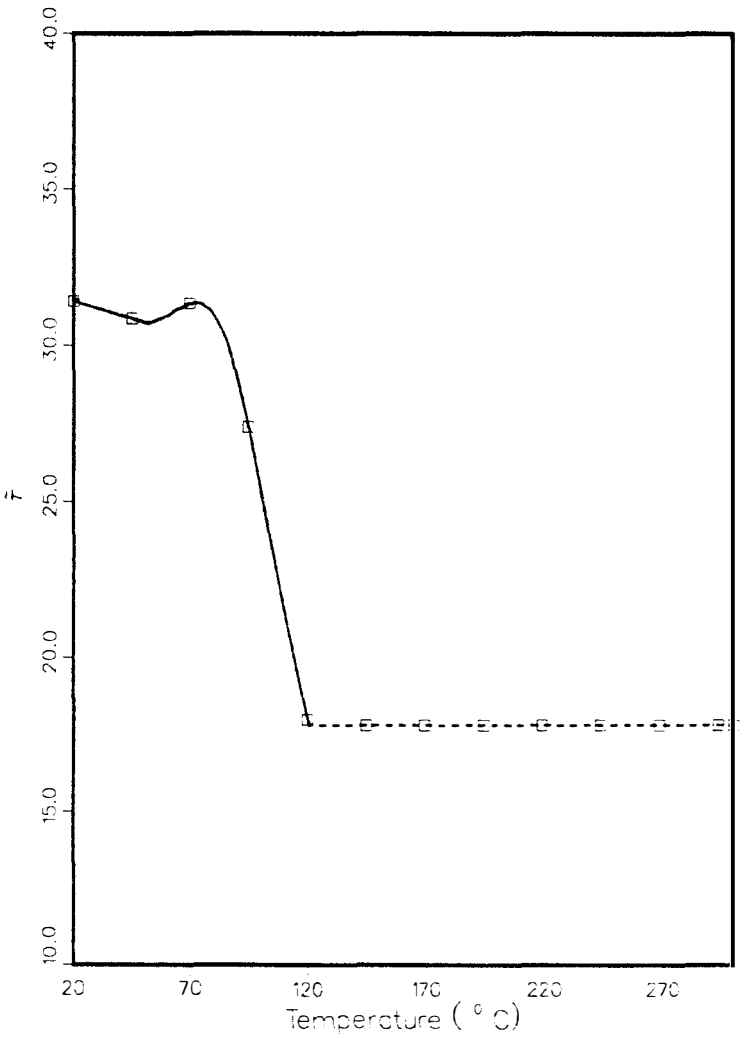

Figure 4.7: Nonlinear function $\tilde{\tau}$, reduced from the experimental data (Ref. [1]) and extended by extrapolation (extrapolated values shown in dashed line).

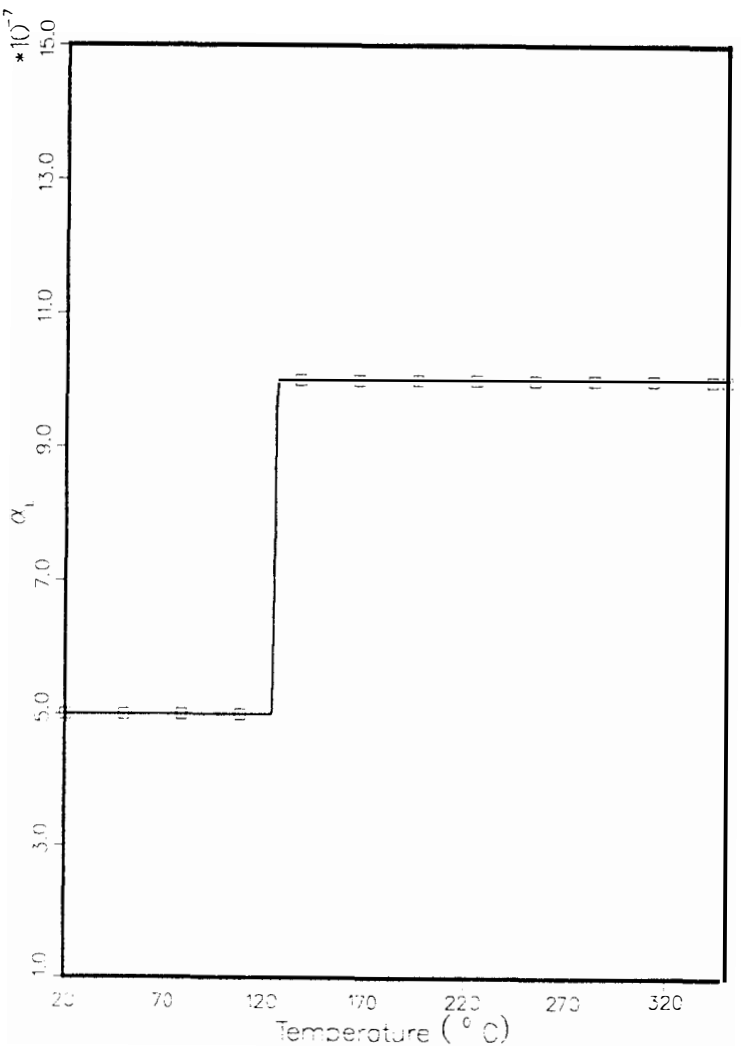

Figure 4.8: Thermal expansion coefficients in 0^0 fiber direction, reduced from the experimental data (Ref. [1]).

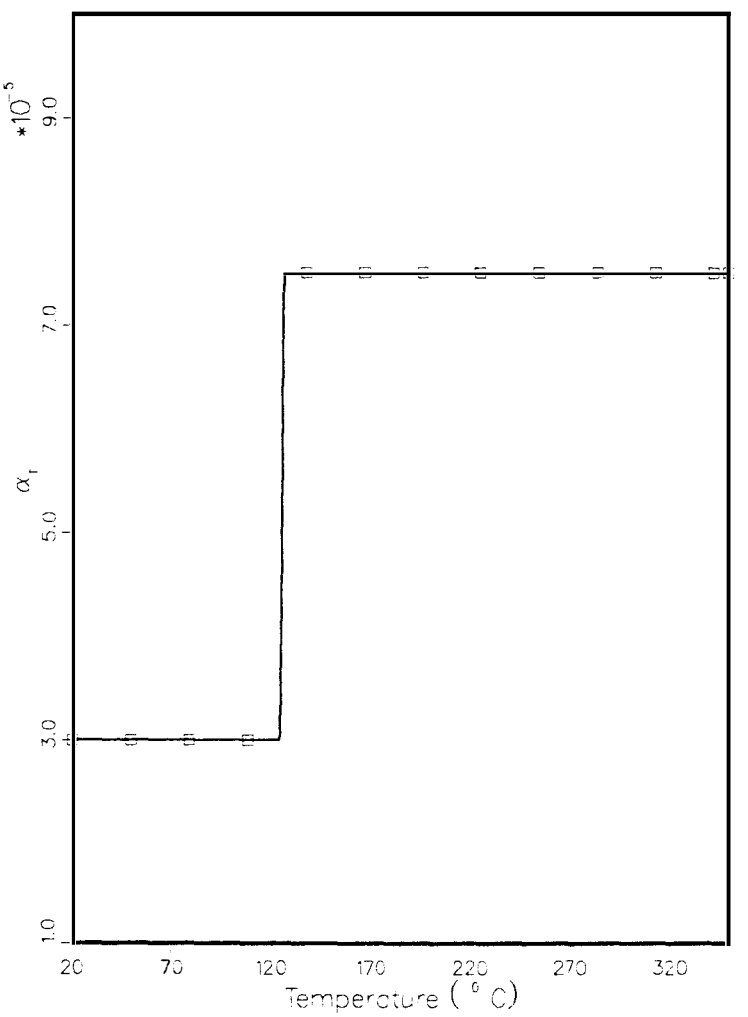

Figure 4.9: Thermal expansion coefficients in 90^0 fiber direction, reduced from the experimental data (Ref. [1]).

Step 1 Consider linear viscoelastic behavior, with response functions $E(t)$, α, $\nu(T)$ and $a(T)$ determined from reduced experimental data and with prescribed T_I, T_F and t_f. Employ Equation 2.26 to determine $T_0 = T(o^+)$ and solve Equation 2.27 numerically to obtain $T_\Omega(t)$, where $T_\Omega(t_f^-)$ is chosen iteratively until a solution of Equation 2.27 yields $T_\Omega(o^+) = T_0$. denote $T_\Omega(t)$ by $T_\Omega^{(0)}(t)$.

Step 2 Employ Equation 2.5 to compute the residual thermal stress $\sigma(t)$, denoted by $\sigma^{(0)}(t)$, due to cool-down along $T_\Omega^{(0)}(t)$.

Step 3 Relate $\sigma_t^{(0)}$ to $T_\sigma^{(\Omega}(t)$ to form $\sigma^{(0)} = F^{(0)}(T^{(0)})$.

Step 4 Consider the nonlinear shift factor function $a(T;\sigma)$ which corresponds to the reduced data for *APC-2*. Along the linear optimal path $a(T^{(0)}; F^{(0)}(T^{(0)}))$ can be expressed as $a^{(1)}(T)$.

Step 5 Repeat steps 1,2,3, and 4 with $a(T)$ replaced by $a^{(1)}(T)$ to obtain $T_0^{(1)}$ with $T_\Omega^{(1)}(t)$, $\sigma^{(1)}(t)$, $\sigma^{(1)} = F^{(1)}(T^{(1)})$ and $a^{(2)}(T)$, respectively.

Continue until the attainment of a prescribed convergence, say $T_\Omega^{(n+1)}(t) = T_\Omega^{(n)}(t) + \epsilon(t)$, where $|\epsilon(t)|$ is less than a given tolerance.

4.4.2 Computational Details

In the nonlinear iterative optimization scheme, the first iteration consisted of the determination of the linear viscoelastic optimal path. The computations employed Equations 2.26 and 2.27 to obtain the optimal path $T_\Omega(t)$ numerically, according to the previously outlined procedure in Section 3.3.2.

In accordance with step 2 of the computational Algorithm, $\sigma(t)$ are evaluated along the optimal path $T_\Omega(t)$ and correlated to the optimal path $T_\Omega(t)$ to form $F(T)$. Subsequently, it is necessary to form $a(T,\sigma) = a(T, F(T)) = \hat{a}(T)$ associated with $T_\Omega(t)$. The path $T_\Omega(t)$ vs. t, denoted by "A," is sketched in Figure 4.10 with initial and final values $(T_0)_A$ and

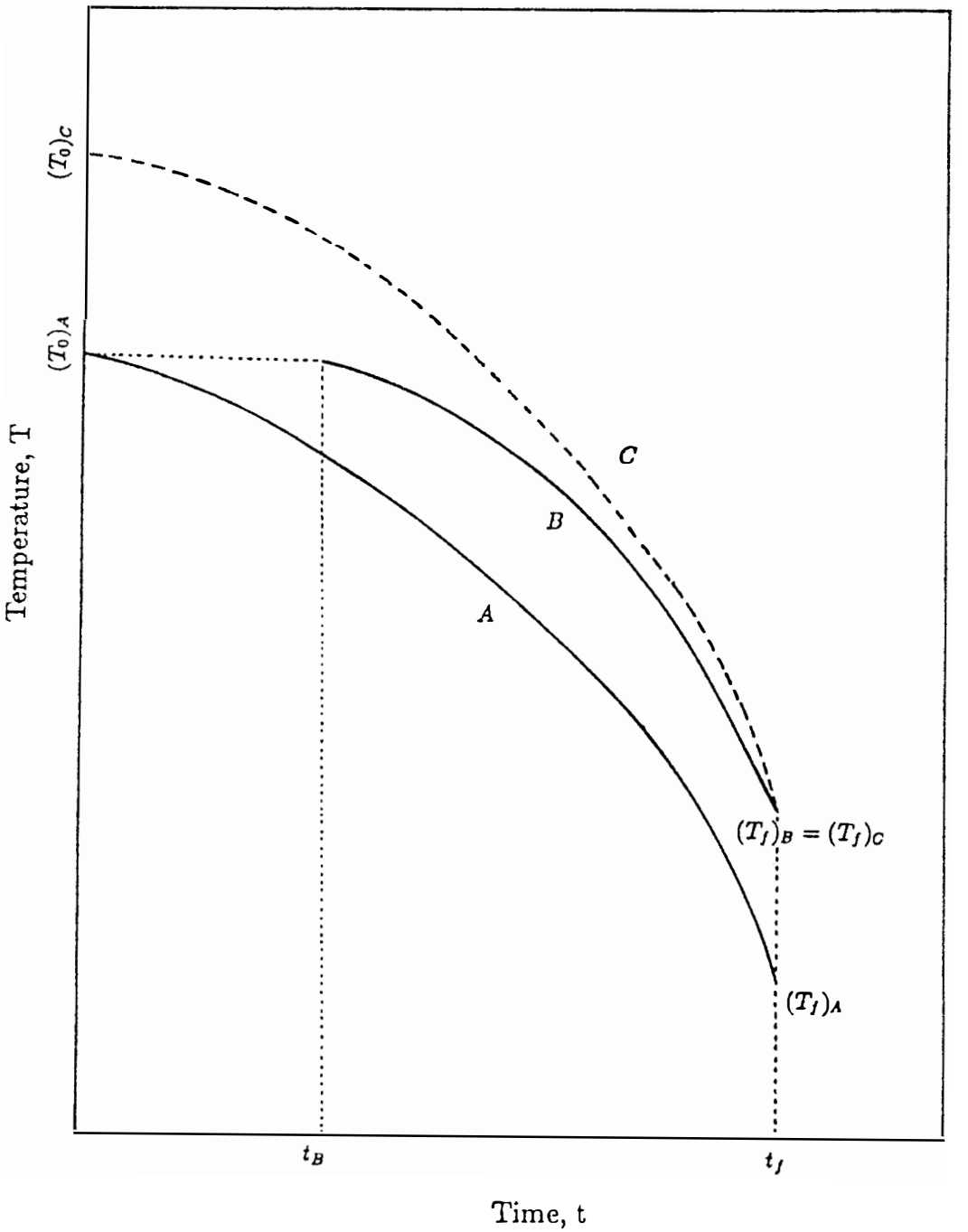

Figure 4.10: The construction of the "pseudo" optimal temperature path

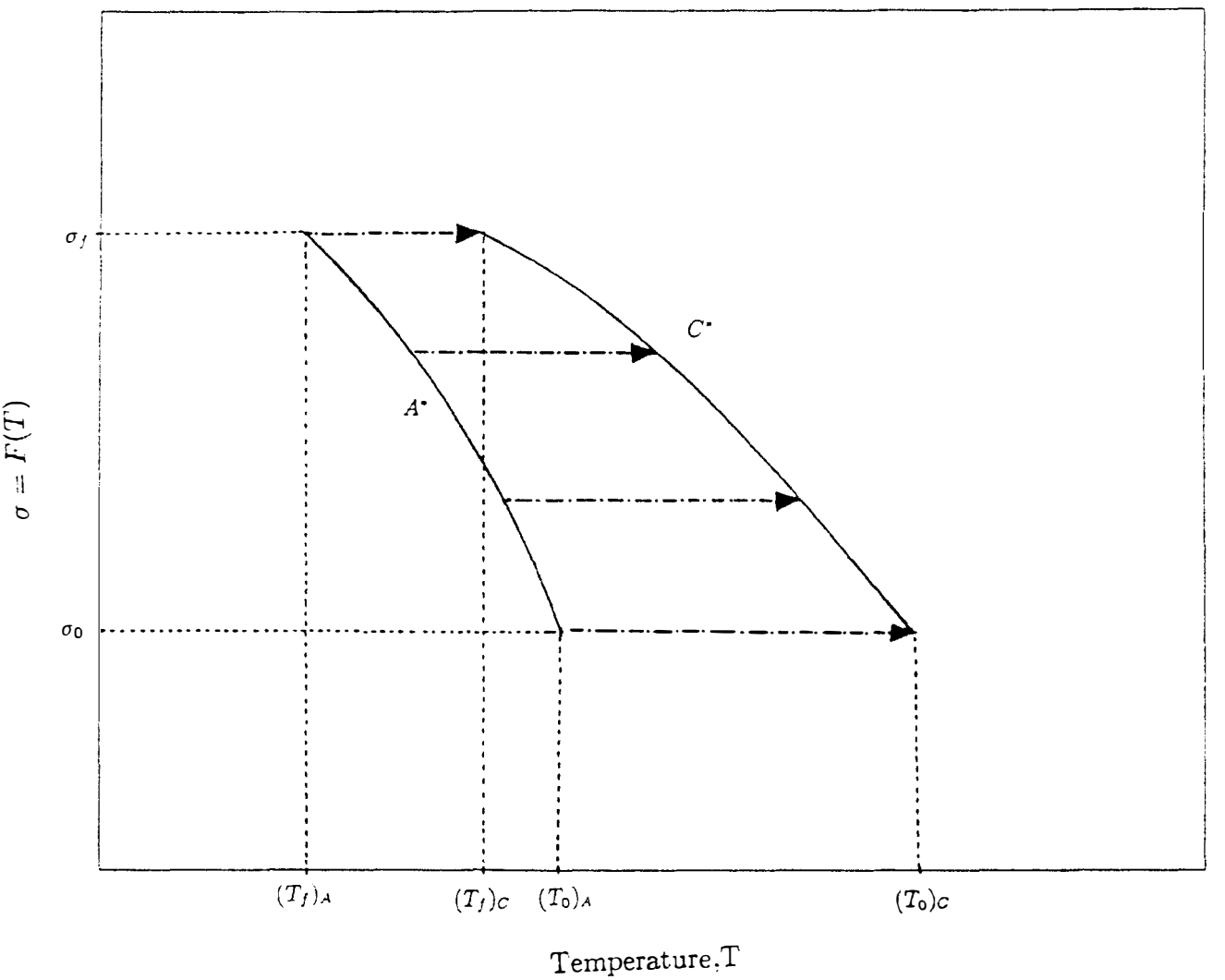

Figure 4.11: The function $F(T)$ modified to cover the extended temperature region

$(T_f)_A$, while the corresponding $F(T)$ vs. T, denoted by A^*, is sketched in Figure 4.11. It is important to recognize that T_Ω (path A in Figure 4.10) and $F(T)$ (path A^* in Figure 4.11) are defined only over the range $(T_f)_A < T < (T_0)_A$. This limited range of definition confines our ability to express of $\hat{a}(T)$ to within the same range of temperatures.

Turning to the next iteration, it is necessary to initiate the computation by selecting a new guess value of T_f, say $(T_f)_B$. Unfortunately, it becomes impossible to construct a new optimal path from $(T_f)_B$ by means of Equation 2.27 with the above $\hat{a}(T)$. The cause of the difficulty is that, for $(T_f)_B < (T_f)_A$, $\hat{a}((T_f)_B)$ is not defined and, for $(T_f)_B > (T_f)_A$, it is impossible to construct an optimal path by means of Equation 2.27 beyond $(T_0)_A$. A sketch of such a path, denoted by B, is shown in Figure 4.10. The path B cannot be established for times $t < t_B$.

In order to surmount the above obstacle, Equation 2.27 is now temporarily abandoned and, still using $T_\Omega(t)$ and $\sigma(t)$ at hand, functions $F(T)$ and "pseudo" optimal paths $T(t)$ are constructed, which enable us to proceed with the current iteration.

For that purpose, a guess value of T_f is selected , *accompanied by a guess value of* T_0. These values, denoted by $(T_f)_C$ and $(T_0)_C$ in Figure 4.10, are then connected by a continuous path $T(t)$, denoted by C in Figure 4.10. The intermediate temperatures $(T_i)_C$ along C are related to $(T_0)_A$, $(T_0)_C$, $(T_f)_A$, $(T_f)_C$ and $(T_i)_A$ according to

$$(T_i)_C = (T_i)_A + \Delta T_i \tag{4.12}$$

where the intermediate values of ΔT_i were determined from the following relationship based on similar triangles

$$\frac{\Delta T_0 - \Delta T_f}{t_0 - t_f} = \frac{\Delta T_i - \Delta T_f}{t_i - t_f} \tag{4.13}$$

In Equation 4.13 ΔT_0 and ΔT_f refer to the differences between the end values of paths "A" and "C" Figure 4.10, namely

$$\Delta T_0 = (T_0)_C - (T_0)_A, \quad \Delta T_f = (T_f)_C - (T_f)_A$$

A modified function $F(T)$ is now constructed by relating the values of $\sigma(t)$, which were computed along $T_\Omega(t)$ of the previous iteration (which are available to us) with $T(t)$ given by Equation 4.12. Such a function $F(T)$, sketched by the curve C^* in Figure 4.11, is defined over the appropritate range $(T_f)_C < T < (T_0)_C$ which, in turn, enables the establishment of $\hat{a}(T) = a(T, F(T))$ over the same range. With the above, ad hoc, values of $\hat{a}(T)$, and for the above selected guess value of T_f, Equation 2.27 is now employed to generate an optimal path $T_\Omega(t)$. Obviously, this $T_\Omega(t)$ cannot rise above $(T_0)_C$ since $\hat{a}(T)$ is undefined for $T > (T_0)_C$. Therefore, if such a circumstance is forced by Equation 2.27, the guess value of $(T_0)_C$ is abandoned in favor of a revised, higher value. Subsequent guess values of $(T_0)_C$ are introduced iteratively until $(T_0)_C$ matches $T_\Omega(0)$ to within a given tolerance. When this match is achieved, the residual stress $\sigma(t_f^+)$ is computed according to Equation 4.3, due to cool-down along the path $T_\Omega(t)$ which ends at the above guess value of T_f and is generated through Equation 2.27 with the abovementioned, ad hoc, $\hat{a}(T)$.

At this stage another guess value of T_f is selected and the procedure is repeated, obtaining a corresponding value for the residual stress $\sigma(t_f^+)$.

In such manner a sequence of values $\sigma(t_f^+)$ are obtained corresponding to various guess values T_f. Note that all these values are derived from a procedure that uses stresses $\sigma(t)$ along the optimal path of the previous iteration as "guide posts" for the formation of $F(T)$ and $\hat{a}(T)$. No up-dating of these "guide posts" is affected during the current process.

It is now necessary to search for the value of T_f which yields a minimal $\sigma(t_f^+)$. This search was performed by means of the IMSL subroutine "UMCGF," which uses a conjugate gradient algorithm and gradients evaluated through finite differences to locate minima of functions.

The conjugate gradient method is closely related to the method of steepest descent. Both methods employ gradients to locate the minima of functions using sequences of linear searches along successive directions until meeting a prescribed convergence condition. In

the method of steepest descent, the search from a current point is carried out along the negative gradient at that point. In contrast, in the conjugate gradient method the search is carried out along mutually conjugate or orthogonal directions, which tends to yield better results for searching for a optimal point. It is, however, noted that for a one-dimensional minimization problem such as our present case, the same results may be expected from all methods, since all would employ identical search directions.

In this manner the "best" T_f is selected as the end point for the current iteration. For this "best" value of T_f, $T_\Omega(t)$ and the corresponding stress $\sigma(t)$ along $T_\Omega(t)$ are computed. These two functions are correlated to form revised $F(T)$ and $\hat{a}(T)$. The revised, above-mentioned, values of $\sigma(t)$ and $F(T)$ form the "guide posts" for the new iteration. This completes the current iteration, which is now used as the beginning step for the following iteration. The procedure is repeated until attaining the prescribed convergence criteria, as expressed in Equation 3.14. Note that in the numerical scheme, as outlined just above, the use of Equation 2.26 was abandoned and the value of T_0 was determined by the search method, which was substituted in its place. As will be noted below, the evaluation of $F(T)$ and hence $\hat{a}(T)$ and , in particular $-\hat{a}'(T)$, near T_0 is not sufficiently reliable to permit the use of Equation 2.26.

The selection of nonuniform Δt_i, adopted for the numerical integration of Equation 2.27, was motivated by the results from the previous "prototype cases". In these prototypes the quantities of dT/dt varied most significantly over certain short time-intervals. Consequently, at each time step t_i the size of the time interval Δt_i was determined by regulating the temperature increment. It was necessary to regulate the temperature increment not only because dT/dt along the optimal path $T_\Omega(t)$ varied from extremly large to extremely small value over $0 < t < t_f$, but also because dT/dt changed very rapidly in relatively short time spans. On the other hand the time increment was also regulated to avoid undesirably long time intervals when dT/dt was extremely small. It was found that with an appropriate

selection of unequal time intervals, sufficient accuracy was obtained for all $T_\Omega(t)$ when the time interval t_f was divided into no more than 500 unequal subintervals.

The solution of Equation 2.27 is, in general, unstable numerically. The condition $dT/dt < 0$ is required if the temperature T continues to drop monotonically over the time interval $(0, t_f)$. This condition also serves as a requirement for the convergence of the T_Ω. Therefore, several corrections were necessary to avoid the computational instablity which occurred when this condition was violated. The solution of Equation 2.27 required the values of E', E'', a',a'', and v', which were evaluated by means of interpolating functions. However, inaccurate functional values were inevitable at certain locations for two reasons: (1) the interpolating functions were generated from a limited number of experimental data and (2) Equation 2.27 involved the evaluation of the first and second derivatives of functions. Thus certain types of interpolation errors as well as truncational errors were inevitable. On several occasions changes of the sign in dT/dt were encountered during the computations. These were corrected simply by replacing an "unacceptable" value of $(dT/dt)_i$ with $(dT/dt)_{i-1}$ of the previous time step, and continuing the computation accordingly. Numerous trial-and-error tests and the need for ongoing corrections required vigilant attention and judgement during the computation of T_Ω. This requirement for ongoing intervention in the computational process precluded its full automation and ruled out the use of the optimization routines developed as commercial software packages (e.g. IMSL and NAG). Nevertheless, it was possible to accomplish steps 1 and 2 of the computational algorithm in a fairly efficient manner.

The numerical correlation $\sigma = F(T)$ and its incorporation into the horizontal shift factor $a(T;\sigma) \rightarrow a(T;F(T)) \rightarrow \hat{a}(T)$, which is indicated in steps 3 and 4 in computational algorithm, required smoothing operations to provide resonable values for the derivatives of $\hat{a}(T)$ to be used in Equation 2.27. The interpolating function of $\sigma = F(T)$ was generated with the n sets of paired values (σ_i, T_i) obtained from the previous steps by means of

the IMSL routine "CSCON," a shape-preserving interpolation routine. The consequent derivative of $F(T)$ was also generated by means of IMSL routine "CSDER."

It is also noted that $F(T)$ varied significantly with T near $t = 0$, and ,moreover, the change of sign of $dF(T)/dT$ occurred due to the initial temperature drop in the immediate vicinity of $t = 0$. Insufficient number of corresponding sets of values (σ_i, T_i) to account for this fluctuation with sufficient accuracy resulted in an erroneous value for first derivative of $F(T)$ with respect to T at the vicinity of $t = 0$. Subsequently, additional intermediate values of σ_i were evaluated at refined time steps t_i until reliable first derivatives of $\sigma = F(T)$ were obtained near $t = 0$. These procedures resulted in reasonable and consistent values for $\hat{\alpha}(T)$ and all its various derivatives which were needed in the computational scheme for $T_\Omega(t)$.

It should be noted that there is no mathematically grounded proof that the iterated optimization scheme must converge, and obviously no proof of uniqueness. Nevertheless, it is interesting to note that for the *APC-2* data at hand, the iterative scheme converged after about five to six iterations.

Efforts to verify our results at least partially were made in two ways. The first attempt at validation was made by introducing several arbitrary, small disturbances in the optimal path and comparing the ensuing values of $\sigma(t_f)$. An upward or a downward perturbation in T_Ω was introduced at the selected locations of the optimal path T_Ω. A total of three such locations were examined along T_Ω - at early, intermediate, and late stages in the time interval $(0, t_f)$ respectively. In all cases the resulting values of $\sigma(t_f)$ exceeded the optimal value.

Next, continuous paths in the neighborhood of the optimal path T_Ω were examined, in contrast to the previous perturbations which occurred at localized neighborhoods of T_Ω. Several such neighboring temperature paths were constructed and the consequent values of $\sigma(t_f)$ were compared. In some circumstances slightly lower values of $\sigma(t_f^+)$ were obtained

for these neighboring temperature paths. However, the results attained in the present computations were deemed acceptable within the numerical accuracy.

4.4.3 Numerical Results

The reduced and extrapolated data, as expressed in Section 4.3. were utilized in the evaluation of optimal cooling paths for *APC-2* composite material. In the computations two cases of initial stress-free temperatures were employed, namely $T_I = 250^0 C$ and $300^0 C$. In all cases, a final cool-down temperature, T_R, was considered $30^0 C$ and a cooling time, t_f, $100\, min$. With those two sets of values, computations were performed for the following three thermorheological sub-cases:

case a the thermorheologically simple case ($v = v_1 = v_2 = 1$)

case b the thermorheologically complex case with $v = v_2$, and $v_1 = 1$.

case c the thermorheologically complex case with $v = v_1$, and $v_2 = 1$.

The results are shown in Figures 4.12 - 4.27. In all those figures the temperature T is in $^0 C$, the stress σ is in ksi, and the time t is in minutes. Results for optimal cool-down paths, $T_\Omega(t)$ vs. t, are shown in Figures 4.12 - 4.19 and results for stress histories along the optimal temperature paths, $\sigma(t)$ vs. t, are plotted in Figures 4.20 - 4.27.

To demonstrate the effects of nonlinearity, for each type of thermorhreological behavior, the linear and nonlinear optimal cooling paths are paired against each other in Figures 4.12 - 4.14 for $T_I = 250^0 C$ and in Figures 4.15 - 4.17 for $T_I = 300^0 C$. Figure 4.18 combines the results for optimal cool-down paths with initial stress-free temperature, $T_I = 250^0 C$, for the all three thermorheological sub-cases, the "simple" case ($v = v_1 = v_2 = 1$), and two "complex" cases ($v = v_2$, $v_1 = 1$ and $v = v_1$, $v_2 = 1$), as well as three analogous nonlinear cases with $a(T;\sigma)$. Analogous results are combined in Figure 4.19 for $T_I = 300^0 C$.

Table 4.1: Comparison of final stresses for *APC-2* composite laminate between linear and nonlinear cases

Description	Final stress			
	$T_I = 300^\circ C$		$T_I = 250^\circ C$	
	linear	nonlinear	linear	nonlinear
case a	13.146	12.390	10.670	10.45
case b	10.655	9.550	10.000	9.11
case c	12.752	11.728	10.354	10.09

The corresponding thermal stresses along the optimal paths are shown in Figures 4.20 - 4.22 for $T_I = 250^0C$ and in Figures 4.23 - 4.25 for $T_I = 300^0C$. Combined plots are shown in Figures 4.26 and 4.27.

For **case a**, the thermorheologically simple case, the major differences between the optimal temperature paths, as shown in Figures 4.12 and 4.15, was that the nonlinear temperature path resembled a stepwise pattern, while the linear case corresponded to more gradual temperature variation. For **case b**, the thermorheologically complex case with $v = v_2$, shown in Figures 4.13 and 4.16, both linear and nonlinear optimal temperature paths followed stepwise patterns, with the nonlinear paths falling below the linear paths. Figures 4.14 and 4.17 exhibit the optimal temperature paths for the **case c**. It is noted that linear optimal paths for the **case c** are identical with those of the **case a** since, in **case c**, the vertical shift factor is located entirely outside the superposition integral. However, the nonlinear optimal paths corresponding to **cases c and a** differed from each other since the vertical shift, though outside of the integral, depended now on stress.

Final residual thermal stresses at $t = t_f$ for various cases are shown in Table 4.1. Note that the highest final stress σ_f occurred in the linear **case a** which entails horizontal shift only, while the lowest final stress σ_f corresponded to the nonlinear **case b** which incorporates the entire vertical shift factor inside the superposition integral.

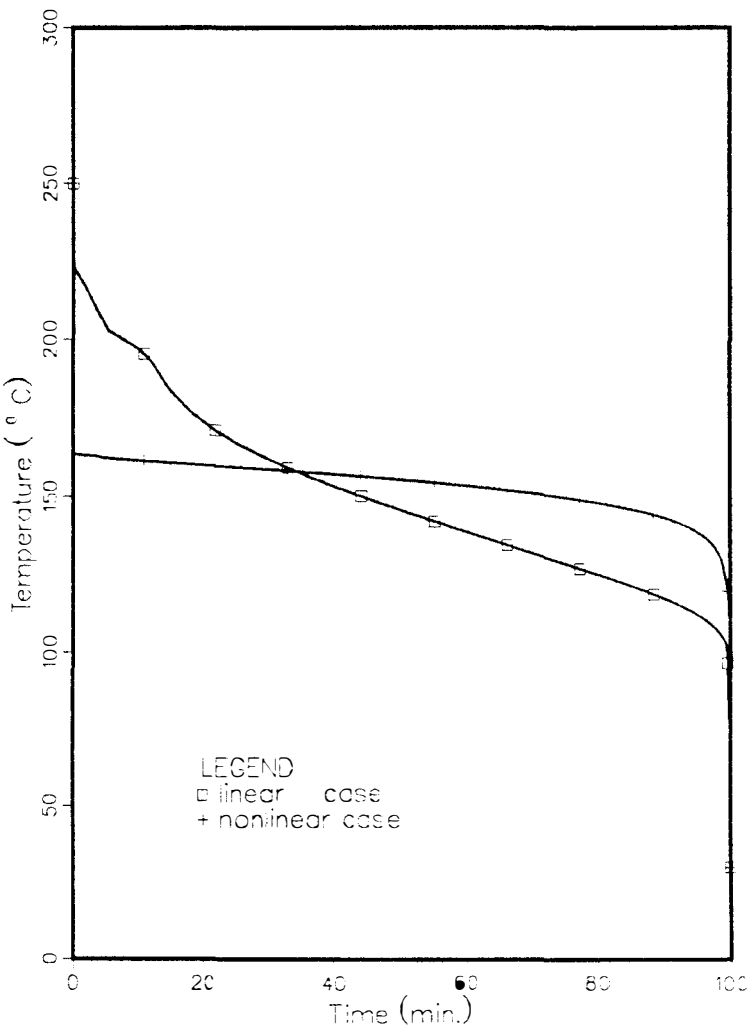

Figure 4.12: Optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$ - **case a**

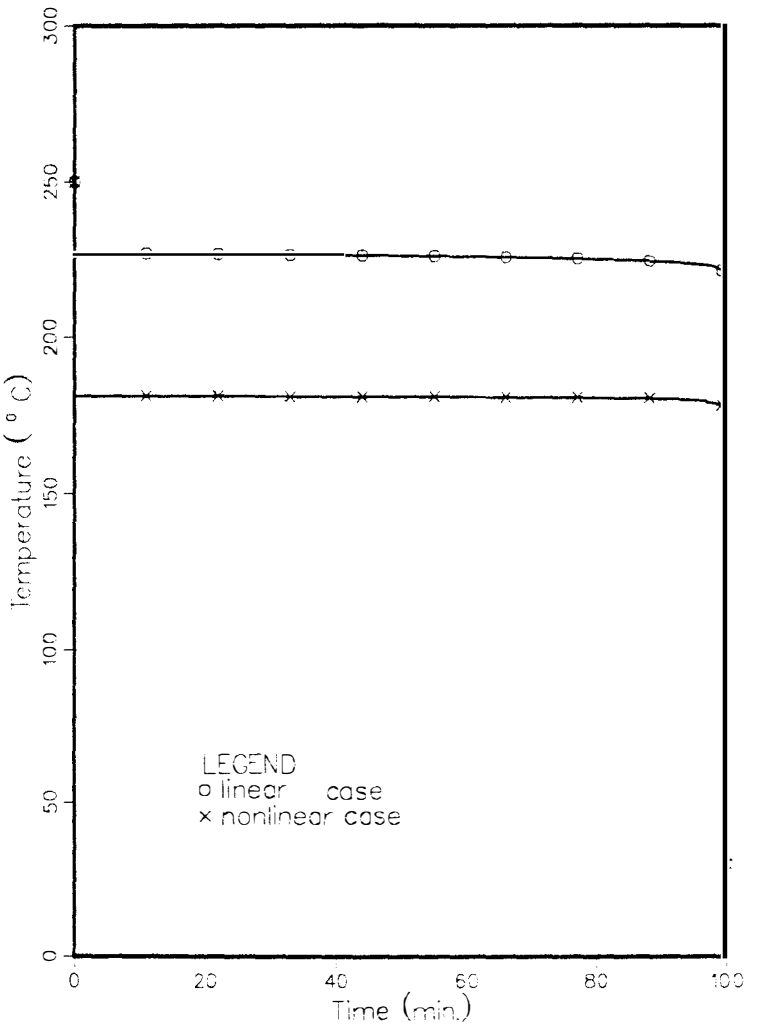

Figure 4.13: Optimal temperature paths for for *APC-2*, assuming $T_I = 250^0C$ - **case b**

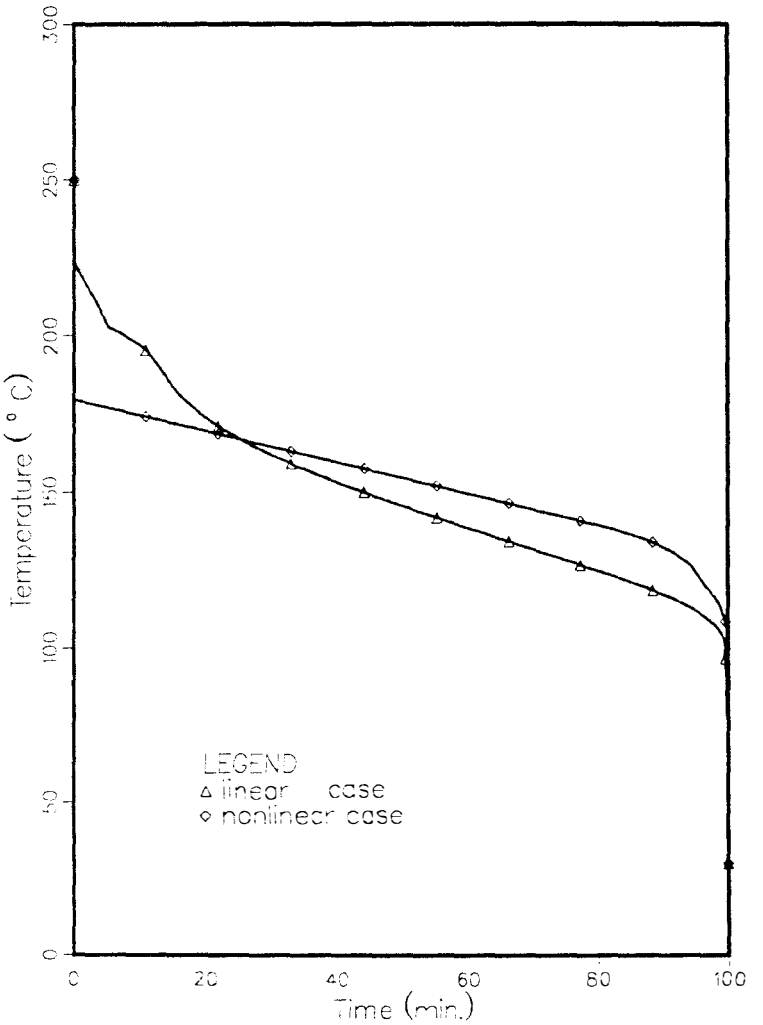

Figure 4.14: Optimal temperature paths for for *APC-2*. assuming $T_I = 250^0C$ - **case c**

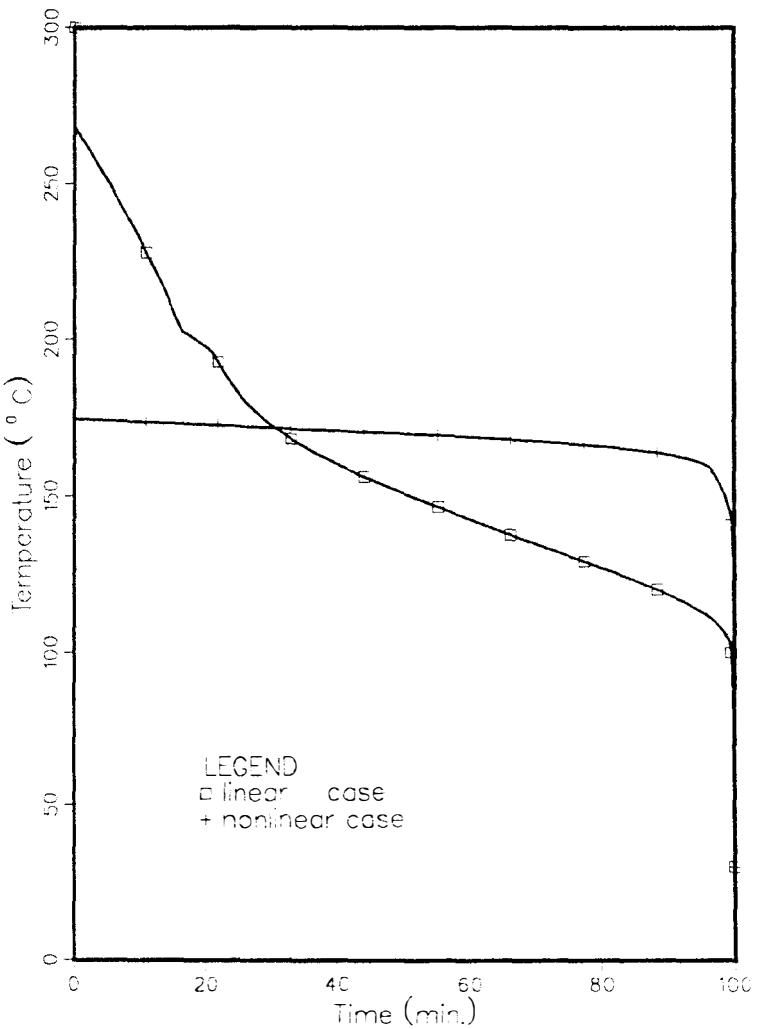

Figure 4.15: Optimal temperature paths for for *APC-2*, assuming $T_I = 300^0C$ - **case a**

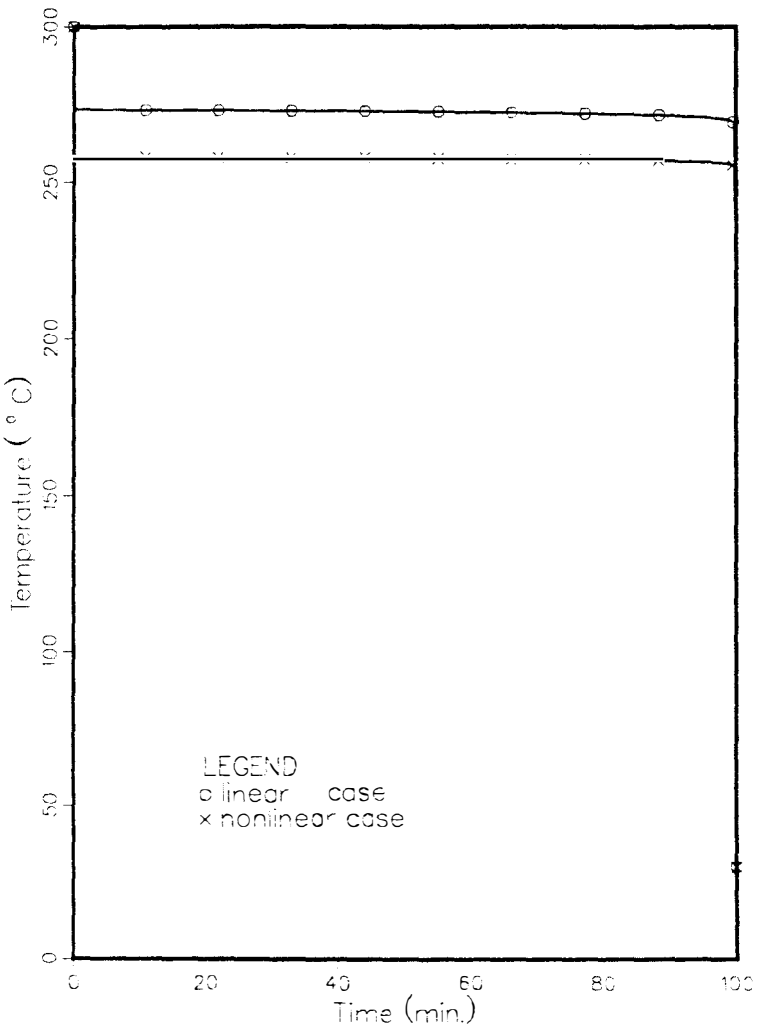

Figure 4.16: Optimal temperature paths for for *APC-2*, assuming $T_I = 300^0C$ - **case b**

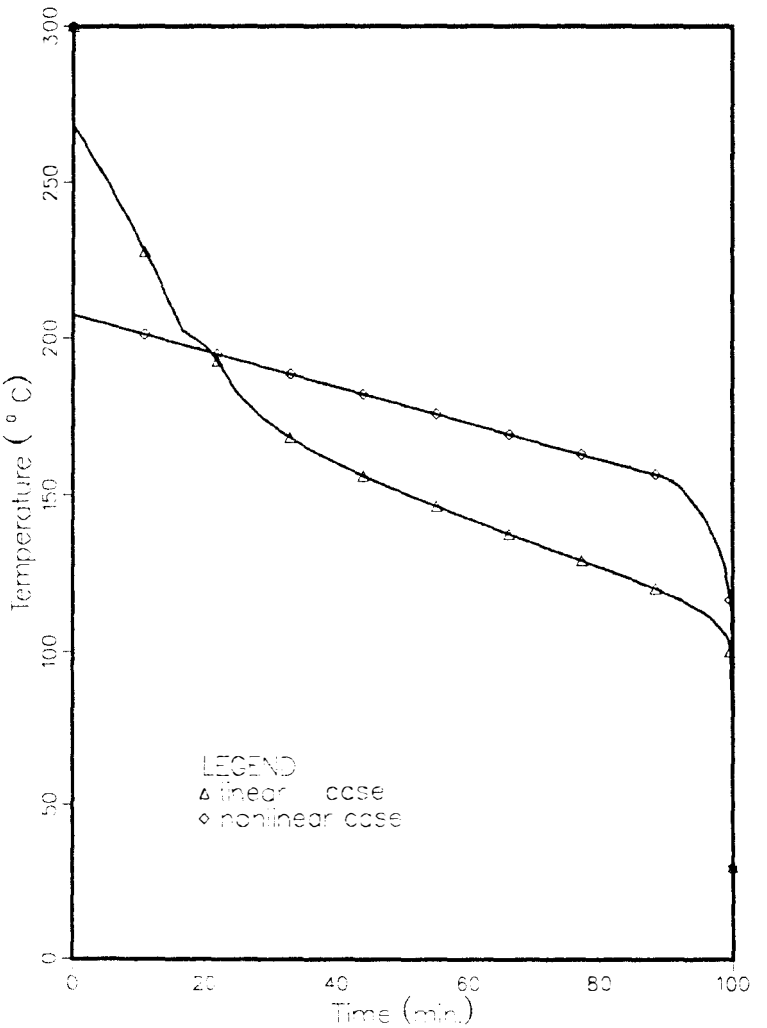

Figure 4.17: Optimal temperature paths for for *APC-2*, assuming $T_I = 300^0C$ - **case c**

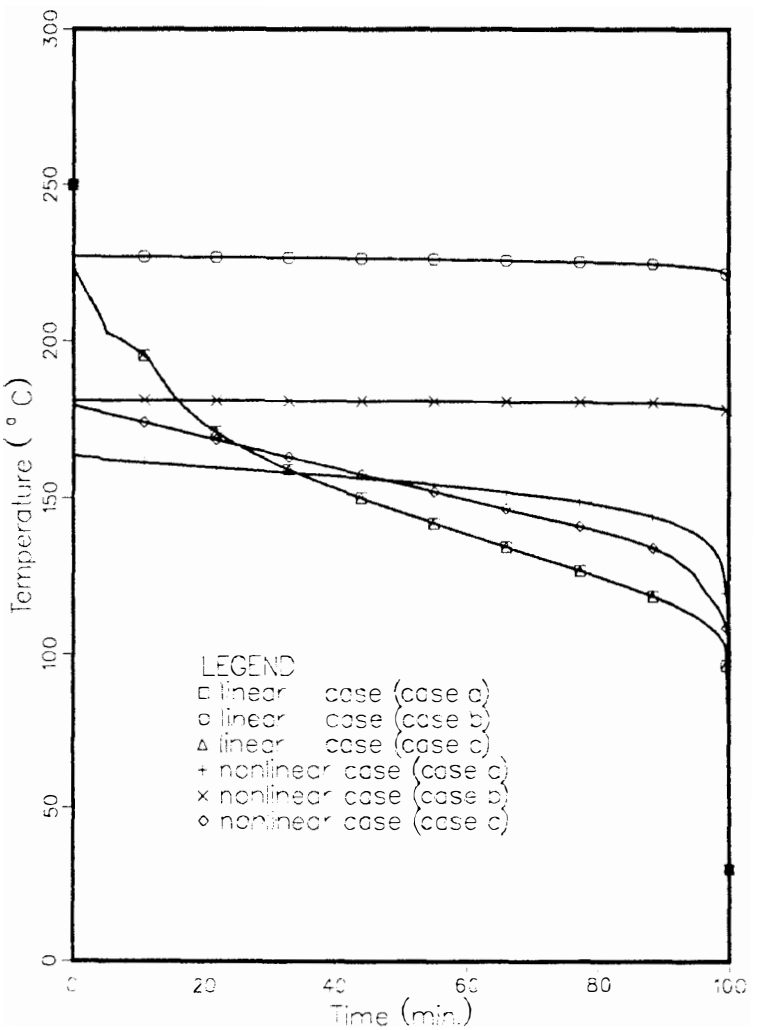

Figure 4.18: Optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$

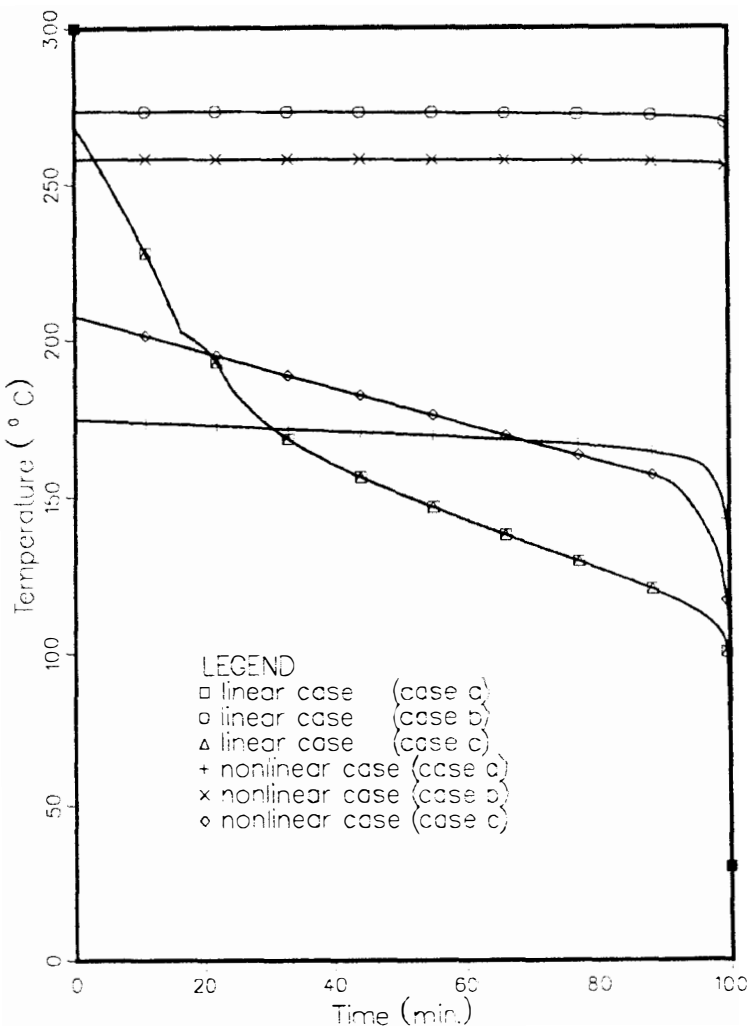

Figure 4.19: Optimal temperature paths for *APC-2*, assuming $T_I = 300^0C$

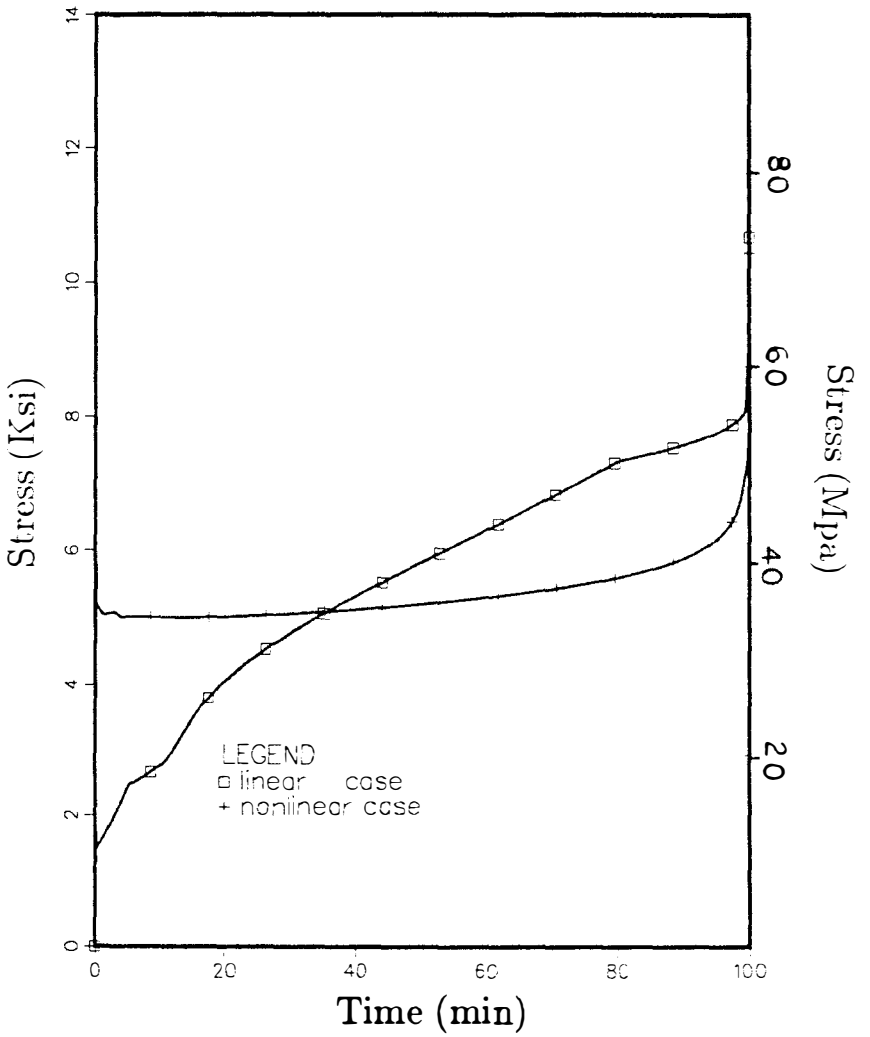

Figure 4.20: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$ - **case a**

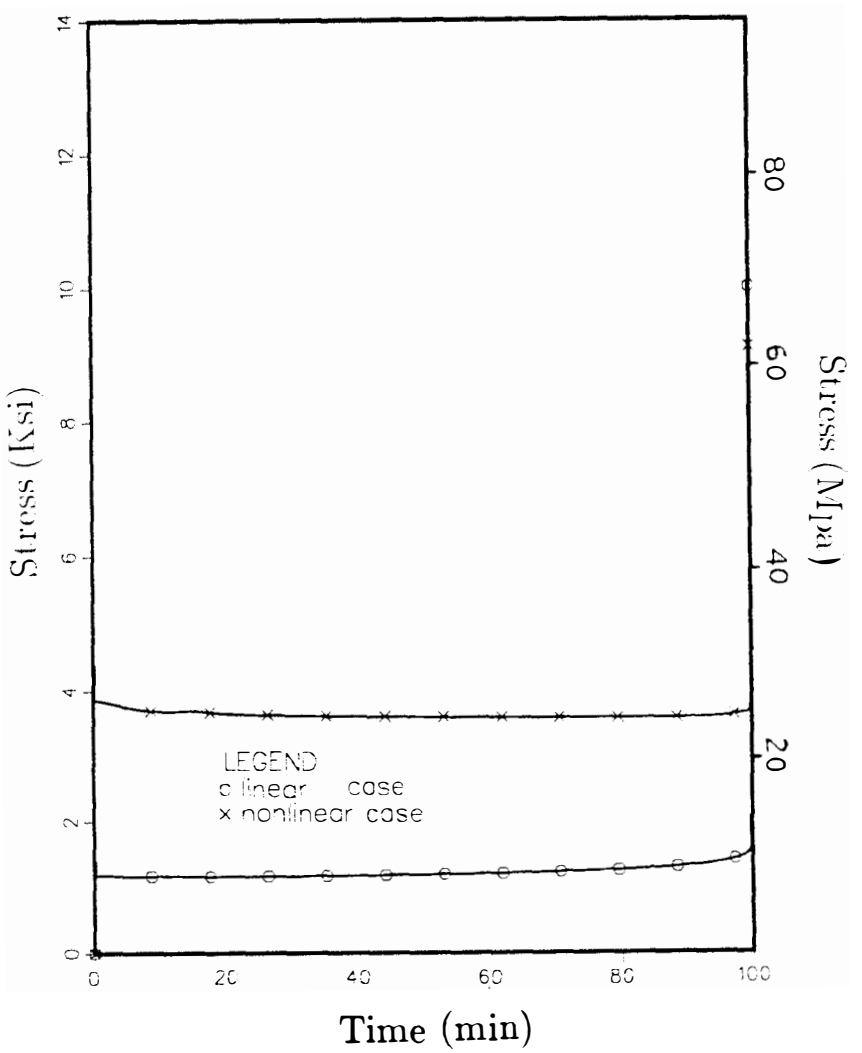

Figure 4.21: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$ - **case b**

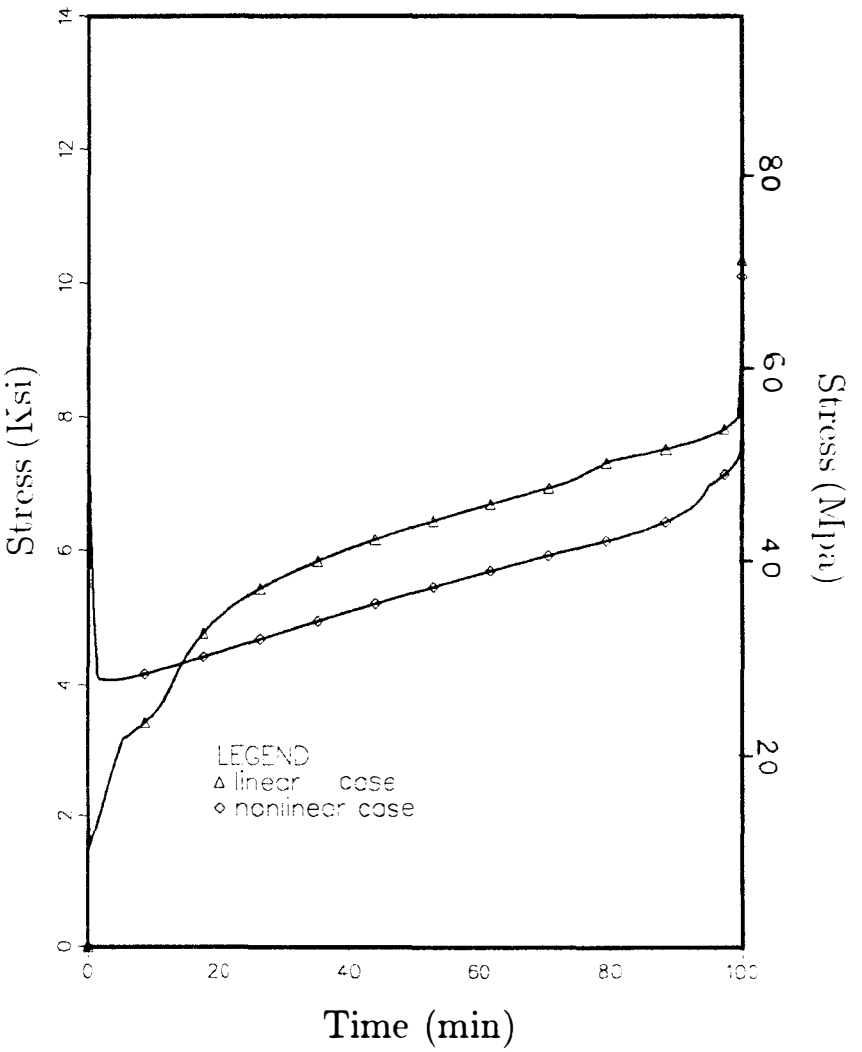

Figure 4.22: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$ - **case c**

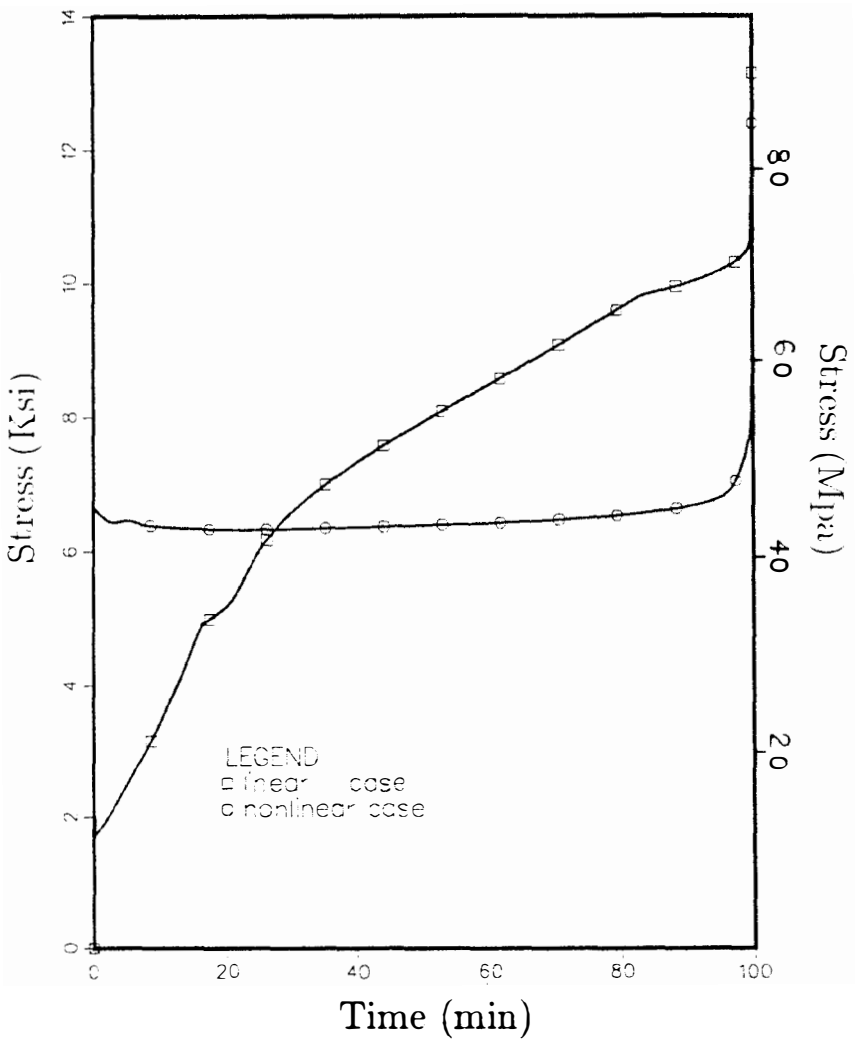

Figure 4.23: Stress histories along optimal temperature paths for *APC-2*, assuming $T_1 = 300^0C$ - **case a**

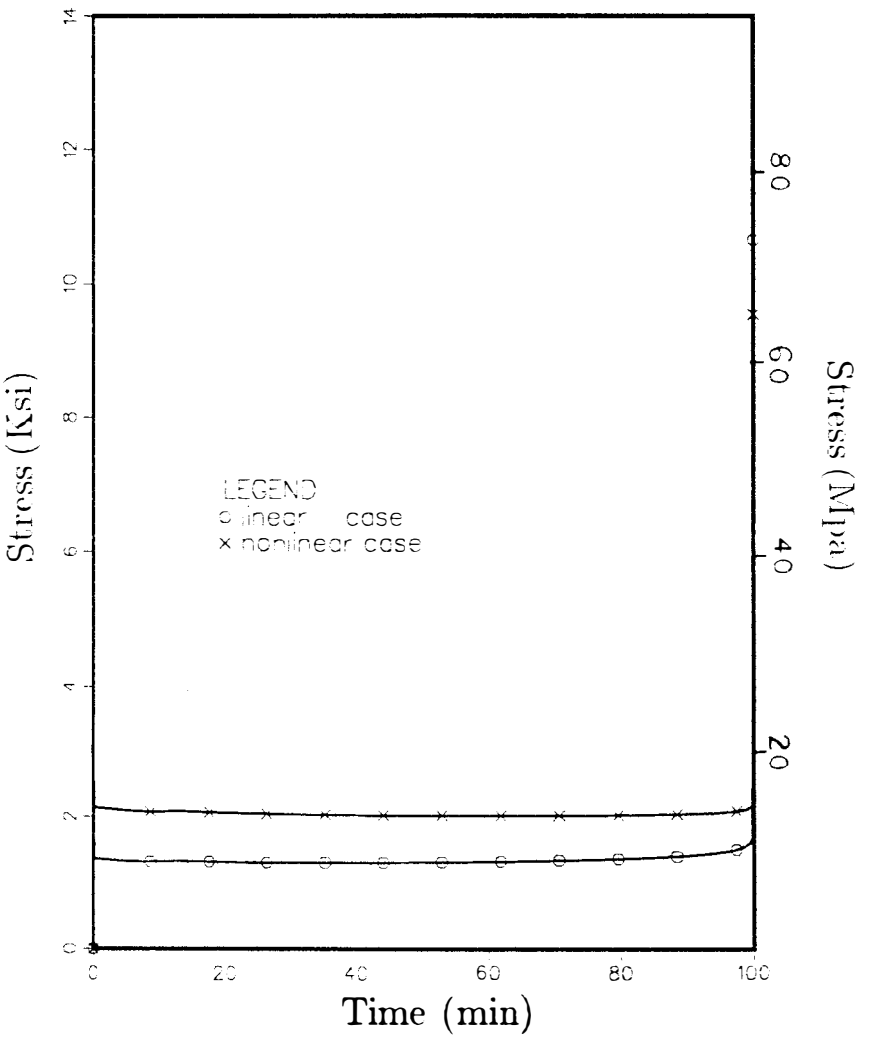

Figure 4.24: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 300^0C$ - **case b**

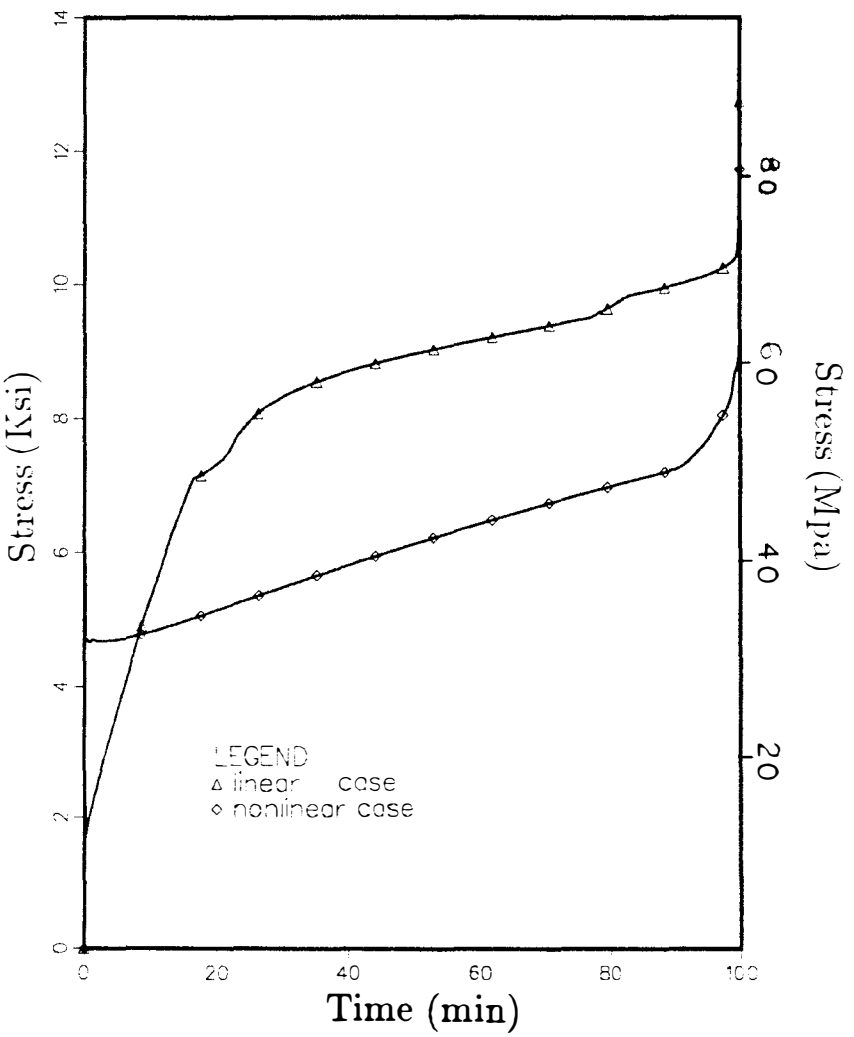

Figure 4.25: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 300^0C$ - **case c**

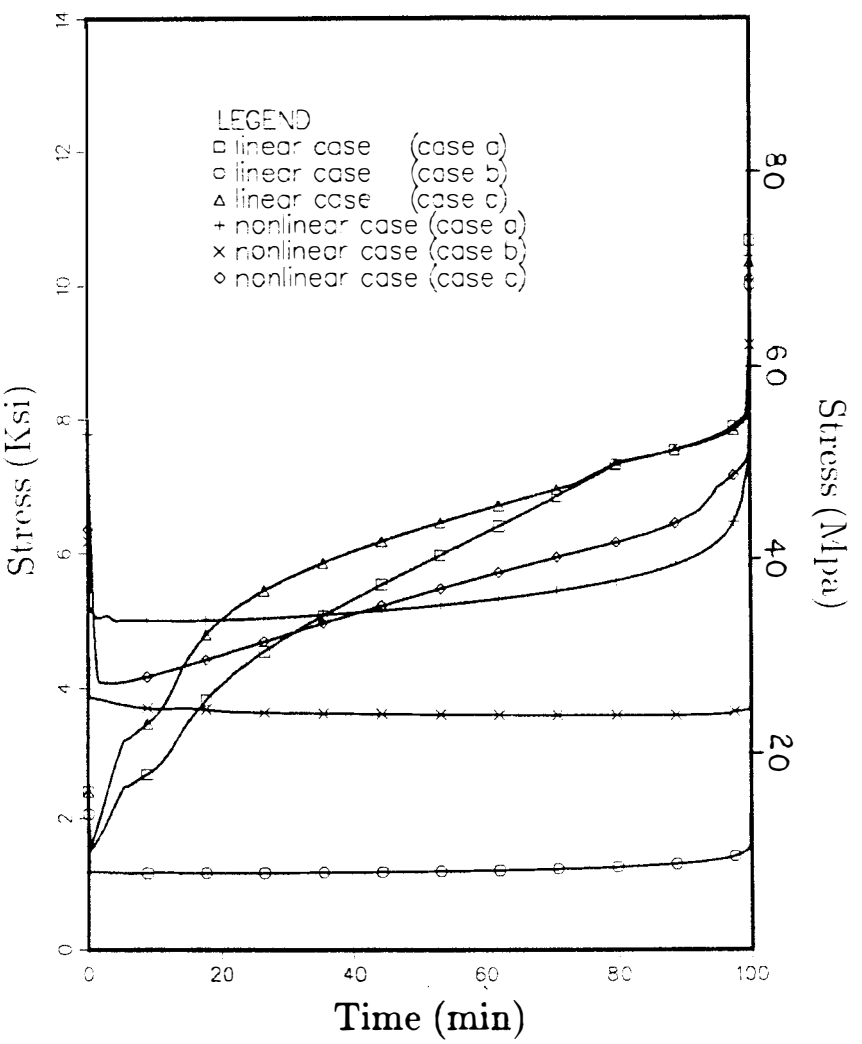

Figure 4.26: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 250^0C$

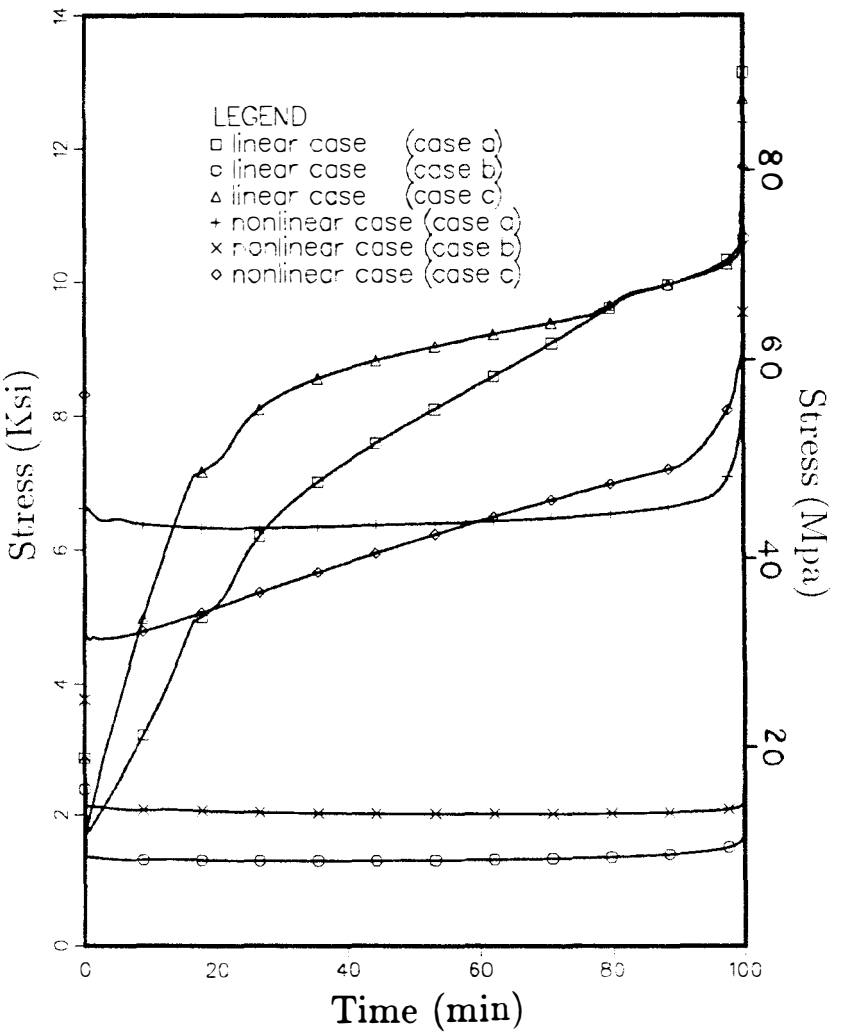

Figure 4.27: Stress histories along optimal temperature paths for *APC-2*, assuming $T_I = 300^0C$

For all cases note the jump discontinuities at $t = 0$ and $t = t_f$ in both temperatures and stresses. The magnitude of the first jump in temperature, which occurs $t = 0$, depends strictly on the shift-factor function $a(T)$ and the vertical shift factor $v_2(T)$ and is observed to be directed downward for all cases in our computations. The second jump, which occurs at the terminal $t = t_f$, depends on the prescribed final temperature T_F as well as on $t = t_f$. It is noted that those paths terminate at levels above the room temperture T_R, requiring an downward jump at $t = t_f$. If $T_f > T_F$, then obviously $\sigma(t) < \sigma(t_f^+)$ for $t > t_f$ because the stresses continue to relax with time.

It can be noted from Table 4.1 that nonlinearity can provide up to 37% reduction in the final residual stresses, beyond the predictions of linear viscoelastic response, for $T_I = 300^\circ C$. For $T_I = 250^\circ C$ nonlinearity can yield a 17% advantage. Figures 4.18 - 4.25 demonstrate the detailed roles played by the various material functions which relate the thermoviscoelastic behavior of *APC-2*.

4.4.4 Conclusions

This part of the dissertation presented an iterative method for predicting optimal cooling paths that minimize residual thermal stresses at time $t = t_f^+$, i.e., immediately after termination of cooling in *APC-2* composite laminate. The response of the material was described by a nonlinear, thermorheologically complex, viscoelastic model.

Based upon the experimental data obtained by Xinran [1] and Schapery's nonlinear viscoelastic model [2], computations were made for two specific cases of initial stress free temperatures $T_I = 250^0 C$ and $T_I = 300^0 C$. From the computational results it is important to note the followings:

- the iterative scheme typically converges within five to six iterations.
- all computed optimal temperature paths, $T_\Omega(t)$, were shown to possess both initial and final jump discontinuities, with continuous, monotonically decreasing, gradual

variation between $t = 0^+$ and up to $t = t_f^-$, and rapid variation at very short period just before $t = t_f$.

- for all the cases considered herein both the first and final temperature jumps were directed downward. The residual thermal stresses will continue to relax at times beyond the final time t_f.

As verification checks for the iterative numerical scheme, prototype solutions were considered for the simpler cases of a three element and "power law" models. In the case of the three element model it was possible to obtain an analytic solution for $T_\Omega(t)$ and, therefore, verify the validity of our numerical scheme.

In the absence of any mathematical proof of convergence for our optimization scheme, it was necessary to provide several ad-hoc numerical verifications. These were obtained by introducing several kinds of perturbations into the optimal path $T_\Omega(t)$ and computing the residual thermal stress $\sigma(t_f^+)$ which corresponded to those perturbed paths. It was found that cooling along the optimal path, indeed, gave $\sigma(t_f^+)$ below the other values.

The results of this study show that the residual thermal stresses that develop in thermoplastic resin composites, such as *APC-2*, are affected most substantially by the time-dependent material response of the resin. In addition, it was observed that these stresses reach magnitudes which approach the level of the ultimate transverse strength of the composite ply. Consequently, it is essential to account for nonlinear effects in the evaluation of those stresses and in the determination of optimal temperature paths. Variations among the various viscoelastic material response functions, as well as uncertainties concerning the stress free temperature lead to predictions of residual thermal stress which range between $9.1 ksi$ and $13.15\ ksi$. This unsatisfactory variation in our predictions stems from the incomplete data base for *APC-2*, which exists at the present time, not from weaknesses in our computational scheme. It is, therefore, desirable to extend the experimental characterization work of Xinran [1] to include a higher range of temperatures as well as transient temperature response to distinguish between the functions $v_1(T)$ and $v_2(T)$.

Part II

Effects of Nonuniform Crystallinity on Stress Distributions in Cross-Ply Graphite/PEEK (APC-2) Laminates

Chapter 5

General

Abstract This part of the dissertation presents analytical/computational results for the inplane stresses in a moderately thick, cross-ply, graphite/PEEK (*APC-2*) laminate accounting for effects of nonuniform PEEK crystallinity across the plate's thickness. These effects are incorporated as crystallinity-induced variations in the longitudinal modulus E_L, including its nonlinear dependence on strain.

It is well known that polyetheretherketone (PEEK) possesses a semicrystalline structure [19] - [23]. The crystalline microstructure and degree of crystallinity depend in a complicated manner on temperature history, [24] - [30]. Furthermore, it has been noticed that the character of the semicrystalline microstructure of PEEK in the presence of carbon fibers differs significantly from the random dispersion of spheroidal crystalline regions which occur in bulk resin. Specifically, it is thought that the fibers provide profuse sources for the nucleation of crystallinity within the PEEK resin, which thereby result in transcrystalline microstructure [29] - [33].

The mechanical response of graphite-PEEK (*APC-2*) composites at various levels of crystallinity was examined by several investigators [34] - [36]. Distinct degrees of crystallinity were attained by cooling the composite at different rates, down from the melt temperature.

It appears that the most comprehensive study to date concerning the effects of the degree of crystallinity on the mechanical properties of *APC-2* was conducted by Crossman

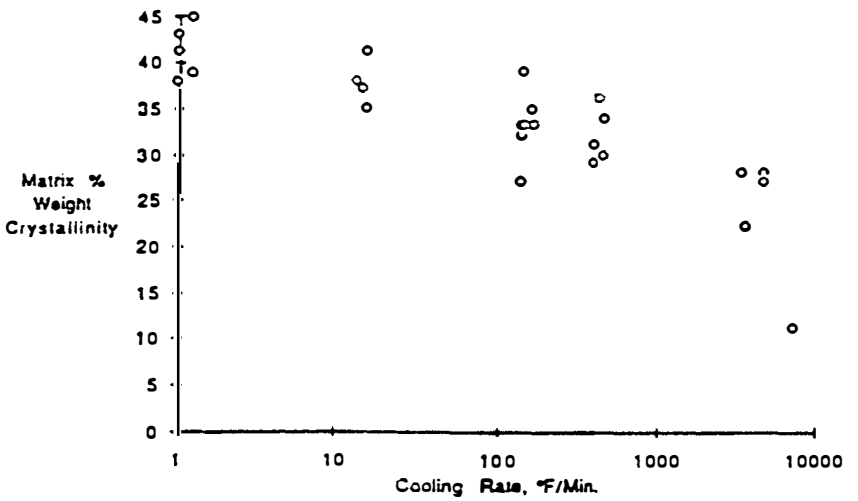

Figure 5.1: Matrix percent weight crystallinity vs. cooling rate in ${}^0F/min$ According to Ref. [37].

and Amateau [15]. Accordingly, Figure 5.1 shows a typical relationship among the degree of crystallinity (in percent weight) vs. cooling rate (in ${}^0F$ per minute), while Figure 5.2 exhibits typical tensile stress- strain curves for unidirectionally reinforced coupons that were cooled at various rates and loaded parallel to the fiber direction. It is argued in Reference [15] that the effect of modulus stiffening observed in Figure 5.2 is due to the initial misalignments, kinks and wobbliness induced in the fibers during processing. These imperfections are "straightened out" under increasing tension, thus magnifying the tensile modulus E_L. This dissertation appears to be corroborated by the wide scatter observed in the measurements of the transverse modulus E_T and Poisson's ratios ν_{LT} and ν_{TL} shown in Figure 5.3. It is indeed reasonable to expect that the latter properties would vary substantially along a specimen reinforced by misaligned fibers. It is interesting to note that an analogous explanation was provided for the modulus stiffening that was observed

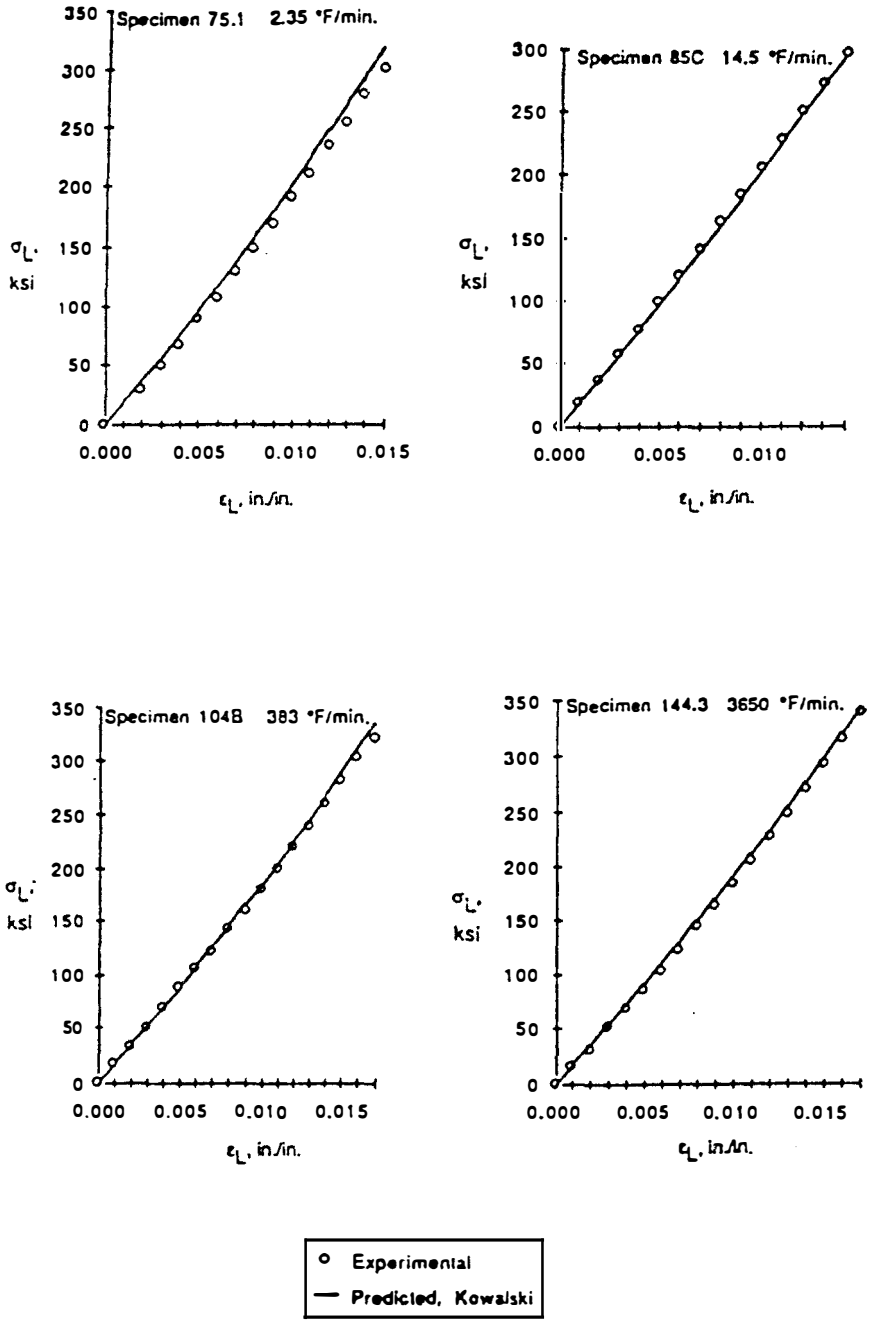

Figure 5.2: Longitudinal stress-strain data for specimens cooled at distinct rates. According to Ref. [37].

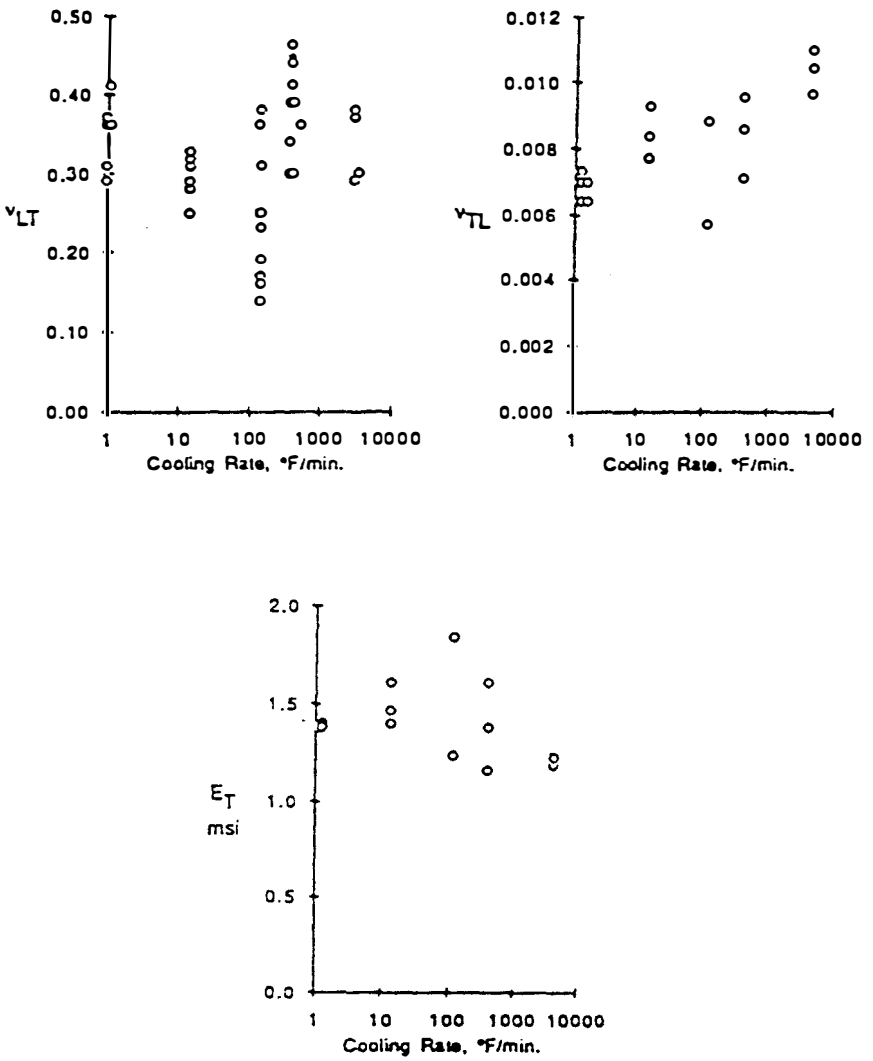

Figure 5.3: The Transverse Modulus E_T and Poisson's Ratios ν_{LT} and ν_{TL} for specimens cooled at distinct rates. According to Ref. [37].

in tendons [37].

The variations in the stress-strain response of coupons cooled at different rates (thereby with distinct degrees of crystallinity), which are shown in Figure 5.2, suggest that the transcrystalline microstructure acts as a "brace" which enhances the alignment of the graphite fibers within the composite. It follows that a variation in the degree of crystallinity across the thickness of an *APC-2* plate would result in a corresponding nonuniform stress-strain response of the laminate. Substantial spatial variations in the degree of crystallinity are possible within thick laminates. Several predictions and measurements of crystalline nonuniformity are given in Reference [26].

Finally, it should be noted that if the proposition of fiber misalignment – as forwarded in References [15] and [37] – is indeed correct, then the effect of degree of crystallinity should be much more noticeable in compression than under tension, as has been observed for some thermoset-resin composites [38]. However, no such experimental information seems to be available at the present time on the compressive response of *APC-2*. Consequently, the analysis and computations presented in this paper are restricted to the tensile range.

Chapter 6

Formulation

The nonlinear, longitudinal, stress-strain response exhibited in Figure 5.2 can be expressed as follows:

$$E_L = E_L^0(1 + A_0\chi + A_1\epsilon + A_2\epsilon^2 + A_3\chi\epsilon + A_4\chi\epsilon^2) \tag{6.1}$$

In Equation 6.1, E_L denotes the longitudinal modulus of *APC-2* parallel to the fiber directions, χ is the degree of crystallinity (in fractional weight) and ϵ the longitudinal strain.

To obtain Equation 6.1 from Figure 5.2, the cooling rates were converted to value of χ (ranging between 0 and 1) with the aid of Figure 5.1.

The numerical values of E_L^0 and $A_i (i = 0, \ldots, 4)$ were determined by a least error numerical routine. The values obtained through this curve fitting approach, were found to be:

$$\begin{aligned} E_L^0 &= 16.22\ Msi \\ A_0 &= 0.0013 \\ A_1 &= -18.411 \\ A_2 &= 600.61 \\ A_3 &= 1.1485 \\ A_4 &= -37.018 \end{aligned}$$

Since cross-ply laminates develop secondary compressive stresses in directions normal to the tensile load even under uniaxial extension, an excursion into the compressive range is

unavoidable. In view of the fact that Equation 6.1 is derived strictly for the tensile range, it has been modified to read $E_L = E_L^0(1 + A_0\chi)$ within the compressive range. This ad hoc modification proved to have only a minor effect on the results computed in the present work, in view of the relatively small amplitudes attained by the compressive stresses.

A somewhat more serious approximation in the formulation has been necessitated by the wide scatter in the experimental values for the transverse modulus E_T and the Poisson's ratios ν_{LT} and ν_{TL} reported in Reference [15]. This scatter precludes the formulation of an analytic expression for a nonlinear elastic strain energy function for the material. On the other hand, in view of the relatively small magnitude of E_T/E_L and the secondary effects of ν_{LT} and ν_{TL} on laminate stress analysis, we shall assume that E_T, ν_{LT} and ν_{TL} are constants, of magnitudes that correspond to the average of all their experimentally recorded values. We take

$$E_T = 1.4\,Msi,\ \nu_{LT} = 0.31,\ \nu_{TL} = 0.0083 \tag{6.2}$$

Following standard notation [4], the stress-strain relations for an individual, $0^\bullet$ ply are

$$\begin{aligned} \sigma_L &= Q_L(\chi, \epsilon_L)\epsilon_L + Q_{LT}\epsilon_T \\ \sigma_T &= Q_{TL}\epsilon_L + Q_T\epsilon_T \end{aligned} \tag{6.3}$$

where

$$\begin{aligned} Q_L(\chi, \epsilon_L) &= rE_L(\chi, \epsilon_L) \\ Q_T &= rE_T \\ Q_{LT} &= \nu_{LT}Q_T = Q_{TL},\ \textit{with}\ r = 1/(1 - \nu_{LT}\nu_{TL}). \end{aligned}$$

In Equation 6.3 subscripts L and T denote longitudinal and transverse directions, respectively. Note that in view of Equation 6.2, both Q_T and Q_{LT} are constants, and the thermodynamic requirement $\partial\sigma_L/\partial\epsilon_T = \partial\sigma_T/\partial\epsilon_L$ is satisfied. It is worth noting that the

expression $Q_{LT} = r\nu_{LT}E_L$, which is a valid alternative for the linear case, does not satisfy the above thermodynamic requirement when $E_L = E_L(\epsilon_L)$ and is therefore unacceptable in the present case.

For a cross-ply lay-up under inplane loads and moments, upon assuming

$$\epsilon_x = \epsilon_x^0 + K_x Z$$
$$\epsilon_y = \epsilon_y^0 + K_y Z \tag{6.4}$$

We have the following expressions for the inplane force resultants N_x, N_y and the moments M_x, M_y:

$$\begin{pmatrix} N_x \\ N_y \end{pmatrix} = \int_{-\frac{h}{2}}^{\frac{h}{2}} \begin{bmatrix} Q_{11}^{(i)} & Q_{12}^{(i)} \\ Q_{12}^{(i)} & Q_{22}^{(i)} \end{bmatrix} \begin{pmatrix} \epsilon_x \\ \epsilon_y \end{pmatrix} dz$$
$$\begin{pmatrix} M_x \\ M_y \end{pmatrix} = \int_{-\frac{h}{2}}^{\frac{h}{2}} \begin{bmatrix} Q_{11}^{(i)} & Q_{12}^{(i)} \\ Q_{12}^{(i)} & Q_{22}^{(i)} \end{bmatrix} \begin{pmatrix} \epsilon_x \\ \epsilon_y \end{pmatrix} z\, dz \tag{6.5}$$

In Equation 6.5, $Q_{12}^{(i)} = Q_{LT}$ for all plies, while $Q_{11}^{(i)} = Q_L$, $Q_{22}^{(i)} = Q_T$ for the 0^0 plies and $Q_{11}^{(i)} = Q_T$, $Q_{22}^{(i)} = Q_L$ for the 90^0 plies.

Chapter 7

Analysis

To calculate the stresses within a cross-ply laminate under prescribed values of N_x, N_y, M_x and M_y, it is necessary to first determine the strains in Equation 6.4, hence the midplane values ϵ_x^0, ϵ_y^0 and the curvatures K_x, K_y.

It should be noted that departures from classical laminate analysis occur in the present work due to two causes in $E_L = E_L(\chi, \epsilon_L)$. The spatial dependence of the degree of crystallinity, namely $\chi = \chi_{(Z)}$, induces a material inhomogeneity, while the strain-dependence of E_L introduces response nonlinearity.

Substitution of Equations 6.3 and 6.4 into Equation 6.5 yields four nonlinear Equations in ϵ_x^0, ϵ_y^0, K_x, K_y of the form

$$\begin{aligned} N_x &= N_x(\epsilon_x^0, \epsilon_y^0, K_x, K_y) \\ N_y &= N_y(\epsilon_x^0, \epsilon_y^0, K_x, K_y) \\ M_x &= M_x(\epsilon_x^0, \epsilon_y^0, K_x, K_y) \\ M_y &= M_y(\epsilon_x^0, \epsilon_y^0, K_x, K_y) \end{aligned} \tag{7.1}$$

Consider a given distribution of $\chi(z)$ and some guess values $\bar{\epsilon}_x^0$, $\bar{\epsilon}_y^0$, $\bar{K}_x$ and $\bar{K}_y$ with $N_x = \bar{N}_x$, $N_y = \bar{N}_y$, $M_x = \bar{M}_x$, and $M_y = \bar{M}_y$ determined from Equation 6.5. Obviously, these values are unlikely to correspond to the prescribed loads, whereby

$$N_x - \bar{N}_x(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) = \triangle N_x$$

$$\begin{aligned} N_y - \bar{N}_y(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) &= \triangle N_y \\ M_x - \bar{M}_x(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) &= \triangle M_x \\ M_y - \bar{M}_y(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) &= \triangle M_y \end{aligned} \tag{7.2}$$

Increment $\bar{\epsilon}_i^0$ and $\bar{K}_i$ to $\epsilon_i^0 + \triangle\epsilon_i$ and $\bar{K}_i + \triangle K_i$, respectively, aiming at

$$\begin{aligned} N_x - \hat{N}_x(\bar{\epsilon}_x^0 + \triangle\epsilon_x^0, \bar{\epsilon}_y^0 + \triangle\epsilon_y^0, \bar{K}_x + \triangle K_x + \bar{K}_y + \triangle K_y) &= 0 \\ N_y - \hat{N}_y(\bar{\epsilon}_x^0 + \triangle\epsilon_x^0, \bar{\epsilon}_y^0 + \triangle\epsilon_y^0, \bar{K}_x + \triangle K_x + \bar{K}_y + \triangle K_y) &= 0 \\ M_x - \hat{M}_x(\bar{\epsilon}_x^0 + \triangle\epsilon_x^0, \bar{\epsilon}_y^0 + \triangle\epsilon_y^0, \bar{K}_x + \triangle K_x + \bar{K}_y + \triangle K_y) &= 0 \\ M_y - \hat{M}_y(\bar{\epsilon}_x^0 + \triangle\epsilon_x^0, \bar{\epsilon}_y^0 + \triangle\epsilon_y^0, \bar{K}_x + \triangle K_x + \bar{K}_y + \triangle K_y) &= 0 \end{aligned} \tag{7.3}$$

To determine $\triangle\epsilon_i$ and $\triangle K_i$ expand $\hat{N}_i$ and $\hat{M}_i$ in Taylor series, terminating after the linear terms, namely

$$\begin{aligned} N_x - \bar{N}_x(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) - \frac{\partial \bar{N}_x}{\partial \bar{\epsilon}_x^0}\triangle\epsilon_x^0 - \frac{\partial \bar{N}_x}{\partial \bar{\epsilon}_y^0}\triangle\epsilon_y^0 - \frac{\partial \bar{N}_x}{\partial \bar{K}_x}\triangle K_x - \frac{\partial \bar{N}_x}{\partial \bar{K}_y}\triangle K_y &= 0 \\ N_y - \bar{N}_y(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) - \frac{\partial \bar{N}_y}{\partial \bar{\epsilon}_x^0}\triangle\epsilon_x^0 - \frac{\partial \bar{N}_y}{\partial \bar{\epsilon}_y^0}\triangle\epsilon_y^0 - \frac{\partial \bar{N}_y}{\partial \bar{K}_x}\triangle K_x - \frac{\partial \bar{N}_y}{\partial \bar{K}_y}\triangle K_y &= 0 \\ M_x - \bar{M}_x(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) - \frac{\partial \bar{M}_x}{\partial \bar{\epsilon}_x^0}\triangle\epsilon_x^0 - \frac{\partial \bar{M}_x}{\partial \bar{\epsilon}_y^0}\triangle\epsilon_y^0 - \frac{\partial \bar{M}_x}{\partial \bar{K}_x}\triangle K_x - \frac{\partial \bar{M}_x}{\partial \bar{K}_y}\triangle K_y &= 0 \\ M_y - \bar{M}_y(\bar{\epsilon}_x^0, \bar{\epsilon}_y^0, \bar{K}_x, \bar{K}_y) - \frac{\partial \bar{M}_y}{\partial \bar{\epsilon}_x^0}\triangle\epsilon_x^0 - \frac{\partial \bar{M}_y}{\partial \bar{\epsilon}_y^0}\triangle\epsilon_y^0 - \frac{\partial \bar{M}_y}{\partial \bar{K}_x}\triangle K_x - \frac{\partial \bar{M}_y}{\partial \bar{K}_y}\triangle K_y &= 0 \end{aligned} \tag{7.4}$$

Using Expressions 7.2 and rearranging we obtain

$$\begin{bmatrix} \frac{\partial \bar{N}_x}{\partial \bar{\epsilon}_x^0} & \frac{\partial \bar{N}_x}{\partial \bar{\epsilon}_y^0} & \frac{\partial \bar{N}_x}{\partial \bar{K}_x} & \frac{\partial \bar{N}_x}{\partial \bar{K}_y} \\ \frac{\partial \bar{N}_y}{\partial \bar{\epsilon}_x^0} & \frac{\partial \bar{N}_y}{\partial \bar{\epsilon}_y^0} & \frac{\partial \bar{N}_y}{\partial \bar{K}_x} & \frac{\partial \bar{N}_y}{\partial \bar{K}_y} \\ \frac{\partial \bar{M}_x}{\partial \bar{\epsilon}_x^0} & \frac{\partial \bar{M}_x}{\partial \bar{\epsilon}_y^0} & \frac{\partial \bar{M}_x}{\partial \bar{K}_x} & \frac{\partial \bar{M}_x}{\partial \bar{K}_y} \\ \frac{\partial \bar{M}_y}{\partial \bar{\epsilon}_x^0} & \frac{\partial \bar{M}_y}{\partial \bar{\epsilon}_y^0} & \frac{\bar{M}_y}{\bar{K}_x} & \frac{\partial \bar{M}_y}{\partial \bar{K}_y} \end{bmatrix} \begin{pmatrix} \triangle\epsilon_x^0 \\ \triangle\epsilon_y^0 \\ \triangle K_x \\ \triangle K_y \end{pmatrix} = \begin{pmatrix} \triangle N_x \\ \triangle N_y \\ \triangle M_x \\ \triangle M_y \end{pmatrix} \tag{7.5}$$

The above scheme is known as the Newton-Raphson method. It can be repeated until the increments $\triangle\epsilon_i$ and $\triangle K_i$ attain desired small values. The detailed expressions for $\bar{N}_x$, $\bar{N}_y$, $\bar{M}_x$, $\bar{M}_y$ and their derivatives with respect to ϵ_x^0, ϵ_y^0, K_x, and K_y are given in the Appendix B.

Chapter 8

Computations and results

Computations were performed for a $48 - ply$ thick laminate of a $[(O_2/90)_8]_s$ lay-up. The degree of crystallinity χ was assumed to vary nonuniformly across the thickness. Guided by Reference [26] we chose $\chi(Z) = 0.4 - 0.3(|Z|/h)$ where Z is the thickness coordinate measured from the center of the laminate and h is half the plate's thickness. The quantity h corresponds to an assumed ply-thickness of $0.14 \times 10^{-3} m$ $(5.5 \times 10^{-3} inch)$.

Calculations were performed for twenty loading conditions, consisting of a uniaxial in-plane load N_x of several magnitudes, and of various combinations of N_x and a bending moment M_x. In all these computations, the initial guess values for ϵ_x^0, ϵ_y^0, K_x and K_y were obtained from the linear results. Convergence was achieved after about three steps. The convergence criterion was $(\triangle\epsilon_x^\bullet/\epsilon_x^\bullet)^2 + (\triangle\epsilon_y^0/\epsilon_y^0)^2 + (\triangle K_x/K_x)^2 + (\triangle K_y/K_y)^2 \leq 10^{-8}$. All cases involved low-level compressive stresses along the fiber directions in the $90^\bullet$ plies, when Equation 6.1 no longer applies. As stated earlier the response in compression was described by selecting $A_i = 0$ in Equation 6.1, with $i = 1, 2, 3, 4$. It was observed that for the relatively low levels of σ_y the results were insensitive to the uncertainty in the compressive values of E_L.

Results are listed in Table 8.1, where comparison is provided between values of maximal stresses. A detailed comparison between the present results and predictions of classical laminate theory is given in Figure 8.1, where the distribution of σ_x across the thickness of the laminate is shown for the loading condition $N_x = 25\ Kips/in$ and $M_x = 0$.

Table 8.1: Maximal stresses in *ksi* in the 0^0 plies σ_x and the 90^0 plies σ_y of a $[(0_2/90)_8]_s$ laminate under several loading conditions. Comparison is provided between the present results which account for nonuniform crystallinity χ with strain-dependent modulus E_L and predictions of linear laminate theory.

Loading Condition		Maximal Stress Value, ksi			
N_x	M_x	σ_x		σ_y	
$(kips/in)$	$(kips/in/in)$	Present work	Linear theory	present work	Linear theory
0.1		0.556	0.545	0.0446	0.0400
1.0		5.586	5.482	0.4455	0.4005
10.0		58.041	54.820	4.3565	4.0050
20.0		119.600	109.600	8.5530	8.0100
25.0		151.162	137.000	10.6150	10.0100
0.1	0.1	11.896	11.860	0.9136	0.7935
0.1	1.0	110.513	113.700	8.7330	7.5740
0.1	1.1	121.189	125.000	9.6009	8.3280
0.1	1.2	131.834	136.300	10.4670	9.0810
0.1	1.3	142.453	147.700	11.3340	9.8350
1.0	0.1	16.641	16.800	1.3140	1.1540
1.0	0.5	60.991	62.060	4.7870	4.1680
1.0	0.8	93.509	96.010	7.3920	6.4280
1.0	1.0	114.932	118.600	9.1260	7.9350
1.0	1.2	136.214	141.300	10.8590	9.4420
10.0	0.1	62.152	66.130	5.2210	4.7580
10.0	0.2	72.996	77.450	6.0860	5.5120
10.0	0.5	105.163	111.400	8.6760	7.7720
10.0	0.8	136.967	145.300	11.2630	10.0300
20.0	0.1	120.512	120.900	9.4150	8.7630

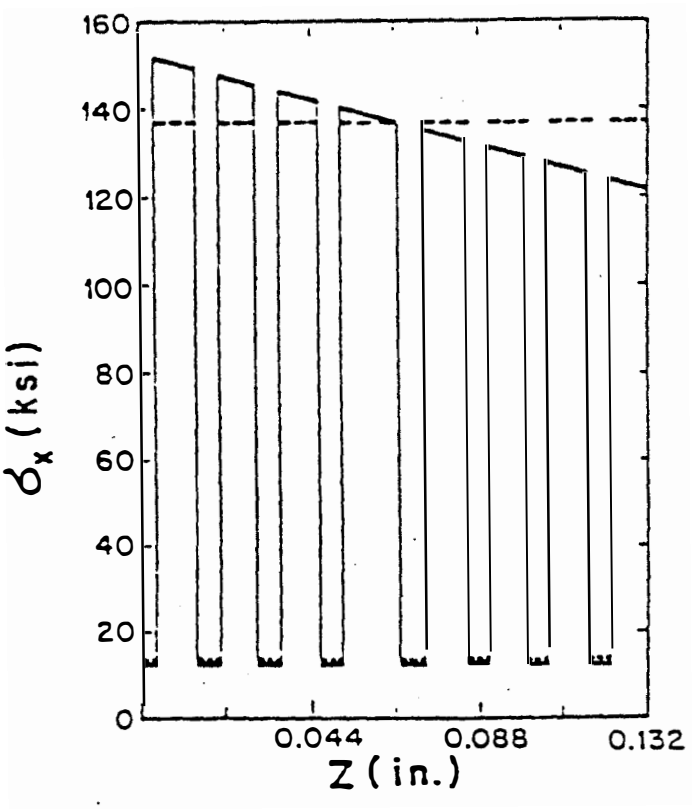

Figure 8.1: The distribution of the stress σ_x across the thickness of a $[(0_2^0/90^0)_8]_s$ *APC-2* laminate subjected to a uniaxial load $N_x = 25 Kips/in$. Results of present analysis (solid lines) and of classical laminate theory (dashed lines). Results shown for $0 \leq z \leq 0.132''$. In view of symmetry $\sigma_x(-z) = \sigma_x(z)$.

Note that for the loading combinations listed in Table 8.1 the largest discrepancy between linear predictions and $E_L = E_L(\chi, \epsilon)$ effects is about 10% in values of σ_x^{max} and 15% in σ_y^{max}. Furthermore, linear theory appears to under-predict these values. More significant discrepancies are likely to occur when nonlinear, crystallinity affected, compressive response is to be considered. However, as noted in the introduction to this work, this aspect of *APC-2* response still awaits experimental characterization.

BIBLIOGRAPHY

Bibliography

[1] Xinran, X. *Viscoelastic characterization of thermoplastic matrix composites.* Ph. D. Thesis, Free University of brussels. November 1987.

[2] Schapery, R. A. *Further development of the thermodynamic constitutive theory: stress formulation.* Purdue University Report AA and ES 69-2 (AFML Contract F33615-67-C-1412), 1969.

[3] Peretz, D., and Weitsman, Y. *The non-linear thermoviscoelastic characterization of a structural adhesive. J. Rheology.* 1983. 27, 97-114.

[4] Jones, R. M. *Mechanics of composite materials*, Scripta Book Co., Washington DC, 1975

[5] Weitsman, Y. *Optimal cool-down in linear viscoleasticity, J. App. Mech.,* 1980, 47, 35-39.

[6] Gurtin, M. E., and Murphy, L. F. *On optimal temperature paths for themorheologically simply viscoelastic materials, Quart. App. Math.,* 1980, 38. 179-190.

[7] Weitsman, Y., and Harper, B. D. *Optimal cool-down of cross-ply composite laminates and adhesive joints, J. App. Mech.,* 1982, 49, 735-739.

[8] Harper, B. D. *Optimal cooling paths for a class of thermortheologically complex viscoelastic materials, J. App. Mech.,* 1985, 52, 634-638.

[9] Schapery, R. A. , *Polym. Eng. Sci.,* 1969, 9, 295 - 310

[10] Schapery, R. A. , 1965,Purdue University Report A and ES 65-7

[11] Schapery, R. A. , *Int. J. Solids Structures*, 1966, 2, 407 - 425

[12] Schapery, R. A. , 1968, Purdue University Report AA and ES 68-4

[13] Peretz, D. and Weitsman, Y. *Nonlinear viscoelastic characterization of FM-73 adhesive J. Rheology*, 1982, 26, 245-261.

[14] Stoer, J., and Bulirsch, R. *Introduction to Numerical Analysis*

[15] Crossman, S. P., and Anateau, M. F. *The effect of processing on a graphite fiber/ polyehtereterketone thermoplastic composite*, Proceedings 33rd international SAMPE symposium, 1988, 681 - 692.

[16] Barnes, J. A., and Cogswell F. N. *Transverse flow processes in thermoplastic composite*. International polymer processing, 1987, 1, 157-165.

[17] Pipkin, A. C. *Lectures on viscoelasticity theory*, Springer-verlag, New York, 1972

[18] Fletcher, R. *Practical Methods of Optimization*, 1987, John Wiley and Sons Ltd.

[19] Blundell, D. J., and Osborn, B. N. *The Morphology of Poly(arly-ether-ether-ketone)*. Polymer,Vol.24,pp. 953-958 (1983)

[20] Hay, J. N., Kemmish, D. J., Langford, J. I., Rae, A. I. M. *The Structure of Crystalline PEEK*. Polymer Communications, Vol 25,pp. 175-178 (1984)

[21] Wakelyn, N. T. *On the structure of Poly(etheretherketone)(PEEK)*. Polymer communications, Vol. 25,pp. 306-308 (1984)

[22] Blundell, D. J., Chalmers, J. M., Mackenzie, M. W., and Gaskin, F. *Crystal Morphology of Matrix of PEEK-Carbon Fiber Aromatic Polymer Composites, I. Assessment of Crystallinity*. SAMPE Quarterly. Vol. 16, No. 4, pp. 22-30 (1985)

[23] Kumar, S., Anderson, D., and Adams, W. W. *Crystallization and Morphology of Poly(aryl-ether-ether-ketone)* Polymer, Vol.27, pp. 329-336 (1986)

[24] Muzzy, J. D., Bright, D. G., and Hoyos, G. H. *Solidification of Poly(Ethylene Terephthalate) with incomplete Crystallization.* Polymer Engineering and Science, Vol. 18, No. 6, pp. 437-442 (1978)

[25] Blundell, D. J., and Osborn, B. N. *Crystalline Morphology of the Matrix of PEEK-Carbon Fiber Aromatic Polymer Composites, II. Crystallization Behavior.* SAMPE Quarterly, Vol. 17, No. 1, pp. 1-6 (1985)

[26] Seferis, J. C., and Velisaris, C. N. *Modeling - Processing - Structure Relationships of Polyetheretherketone (PEEK) Based Composites.* Proceedings 31st international SAMPE Symposium, pp. 1236-1250 (1986)

[27] Seferis, J. C. *Polyetheretherketone (PEEK): Processing - Structure and Properties Studies for Matrix in High Performance Composites.* Polymer Composites, Vol. 7, pp. 158-169 (1986)

[28] Manson, J. A. E., and Seferis, J. C. *Process Analysis and Properties of High Performance Thermoplastic Composites.* Engineering Appl. of New Comp. (1988)

[29] Cebe, P., Chung, S. Y., and Hong, S. D. *Effect of Thermal History on Mechanical Properties of Polyetheretherketone Below the Glass Transition Temperature.* Journal of Applied Polymer Science, Vol. 33, pp. 487-503 (1987)

[30] Seferis, J. C., Ahlstrom, C., and Dillman, S. H. *Cooling Rate and Annealing as Processing Parameters for Semicrystalline Thermoplastic-Based composites.* ANTEC '87, pp. 1467-1471 (1987)

[31] Lustiger, A. *Considerations in the Utilization of Semicrystalline Thermoplastic Advanced Composites.* SAMPE Journal, pp. 13-16 (1984)

[32] Lee, Y. C., and Porter, R. S. *Crystallization of Polyetheretherketone (PEEK) in Carbon Fiber Composites.* ONR Technical Report No. 26, Contract N00014-83-K-0228, April 25, 1985.

[33] Geil, P. H. Private Communication

[34] Davies, M., Leach, D. C., Moore, D. R., and Turner, R. M. *Mechanical Performance of Semi-Crystalline, Thermoplastic Matrix Composites for Elevated Temperature Service.* Proceedings of ICCM VI and ECCM2 1.299-1.309 (1987)

[35] Curtis, P. T., Davies, P., Partridge, I. K., and Sainty, J. P. *Cooling Rate Effects in Peek and Carbon Fibre-PEEK Composites* Proceedings of ICCM VI and ECCM2, 4.401-4.412

[36] Lee, W. I., Talbott, M. F., and Springer, G. S. *Effects of Cooling Rate on the Crystallinity and Mechanical Properties of Thermoplastic Composites.* Journal of Reinforced Plastics and Composites, vol. 6, pp. 2-12 (1987)

[37] Kastelic, J., Palley, I., and Baer, E. *A Structural Mechanical Model for Tendon Crimping.* Journal Biomechanics, vol. 13, pp. 887-893 (1980)

[38] Mathison, S. R., Pindera, M. J., and Herakovich, C. T. *Nonlinear Response of Resin Laminates Using Endochronic Theory.* In *Visco-Plastic Behavior of New Materials* D. Hui and T.J. Kozik, Editors. ASME Publication PVP - Vol.184 MD-Vol. 17, pp. 111-117 (December 1989)

APPENDICES

Appendix A

Computer Code

```
C
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                               *
*  Purpose:   To  predict the optimal cooling temperature paths which          *
*             minimizes residual thermal stresses at time t = tf              *
*             immediately after termination of cooling in APC-2 composites    *
*             laminate, modelled as nonlinear, thermorheologically complex,   *
*             viscoelastic  thin plate, and also computes stress  history     *
*             along the temperature path.                                     *
*                                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      PARAMETER(IDATB=9,IDATR=10,IDATP=11,IDATI=12,IDATD=19,
     +IDATC=39,IDATW=30,IDATX=31,IDATY=32,IDATZ=33,IDATV=34,
     +IDATA=20,IDATE=21,IDATS=22)
      PARAMETER  (NDEG=3,NOBSA=5,NOBSB=8,NOBSC=9,NOBSD=17)
C
      PARAMETER(NDATA=6,MDATA=3)
      REAL*8 BREAKN(2*NDATA),CSCOEFN(4,2*NDATA)
      REAL*8 BREAKM(2*MDATA),CSCOEFM(4,2*MDATA)
C
      PARAMETER(M=3)
      REAL*8 ATLG(M),TRCP(M)
      COMMON/VATDATA/ATLG,TRCP
C
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      COMMON/CSN/CSCOEFN,BREAKN,IBREAKN
      COMMON/CSM/CSCOEFM,BREAKM,IBREAKM
C
      REAL*8 T(0:NM),OPTEMP(0:NM),STRESS(0:NM),OPTSTR(0:NM),
     +       TEMPSTR(0:NM),STEMP(0:NM),
     +       DERZER(NM),DERFIR(NM),DERSEC(NM),SXI(NM)
      COMMON/BBB/ TIMEF,TI,TF,TOL
C
      PARAMETER(ND=5000)
```

```
      REAL*8 TIME(ND),TEMP(ND),TDELT(ND),SDELT(ND)
C
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      COMMON/CISML/ERRABS,ERRREL,IRULE
      EXTERNAL DQDAG,DQDAGS,DZBREN
      EXTERNAL DU4INF, UMACH, DUMINF

C
      PARAMETER(NNDATA=5000)
      REAL*8  BREAK(2*NNDATA), CSCOEF(4,2*NNDATA), STR(NNDATA),
     &        TEM(NNDATA)
      COMMON/CCC/ BREAK, CSCOEF, NINTV,JDBUG
      COMMON/TDROP/ TEMPI,TEMPF,NSTEP
C
      COMMON/CURV/B,CSB,STIME
      COMMON/PPP/EDERS,EDERF,TOP,BOT
C
      REAL*8  B(NDEG+1), SSPOLY(NDEG+1), STAT(10), CSB(4,ndeg+1)
      REAL*8  tlga(NOBSa), edata(NOBSa), tlgb(NOBSb), edatb(NOBSb),
     &        tlgc(NOBSc), edatc(NOBSc), tlgd(NOBSd), EDATD(NOBSD)
C
      CHARACTER  CLABEL(11)*15, RLABEL(1)*4
      EXTERNAL DCSCON, DCSVAL, DCSDER,DRCURV, WRRRL, WRRRN
C
      DATA RLABEL/'NONE'/, CLABEL/' ', 'Mean of X', 'Mean of Y',
     &     'Variance X', 'Variance Y', 'R-squared',
     &     'DF Reg.', 'SS Reg.', 'DF Error', 'SS Error',
     &     'Pts. with NaN'/
C
      DATA EDATA/9.17,9.13,9.10,9.06,9.02/
      DATA EDATB/9.13,9.10,9.06,9.02,8.98,8.77,8.44,7.89/
      DATA EDATC/8.77,8.44,7.89,7.32,6.76,6.37,5.98,5.74,5.70/
      DATA EDATD/5.98,5.74,5.70,5.65,5.60,5.40,5.10,4.60,
     +     4.20,3.70,3.20,2.70,2.30,1.80,1.35,.80,0D0/
C
      DATA TLGA/-6.,-5.,-4.,-3.,-2./
      DATA TLGB/-5.,-4.,-3.,-2.,-1.,0D0,1.,2./
      DATA TLGC/0d0,1.,2.,3.,4.,5.,6.,7.,8./
      DATA TLGD/6.,7.,8.,9.,10.,11.,12.,13,14.,15.,16.,17.,18.,
     +     19.,20.,21.,22./
      COMMON/WORKSP/ RWKSP
      REAL RWKSP(11616)
      CALL IWKIN(11616)
C
      OPEN(UNIT=IDATD,STATUS='UNKNOWN',FILE='DBGTEM.DAT')
      OPEN(UNIT=IDATS,STATUS='UNKNOWN',FILE='DBGSTR.DAT')
      OPEN(UNIT=IDATI,STATUS='UNKNOWN',FILE='DBGOPT.DAT')
      OPEN(UNIT=IDATB,STATUS='UNKNOWN',FILE='DBGOPT.LUK')
      OPEN(UNIT=IDATR,STATUS='UNKNOWN',FILE='DBGOPT.RCD')
      OPEN(UNIT=IDATP,STATUS='UNKNOWN',FILE='DBGOPT.PLT')
```

```
C
C --------->      Piecewise least square curve fitting
C
      CALL dRCURV (NOBSa, tlga, eDATA, NDEG, B, SSPOLY, STAT)
C
C      print *,' First curv ....'
C      CALL WRRRN ('B', 1, NDEG+1, B, 1, 0)
C      CALL WRRRN ('SSPOLY', 1, NDEG+1, SSPOLY, 1, 0)
C      CALL WRRRL ('%/STAT', 1, 10, STAT, 1, 0, '(2W10.4)', RLABEL,
C     &            CLABEL)
C      READ(5,*) ICON
C
      DO 1400 K = 1, NDEG+1
1400  CSB(1,K)=B(K)
C
      CALL dRCURV (NOBSb, tlgb, eDATb, NDEG, B, SSPOLY, STAT)
C
C      PRINT*,' Second curv ....'
C      CALL WRRRN ('B', 1, NDEG+1, B, 1, 0)
C      CALL WRRRN ('SSPOLY', 1, NDEG+1, SSPOLY, 1, 0)
C      CALL WRRRL ('%/STAT', 1, 10, STAT, 1, 0, '(2W10.4)', RLABEL,
C     &            CLABEL)
C      READ(5,*) ICON
      DO 1450 K = 1, NDEG+1
1450  CSB(2,K)=B(K)
C
      CALL dRCURV (NOBSc, tlgc, eDATc, NDEG, B, SSPOLY, STAT)
C
C      PRINT*,' Third curv ....'
C      CALL WRRRN ('B', 1, NDEG+1, B, 1, 0)
C      CALL WRRRN ('SSPOLY', 1, NDEG+1, SSPOLY, 1, 0)
C      CALL WRRRL ('%/STAT', 1, 10, STAT, 1, 0, '(2W10.4)', RLABEL,
C     &            CLABEL)
C      READ(5,*) ICON
      DO 1500 K = 1, NDEG+1
1500  CSB(3,K)=B(K)
C
      CALL dRCURV (NOBSD, tlgD, eDATD, NDEG, B, SSPOLY, STAT)
C
C      PRINT*,' Fourth curve ....'
C      CALL WRRRN ('B', 1, NDEG+1, B, 1, 0)
C      CALL WRRRN ('SSPOLY', 1, NDEG+1, SSPOLY, 1, 0)
C      CALL WRRRL ('%/STAT', 1, 10, STAT, 1, 0, '(2W10.4)', RLABEL,
C     &            CLABEL)
C      READ(5,*) ICON
      DO 1550 K = 1, NDEG+1
1550  CSB(4,K)=B(K)
C
      WRITE(6,166)
      PRINT*,'+ + + + + + + + + + + + + + + + + + + + + + + + + + + + + '
      PRINT*,'+                                                       +'
```

```
      PRINT*,'+                                                    +'
      PRINT*,'+                                                    +'
      PRINT*,'+                                                    +'
      PRINT*,'+         NONLINEAR VISCO-ELASTIC ANALYSIS           +'
      PRINT*,'+                                                    +'
      PRINT*,'+              NL = EXP(-A*(B-C))                    +'
      PRINT*,'+                                                    +'
      PRINT*,'+                                                    +'
      PRINT*,'+                                                    +'
      PRINT*,'+          Program 4:  revised on 4-25-1991          +'
      PRINT*,'+          Program 3:  revised on 3-28-1991          +'
      PRINT*,'+                                                    +'
      PRINT*,'+          Program 4:    DCDXMDBGN                   +'
      PRINT*,'+                  3:    DCDXMDBGR                   +'
      PRINT*,'+                  2:    CDXMDBGR                    +'
      PRINT*,'+                  1:    CDXMDBG                     +'
      PRINT*,'+                                                    +'
      PRINT*,'+                        (DBUG)                      +'
      PRINT*,'+                                                    +'
      PRINT*,'+ + + + + + + + + + + + + + + + + + + + + + + + + + + '
      PRINT*
      PRINT*,' INPUT  1:            For          H1 = 1'
      PRINT*,'        2:                         H2 = 1'
      PRINT*,'        3:                Simple material'
      READ(5,*) IH
C
      IF(IH.EQ.1.OR.IH.EQ.2)THEN
      PRINT*
      PRINT*,' Supply and change data for vertical shift factor ? '
      PRINT*
      PRINT *, ' INPUT 1:                   For   Yes '
      PRINT *, '       0:                          No '
      READ(5,*) JVAT
      IF(JVAT.EQ.1)THEN
      DO 678 MMM=1, 3
      PRINT*
      PRINT*,'ATLG(',MMM,') =  ? ','  TRCP(',MMM,') = ? '
      READ(5,*)   ATLG(MMM),TRCP(MMM)
 678  CONTINUE
                                                        ELSE
      ATLG(1) = -0.21
      TRCP(1) = 200.0
      ATLG(2) = -0.0031
      TRCP(2) = 129.375
      ATLG(3) = 0.025
      TRCP(3) = 40
      ENDIF
      ENDIF
C
C
C ---------->      Create interpolation function for nonlinear function ALPA
```

```
C
      CALL ALPACS(CSCOEFN,BREAKN,IBREAKN)

C
C --------->      Create interpolation function for nonlinear function OCTA
C
      CALL OCTACS(CSCOEFM,BREAKM,IBREAKM)
C
C --------->      Initialization
C
      R    =  .9125
      TG   = 125.
      ULTSTRL=325
      ULTSTRT=11
      ULTSTNL=0.015
      ULTSTNT=0.008
      CHKMIN=1.0
C
      PRINT*,' Input curing Temperature, TI = ?'
      READ(5,*) TI
      PRINT*,' Input room temperature, TF = ?'
      READ(5,*) TF
      TEMPI=TI
      TEMPF=TF
C
C --------->      Compute optimal temperature path using
C           step-by-step integration:
C
      ITERATION = 1
C
C --------->      Set dummy NSTEP for starting:
      NSTEP=50
C
C --------->      Input data necessary for numerical scheme:
C
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                             *
*                                                                             *
*             Algorism for numerical iterative method                         *
*                                                                             *
*   1. calculate optimal temperature path for linear viscoelasticity          *
*                                                                             *
*   2. continue iterative method until either                                 *
*                                                                             *
*      tolerance in stress reaches prescribed tolerance (TOLS)                *
*                                                                             *
*      or                                                                     *
*                                                                             *
*      number of iteration reaches prescribed maximum number                  *
*                                                                             *
*      of iterations (ITERMAX)                                                *
*                                                                             *
```

```
*                                                                      *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
      PRINT *,'  Do you want to debug   ?'
      PRINT*
      PRINT *, ' INPUT 1:                  For   Yes '
      PRINT *, '       0:                         No '
      READ(5,*) IDBG
C
      PRINT *,'  Input tolerance, TOLS = ?'
      READ(5,*) TOLS
C
      PRINT *,'  Input maximum number of iteration  ?'
      READ(5,*) ITERMAX
C
C      PRINT*,'DO YOU WANT TO READ FILE ? '
C      PRINT*,'YES.............. 1'
C      PRINT*,'NO ............... 0'
C      READ(5,*) IFILE
      IFILE=0
C
  900 CONTINUE
C
C --------->      Generate stress function of temperature
C           using shape preserve function:
C
      IF(IDBG.EQ.1.AND.ITERATION.NE.1)THEN
      INTV=6
      WRITE(6,602) ITERATION,NSTEP
      CALL PDBGFILE(T,TEM,STR,6,INTV,NSTEP)
      PRINT*,' Input any number to continue '
      READ(5,*)IRESUME
      ENDIF
C
      CALL STRFUNC (TEM,STR,ITERATION,NSTEP,IFILE)
C
      IF(ITERATION.EQ.1)THEN
      TILIM =TI
                                                    ELSE
      TILIM = TEM(1)
      ENDIF
      PRINT*
      PRINT*,' New TILIM is ',TILIM
      PRINT*
      PRINT*
      PRINT*,' ... input 1 to change New TILIM.'
      READ(5,*) ITL
C
      IF(ITL.EQ.1)THEN
      READ(5,*) TILIM
      PRINT*
```

```
      PRINT*,' NEW TILIM IS ',TILIM
      PRINT*
      ENDIF
C
      IF(IDBG.EQ.1.AND.ITERATION.NE.1)THEN
      INTV=6
      WRITE(6,603) ITERATION,NSTEP
      CALL PDBGFILE(T,TEM,STR,6,INTV,NSTEP)
      PRINT*,' Input any number to continue '
      READ(5,*)IRESUME
      ENDIF
C
      PRINT*,' To compute  Initial Drop Temperature,  '
      PRINT*
      PRINT *, ' INPUT 1:                  For   Yes '
      PRINT *, '       0:                         No '
      READ(5,*) IW
      IF(IW.EQ.1)THEN
      PRINT*
      PRINT*,' Computing  Initial Drop Temperature.........'
      PRINT*
C
      CALL TIDROP(TZERO)
      PRINT*
      PRINT*
      PRINT*,' Initial Drop Temperature from analysis is   ',TZERO
      PRINT*
      ENDIF
C
C ---------->      Compute optimal temperature path using step-by-step
C           integration:
C
      CALL OPTPATH(TIME,T,TEMP,TEM,STR,TDELT,TILIM,NSTEP,
     +             ITERATION,IKCON,IDBG,IFILE)
C
C ---------->      Compute optimal temperature path using analytic
C           solution for three element model:
C
      DO 10 ID=1, NSTEP
      T(ID)=TIME(NSTEP-ID+1)
      TEMPSTR(ID)=TEMP(NSTEP-ID+1)
      TEM(ID)=TEMP(NSTEP-ID+1)
      SDELT(ID)=TDELT(NSTEP-ID+1)
 10   CONTINUE
C
      DO 20 ID=1, NSTEP
      TDELT(ID)=SDELT(ID)
 20   CONTINUE
C
      T(0)=0D0
      T(NSTEP+1)=TIMEF
```

```
      TEMPSTR(0)=TI
      TEMPSTR(NSTEP+1)=TF
C
      DO 25 K= 0, NSTEP+1
      STEMP(K) = TEMPSTR(K)
  25  CONTINUE
C
C --------->      Compute final stress using numerical method:
C
      PRINT*
      PRINT*,' ....check if convergence is made. '
      PRINT*
      CALL  VSTRESS(NSTEP,T,TEMPSTR,OPTSTR,TDELT,SXI,STRTF)
C
      DO 50 I=1, NSTEP
      STR(I)=OPTSTR(I)
  50  CONTINUE
      IF(IDBG.EQ.1.AND.ITERATION.NE.1)THEN
      INTV=6
      WRITE(6,604) ITERATION,NSTEP
      CALL PDBGFILE(T,TEM,STR,6,INTV,NSTEP)
      PRINT*,' Input any number to continue'
      READ(5,*)IRESUME
      ENDIF
C
C
C --------->      Pring interim results (summary - total 20 data ):
C
      INTV=20
      WRITE(IDATD,650) ITERATION
      CALL PFILE(T,TEMPSTR,OPTSTR,IDATD,INTV,NSTEP)
      PRINT*
      PRINT*,' Do you want to plot interim results  ? '
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) IPLT
C
      IF(IPLT.EQ.1)THEN
      INTV=500
      WRITE(IDATP,601) ITERATION
      IF(NSTEP.LE.INTV)INTV=NSTEP
      CALL PFILE(T,TEMPSTR,OPTSTR,IDATP,INTV,NSTEP)
      ENDIF

C
C --------->      Check convergence here:
C
      IF(ITERATION.EQ.1) THEN
      IF(IKCON.EQ.3)THEN
      DO 3000 I=1, NSTEP
      X=TEMPSTR(I)
      DERZER(I) = DCSDER(0, X,NINTV,BREAK,CSCOEF)
```

```
      DERFIR(I) = DCSDER(1, X,NINTV,BREAK,CSCOEF)
      DERSEC(I) = DCSDER(2, X,NINTV,BREAK,CSCOEF)
 3000 CONTINUE
      INTV=NSTEP
      WRITE(IDATW,850) ITERATION,TR
      IF(NSTEP.LE.INTV)INTV=NSTEP
      CALL PDFILE(T,TEMPSTR,OPTSTR,DERZER,DERFIR,DERSEC,IDATW,
     +            INTV,NSTEP)
      ENDIF
C
      S = 1D2
      SSTRTF=STRTF
                                                        ELSE
      CALL STRCOMP(SSTRTF,STRTF,S,ITERATION)
      PRINT*
      PRINT*
      PRINT*,' Number of Iteration     = ',ITERATION
      PRINT*,' Final Stress (Previous) = ',SSTRTF
      PRINT*,' Final Stress (New)      = ',STRTF
      PRINT*,' Deviation               = ',S
      WRITE(IDATD,660) SSTRTF,STRTF,S
                       ENDIF
C
C
      IF(IKCON.EQ.3)THEN
      DO 3500 I=1, NSTEP
      X=TEMPSTR(I)
      DERZER(I) = DCSDER(0, X,NINTV,BREAK,CSCOEF)
      DERFIR(I) = DCSDER(1, X,NINTV,BREAK,CSCOEF)
      DERSEC(I) = DCSDER(2, X,NINTV,BREAK,CSCOEF)
 3500 CONTINUE
      INTV=NSTEP
      WRITE(IDATW,850) ITERATION,TR
      IF(NSTEP.LE.INTV)INTV=NSTEP
      CALL PDFILE(T,TEMPSTR,OPTSTR,DERZER,DERFIR,DERSEC,IDATW,
     +     INTV,NSTEP)
      ENDIF
      IF(IKCON.EQ.2) GOTO 950
C
      IF(ITERATION.LT.ITERMAX)THEN
      SSTRTF=STRTF
      ITERATION = ITERATION+1
      DS=DABS(S)
      IF(DS.GT.TOLS) GOTO 900
      ENDIF
C
C   --------->      Post processing:
C
  950 CONTINUE
      PRINT*,' Do you want to creat  INPUT RESTART FILE ?'
      PRINT *, ' INPUT 1:                  For   Yes '
```

```
      PRINT *, '       0:                          No '
      READ(5,*)ICR
C
      IF(ICR.EQ.1)THEN
      WRITE(IDATA,*) ITERATION,NSTEP
      DO 965 I=1, NSTEP
      WRITE(IDATA,*)TEM(I),STR(I),TDELT(I)
 965  CONTINUE
      DO 9655 K=0,NSTEP+1
 9655 WRITE(IDATA,*) T(K)
      ENDIF
C
      CALL POSTPRO(IDATB,IDATR)
      INTV=20
      WRITE(IDATB,601) ITERATION
      CALL PFILE(T,TEMPSTR,OPTSTR,IDATB,INTV,NSTEP)
C
C   --------->      DATA FILE FOR PLOTTING ( 500 DPTS):
C
      INTV=500
      WRITE(IDATP,601) ITERATION
      IF(NSTEP.LE.INTV)INTV=NSTEP
      CALL PFILE(T,TEMPSTR,OPTSTR,IDATP,INTV,NSTEP)
C
C   --------->      DATA FILE FOR RECORDING:
C
      INTV=NSTEP
      WRITE(IDATR,601) ITERATION
      CALL PFILE(T,TEMPSTR,OPTSTR,IDATR,INTV,NSTEP)
C
      PRINT*,' Output data for STRFUNC(TEMPERATURE) ...DBGDBG.DAT'
      PRINT*,' For interim Temp. and stress history ...DBGTEM.DAT'
      PRINT*,' For summary file .....DBGOPT.LUK'
      PRINT*,' For record file .....DBGOPT.RCD'
      PRINT*,' For plotting file .....DBGOPT.PLT'
C
C --------->      FORMAT:
C
 166  FORMAT('1',//)
 602  FORMAT('1',/,3X,'Before  STRFUNC...',/,
     +3X,'Number of Iteration = ',I5,
     +3X,'Number of Steps     = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
 603  FORMAT('1',/,3X,'After STRFUNC...',/,
     +3X,'Number of Iteration = ',I5,3X,
     +3X,'Number of steps  = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
 604  FORMAT('1',/,3X,'After VSTRESS...',/,
     +3X,'Number of Iteration = ',I5,3X,
     +3X,'Number of Step = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
```

```
 500  FORMAT(/,10X,'ITERATION',5X,'ADINT',10X,'A',12X,'B',
     +10X,'I.D. TEMP',14X,'FINAL TEMP',10X,'STRESS',/)
 550  FORMAT(5X,I10,3X,6(D12.6,3X))
 601  FORMAT('1',///,5X,'NUMBER OF ITERATION = ',I5,//,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/,
     +24X,'(NUMERICAL)',8X,'(NUMERICAL)',/)
 650  FORMAT('1',///,5X,'NUMBER OF ITERATION = ', I5,//,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/,
     +24X,'(NUMERICAL)',8X,'(NUMERICAL)',/)
 660  FORMAT(///,5X,' Final stress (previous) = ',D16.6,/,
     +5X,' Final stress            = ',D16.6,/,
     +5X,' Deviation (%)           = ',D16.6,///)
 750  FORMAT(//,5X,'Initial drop temperature  = ',D16.6)
 850  FORMAT('1',///,5X,'Number of iteration  = ', I5,/,
     +5X,'TR = ',F12.3,//,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',9X,'STRESS',9X,3X,
     +'1ST DERV.',9X,'2ND DERV.'/,24X,'(NUMERICAL)',8X,'(NUMERICAL)',
     +5X,'(GENERATED)',6X,'(GENERATED)',8X,'(GENERATED)',/)
      STOP
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                     SUBROUTINE TIDROP(TZERO)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: computes initial temperature drop                           +
*                                                                       +
*           (a) for thermorheologically simple material,                +
*                                                                       +
*                   To -Ti = ( a(To)/ a'(To))                           +
*                                                                       +
*           (b) for thermorheologically complex material:               +
*                                                                       +
*              To -Ti =  h(To)a(To) / ( a'(To)h(To) -h'(To)a(To) )      +
*                                                                       +
*           This subroutine calls Function  '' COPT ''                  +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/TDROP/ TEMPI,TEMPF,NSTEP
      COMMON/BBB/ TIMEF,TI,TF,TOL
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      INTEGER    ITMAX, NROOT
      REAL*8     ERRABS, ERRREL, ETA
      PARAMETER  (NROOT=1)
C
      REAL*8     OPT, X(NROOT), XGUESS(NROOT)
      EXTERNAL   OPT,  DZREAL
C
```

```
C -----> Calculate initial drop temperature:
C
      OPEN(UNIT=7,STATUS='UNKNOWN',FILE='TDROP.DAT')
C
      EPSS    = 1.0E-5
      ERRABS = 1.0E-5
      ERRREL = 1.0E-5
      ETA    = 1.0E-5
      ITMAX  = 500
      STZERO= TI
      XGUESS(1)=STZERO
C
 900      CALL DZREAL (OPT, ERRABS, ERRREL, EPSS, ETA, NROOT,
     +         ITMAX, XGUESS, X, INFO)
C
      WRITE(7,600) X(1),ITMAX
      WRITE(6,600) X(1),ITMAX
      PRINT*
      PRINT*,' Input any number to continue'
      READ(5,*)IRESUME
      TR=X(1)
C
      IF(IH.EQ.1) THEN
      AFUNCL= ATL(TR)
      AFUNC = AT (TR,AFUNCL)
      ADERFL= ATFL(TR,AFUNCL)
      ADERF = ATF (TR,AFUNCL,ADERFL)
C
      EQLS=TR-TI
      EQRS=HT(TR)*AFUNC/(ADERF*HT(TR)-DTDHT(TR)*AFUNC)
C
      WRITE(7,660) EQLS, EQRS
      WRITE(6,660) EQLS, EQRS
C
                                                        ELSE
      EQLS=TR-TI
      EQRS=-(TR+273.3)*(TR+273.3)/1000/DAT(TR)/DLOG(10D0)
      WRITE(6,650)EQLS,EQRS
      WRITE(7,650)EQLS,EQRS
      ENDIF
C
      PRINT*,' Want to change; 1 ... FOR YES '
      READ(5,*) ICHG
C
      IF(ICHG.EQ.1)THEN
      PRINT*,' INPUT ITMAX = ?'
      READ(5,*) ITMAX
      GOTO 900
      ENDIF
C
C --------->      FORMAT:
```

```
C
 600  FORMAT(///,5X,' The best approximation to zero of function is',
     +' equal to',D16.6,/,5X,' The number of function evaluations',
     +' required was ',I10,'.',//)
 660  FORMAT(5X,' TO-TI =',F16.6,/,
     +       5X,' h(To)a(To) / ( Da(To)h(To) -Dh(To)a(To) ) = ',
     +       F16.6 )
 650  FORMAT(5X,'TO - TI =',F16.6,/,5X,'AT(TO)/AT''(TO)=',F16.6)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION OPT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/BBB/ TIMEF,TI,TF,TOL
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      EXTERNAL DAT
      EXTERNAL ATL,AT,ATFL,ATF,HT,DTDHT
C
      IF(IH.EQ.1)THEN
      AFUNCL= ATL(T)
      AFUNC = AT (T,AFUNCL)
      ADERFL= ATFL(T,AFUNCL)
      ADERF = ATF (T,AFUNCL,ADERFL)
C
      EQLS=T-TI
      EQRS=HT(T)*AFUNC/(ADERF*HT(T)-DTDHT(T)*AFUNC)
      OPT=EQLS-EQRS
      RETURN
                                                        ELSE
      EQLS=T-TI
      EQRS=-(T+273.3)*(T+273.3)/1000/DAT(T)/DLOG(10D0)
      OPT=EQLS-EQRS
      RETURN
      ENDIF
      END
*

* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
               SUBROUTINE STRCOMP(X,Y,S,ITERATION)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      S = (Y-X )/X*100.0
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
        SUBROUTINE STRFUNC (TEMP,STRESS,ITERATION,NSTEP,IFILE)
```

```
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
*  Purpose: correlate temperature  and stress along the temperature path
*           using cubic spline interpolation function
*
      PARAMETER(NDATA=5000,MDATA=5000)
      PARAMETER(IDATC=39)
      REAL*8  BREAK(2*NDATA), CSCOEF(4,2*NDATA), STRESS(NSTEP),
     &        TEMP(NSTEP),SSTRESS(MDATA),STEMP(MDATA)
      COMMON/CCC/ BREAK, CSCOEF, NINTV,JDBUG
      EXTERNAL    DCSCON,DCSVAL,DCSDER
C
      NINTV=NSTEP-1
C
      DO 10 I=1, NSTEP
      SSTRESS(I)=STRESS(I)
      STEMP(I)=TEMP(I)
  10  CONTINUE

C
C --------->      Create interpolation function:
C
      IF(IFILE.EQ.0.AND.ITERATION.EQ.1)THEN
      NTOT = 2*NDATA
      DO 100 I= 1, NTOT
      BREAK(I)=0D0
      DO 100 J=1,4
      CSCOEF(J,I)=0D0
 100  CONTINUE
      RETURN
      ENDIF
C
   50 NINTV=NSTEP-1
C
C
      CALL DCSCON (NSTEP, TEMP, STRESS, IBREAK, BREAK, CSCOEF)
C
C --------->      Evaluate first derivate and eliminate positives
C           and reconstruct shape reserve function:
C
C
      ICHK=0
      DO 200 I= 1, NSTEP
      X=STEMP(I)
      DERFIR = DCSDER(1, X,NINTV,BREAK,CSCOEF)
C
      IF(DERFIR.GE.0D0)ICHK=I
  200 CONTINUE
C
      IF(ICHK.NE.0)THEN
      PRINT*
```

```
      PRINT*, '  ... STRFUNC : '
      PRINT*
      PRINT*, '  D(SIGMA)/DT > 0,  when temperature is  ',X
      PRINT*, '  ICHK/NSTEP = ',ICHK,' / ',NSTEP
      PRINT*
      PRINT*,' To see results, INPUT 1 :'
      READ(5,*) ISEE
      IF(ISEE.EQ.1)THEN
      WRITE(6,650)
      NCHK=ICHK+5
      IF(NCHK.GE.NSTEP) NCHK=NSTEP
      DO 275 K=1, NCHK
      X=STEMP(K)
      DERFIR = DCSDER(1, X,NINTV,BREAK,CSCOEF)
      WRITE(6,700) X, DERFIR
 275  CONTINUE
      ENDIF
C
      PRINT*
      PRINT*,'  To reconstruct SIGMA Function, Input 1 '
      READ(5,*) ICONST
C
      IF(ICONST.EQ.1)THEN
      ICHK=ICHK+5
      INUM=0
      DO 250 I=ICHK,NSTEP
      INUM=INUM+1
      TEMP(INUM)=STEMP(I)
      STRESS(INUM)=SSTRESS(I)
  250 CONTINUE
      NSTEP=INUM-1
C
C
C --------->      Message:
C
      PRINT*
      PRINT*, ' During evaluation of  Stress  Function .....'
      PRINT*
      PRINT*, ' NSTEP = ',INUM
C
      GOTO 50
      ENDIF
      ENDIF
C
C --------->      FORMAT:
C
 650  FORMAT(//,5X,'TEMPERATURE', 10X,'FIRST DERIVATIVE',/)
 700  FORMAT(5X,F12.3,5X,D16.6)
      RETURN
      END
*
```

```
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          SUBROUTINE TEMPADJ(TEMP,TADJ,NSTEP,JHEADTAIL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                            +
*  Purpose: generate new temperature path based on a given temperature       +
*           path  with required new new initial and final temperatures       +
*           using linear interpolation.                                      +
*                                                                            +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(ND=5000)
      REAL*8 TEMP(ND),RTEMP(ND)
C
      IF(JHEADTAIL.EQ.0)THEN
      TMAX = TEMP(NSTEP)-TADJ
      DO 100 K=1, NSTEP
      ARGA = (TEMP(K)-TEMP(1))/(TEMP(NSTEP)-TEMP(1))
      RTEMP(K)=TEMP(K)-ARGA*TMAX
 100  CONTINUE
                                                        ELSE
      TMAX = TEMP(1)-TADJ
      DO 200 K=1, NSTEP
      ARGA = (TEMP(K)-TEMP(NSTEP))/(TEMP(1)-TEMP(NSTEP))
      RTEMP(K)=TEMP(K)-ARGA*TMAX
 200  CONTINUE
      ENDIF
      DO 300 K=1, NSTEP
      TEMP(K)=RTEMP(K)
 300  CONTINUE
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            DOUBLE PRECISION FUNCTION XINT(A,B,XK)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
       IMPLICIT REAL*8(a-h,o-z)
C
       CONST=dLOG10(dEXP(1D0))
       UPP=-A*XK-B
       XINT=CONST*10**UPP/(-A)
       RETURN
       END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION QXI(XA,XB,A,B)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
```

```
      COMMON/STEP/TIMEA,TIMEB
      EXTERNAL XINT
      UPER=XINT(A,B,Xb)
      BOTT=XINT(A,B,Xa)
      QXI=(UPER-BOTT)*(TIMEb-TIMEa)/(Xb-Xa)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION DAT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
CHG
      PARAMETER(M=4)
      REAL*8 ATLG(M),TRCP(M),SLOPE(M)
      DATA ATLG/6.,2.5,-8.,-10./
      DATA TRCP/3.3,2.69,2.1,1.6/
CHG      PARAMETER(M=4)
CHG      REAL*8 ATLG(M),TRCP(M),SLOPE(M)
CHG
CHGDATA      DATA ATLG/6.,2.5,-4.,-5.0/
CHGDATA      DATA TRCP/3.3,2.69,2.32,1.6/
C
      Y=1000./(T+273.3)
C
      DO 5 I=1, M-1
      SLOPE(I) =(ATLG(I)-ATLG(I+1))/(TRCP(I)-TRCP(I+1))
 5    CONTINUE

C
      IF(Y.GE.TRCP(1)) THEN
      DAT = SLOPE(1)
      RETURN
C
                                  ELSEIF(Y.LE.TRCP(M)) THEN
      DAT = SLOPE(M-1)
      RETURN
      ENDIF
C
      DO 10 I=1, M-1
C
      IF(Y.LT.TRCP(I).AND.Y.GT.TRCP(I+1)) THEN
      DAT = SLOPE(I)
      RETURN
      ENDIF
 10   CONTINUE
C
      IF(Y.EQ.TRCP(2)) DAT=(SLOPE(1)+SLOPE(2))/2.0
      IF(Y.EQ.TRCP(3)) DAT=(SLOPE(2)+SLOPE(3))/2.0
      RETURN
```

```
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                 DOUBLE PRECISION FUNCTION cfit(x)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      real*8 b(4)
      COMMON/CURV/B,CSB,STIME
      cfit=b(1)+(b(2)+(b(3)+b(4)*x)*x)*x
      return
      end
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                 DOUBLE PRECISION FUNCTION  cder1(x)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      real*8 b(4)
      COMMON/CURV/B,CSB,STIME
      cder1=b(2)+(2*b(3)+3*b(4)*x)*x
      return
      end
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                 DOUBLE PRECISION FUNCTION cder2(x)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      real*8 b(4)
      COMMON/CURV/B,CSB,STIME
      cder2=2*b(3)+6*b(4)*x
      return
      end

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE OPTPATH(TIME,TIM,TEMP,TEM,STR,TDELT,TILIM,NSTEP,
     +           ITERATION,IKCON,IDBUG,IFILE)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                      +
*  Purpose: computes the intermediate temperatures Ti numerically,     +
*           and numerically (but manually) until we obtain T(0) = T0   +
*                                                                      +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(ND=5000)
      PARAMETER(IDATD=19,IDATP=11,IDT=35,IDATE=21,IDATW=30,
     +          IDATX=31,IDATY=32,IDATZ=33,IDATV=34)
C
      REAL*8 TIME(ND),TEMP(ND),TDELT(ND),SDELT(ND),DELTS(ND)
```

```
      REAL*8 T(0:ND),OPTSTR(0:ND),TEMPSTR(0:ND),STEMP(0:ND),TIM(0:ND),
     +       TD(500),ST(500),SS(500),FT(500),
     +       DERZER(ND),DERFIR(ND),DERSEC(ND),SXI(ND)
C
      PARAMETER(NDATA=5000)
      REAL*8  BREAK(2*NDATA), CSCOEF(4,2*NDATA), STR(NDATA),
     &        TEM(NDATA)
      COMMON/CCC/ BREAK, CSCOEF, NINTV, JDBUG
C
      PARAMETER(M=3)
      REAL*8 ATLG(M),TRCP(M)
      COMMON/VATDATA/ATLG,TRCP
C
      COMMON/CISML/ERRABS,ERRREL,IRULE
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      COMMON/PPP/EDERS,EDERF,TOP,BOT
      COMMON/BBB/ TIMEF,TI,TF,TOL
C
C --------->      Initialization:
C
      IRULE=2
      ERRABS=0.D0
      ERRREL=1D-3
      NST=NSTEP
C
      DO 9999 K=1, NST
 9999 DELTS(K)=TDELT(K)
C
      IF(IDBUG.EQ.1.AND.ITERATION.NE.1)THEN
      INTV=6
      WRITE(6,602) ITERATION,NST
      CALL PDBGFILE(T,TEM,STR,6,INTV,NSTEP)
      PRINT*,' Input  any number to continue'
      READ(5,*)IRESUME
      ENDIF
C
      IF(ITERATION.GE.2) GOTO 900
C
C --------->      Input data using INTERACTIVE SCREEN MODE:
C
C
      PRINT*
      PRINT*,'Choose integration scheme for reduced time ;IADP=?'
      PRINT*
      PRINT*,'1        For General Purpose Adaptive Rule;DQDAG'
      PRINT*,'0        For General Purpose Adaptive Rule and Point',
     +' Singularity;DQDAGS'
      READ(5,*) IADP
C
      PRINT*
```

```
      PRINT*,'Input Starting INITIAL TIME;STIME=?'
      PRINT*,'  <  recommendation, STIME=1D-5  >'
      READ(5,*) STIME
C
      PRINT*
      PRINT*,'Input  Final Time;TIMEF=?'
      READ(5,*) TIMEF
C
      PRINT*
      PRINT*,' Input  Allowable Temperature Gradient; DTEGAL= ?'
      READ(5,*) DTEGAL
C
      PRINT*
      PRINT*,'Input  Maximum  Time Step Size; DELTMAX=  ?'
      PRINT*,'   NOTE... INPUT 0 TO NO LIMIT'
      READ(5,*) DELTMAX
C
      PRINT*
      PRINT*,'Input  Minimum Time Step Size; DELTMIN = ?'
      PRINT*,'   NOTE... INPUT 0 TO NO LIMIT'
      READ(5,*) DELTMIN
C
      PRINT*
      PRINT*,' Input Number of Steps for brief report;ICSTEP = ?'
      READ(5,*) ICSTEP
C
C
 900  IFLAG = 0
      ITD=1
      SS(ITD)=0D0
  6   PRINT*
      PRINT*,' Is this for investigation stage ? '
      PRINT*
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) INV
      PRINT*
C      READ(5,*) IDBG
       IDBG =0
C
C      PRINT*
C      PRINT*,' Do you want to print data for < AT, ATF,ATS >  ? '
C      PRINT*,' INPUT   1 ................. FOR YES '
C      PRINT*,'         0 .................     NO  '
C      READ(5,*) JDBUG
       JDBUG=0
C
      PRINT*,' To change data for computation, input  1:'
      READ(5,*) ICHG
      IF(ICHG.NE.1) GOTO 991
C
```

```
      PRINT*
      PRINT*,' Supply or change data for Allowable Temp. Gradient ? '
      PRINT*
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) IC
C
      IF(IC.EQ.1)THEN
      PRINT*
      PRINT*,' OLD DTEGAL = ', DTEGAL
      PRINT*,' ALLOWABLE TEMPERATURE GRADIENT; DTEGAL=?'
      READ(5,*) DTEGAL
      PRINT*,' NEW DTEGAL = ',DTEGAL
      ENDIF
C
      PRINT*
      PRINT*,' Supply or change data for Minimum time step size ? '
      PRINT*
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) IC
C
      IF(IC.EQ.1)THEN
      PRINT*
      PRINT*,' OLD DELTMIN = ', DELTMIN
      PRINT*
      PRINT*,' Input Minimum Time Step Size ?'
      PRINT*,'   NOTE... INPUT 0 TO NO LIMIT'
      READ(5,*) DELTMIN
      PRINT*
      PRINT*,' NEW DELTMIN = ',DELTMIN
      ENDIF
C
      IF(IH.EQ.1)THEN
      PRINT*
      PRINT*,' Supply or change data for vertical shift factor ? '
      PRINT*
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) JVAT
C
      IF(JVAT.EQ.1)THEN
      DO 876 MMM=1, 3
      PRINT*
      PRINT*,'ATLG(',MMM,') =  ? ','  TRCP(',MMM,') = ? '
      READ(5,*)   ATLG(MMM),TRCP(MMM)
 876  CONTINUE
      ENDIF
      ENDIF
C
 991  CONTINUE
```

```
C
      IF(INV.EQ.0) THEN
      PRINT*
      PRINT*,' Input lower bound of initial guess temperature;ATEMI=?'
      PRINT*,' and final ending temperature                  ;ATEMF=?'
      READ(5,*) ATEMI,ATEMF
      PRINT*
      PRINT*,' Input upper bound of initial guess temperature;BTEMI=?'
      PRINT*,' and final ending temperature                  ;BTEMF=?'
      READ(5,*) BTEMI,BTEMF
      PRINT*
      PRINT*,' Input required  Initial Drop Temperature;TDROP=?'
      READ(5,*) TDROP
C
C --------->      Iinital guess for final temperature:
C
      TEMPF=(BTEMI-ATEMI)*(TDROP-ATEMF)/(BTEMF-ATEMF)+ATEMI
C
      GOTO 16
      ENDIF
C
      PRINT*
      PRINT*,'  Input Initial guess Temperature :      TEMPF=?'
      READ(5,*) TEMPF
C
C
C --------->     Initialization:
C
  16  TB=TEMPF
      XB=TIMEF
      TIME(1)=TIMEF
      TEMP(1)=TB
      XI=1D-5
      KK=2
      ISTEP=1
C
 10   CALL CALTEMP(DELT,XI,XTEMP,XTIME,DRVT,TILIM,IFLAG,NSTEP,IDBG)
C
      IF(IFLAG.NE.0)THEN
      IF(IFLAG.EQ.1)THEN
      PRINT*
      WRITE(6,*)' Warning .... DRVT becomes too large. '
      WRITE(6,*)'            BOT = ',BOT
      PRINT*
      ENDIF
C
      IF(IFLAG.EQ.2)THEN
      PRINT*
C      TILIM=400
      WRITE(6,*)' Warning .... Temperature reaches  ',TILIM
      PRINT*
```

```
      ENDIF
C
      WRITE(6,*)' Input 1: To continue to compute optimal temp. path'
      PRINT*,   '       2:    start with new  STRESS FUNCTION (TEMP.) '
      READ(5,*) IKON
C
      IF(IKON.EQ.2)THEN
      PRINT*
      PRINT*,' Input  <1> for  HEAD... <0> for TAIL: JHEADTAIL = ?'
      READ(5,*) JHEADTAIL
      PRINT*,' Input  Temperature  to be adjusted: TADJ = ?'
      READ(5,*) TADJ
      CALL TEMPADJ (TEM,TADJ,NST,JHEADTAIL)
C
      TEMPSTR(0)=TI
      TEMPSTR(NSTEP+1)=TF
      DO 8888 K= 1, NST
      TEMPSTR(K) = TEM(K)
8888  CONTINUE
      PRINT*
      PRINT*,' ....check if convergence is made. '
      PRINT*
      CALL  VSTRESS(NST,TIM,TEMPSTR,OPTSTR,DELTS,SXI,STRTF)
      DO 7777 I=1, NST
      STR(I)=OPTSTR(I)
7777  CONTINUE
C
      IF(IDBUG.EQ.1)THEN
      INTV=6
      WRITE(6,603) NST
      CALL PDBGFILE(T,TEM,STR,6,INTV,NST)
      PRINT*,' INPUT ANY NUMBER TO CONTINUE'
      READ(5,*)IRESUME
      ENDIF
C
      CALL STRFUNC (TEM,STR,ITERATION,NST,IFILE)
C
      IF(ITERATION.EQ.1)THEN
      TILIM =TI
      ELSE
      TILIM = TEM(1)
      ENDIF
      PRINT*
      PRINT*,' NEW TILIM IS ',TILIM
      PRINT*
      PRINT*,' ... input 1 to change New TILIM.'
      READ(5,*) ITL
C
      IF(ITL.EQ.1)THEN
      READ(5,*) TILIM
      PRINT*
```

```
      PRINT*,' NEW TILIM IS ',TILIM
      PRINT*
      ENDIF
C
      IF(IDBUG.EQ.1)THEN
      DO 3000 I=1, NST
      X=TEM(I)
      DERZER(I) = DCSDER(0, X,NINTV,BREAK,CSCOEF)
      DERFIR(I) = DCSDER(1, X,NINTV,BREAK,CSCOEF)
      DERSEC(I) = DCSDER(2, X,NINTV,BREAK,CSCOEF)
 3000 CONTINUE
      INTV=6
      WRITE(6,2850)
      CALL PSCRFILE(TEM,STR,DERZER,DERSEC,6,
     +     INTV,NST)
      PRINT*,'Input any number to continue ...'
      READ(5,*) IRESUME
C
      INTV=500
      WRITE(IDT,1850)
      IF(NST.LE.INTV)INTV=NST
      CALL PDFILE(TIME,TEM,STR,DERZER,DERFIR,DERSEC,IDT,
     +     INTV,NST)
C
      INTV=6
      WRITE(6,604) ITERATION,NST
      CALL PDBGFILE(T,TEM,STR,6,INTV,NST)
C
      READ(5,*)IRESUME
      ENDIF
      ENDIF

C
      PRINT*
      WRITE(6,*)' Previous guess of final temperature  = ',TEMPF
      IFLAG = 0
      GOTO 6
      ENDIF
C
      TEMP(KK)=XTEMP
      TIME(KK)=XTIME
      TDELT(KK)=DELT
C
      IF(TIME(KK).GT.0D0) THEN
      ISTEP=ISTEP+1
      ICOUNT=ISTEP/ICSTEP
      ICOUNT=ISTEP-ICOUNT*ICSTEP
C
      IF(ICOUNT.EQ.0) THEN
        PRINT*
        PRINT*,       '<<< NUMBER OF STEPS = ', ISTEP, ' >>>'
        WRITE(   6,*) '      TIME = ',TIME(KK)
```

```
        WRITE(   6,*) '      TEMP = ',XTEMP
      ENDIF
C
      KK=KK+1
      GOTO 10
                                                        ELSE
      NSTEP=ISTEP
C
      ARGA=(TEMP(KK)-TEMP(KK-1))/(TIME(KK)-TIME(KK-1))
      XTEMP=ARGA*(-TIME(KK-1))+TEMP(KK-1)
      TEMP(KK)=XTEMP
      TIME(KK)=0D0
C
      PRINT*
      NSTEP=KK
      WRITE(6,*) '       NSTEP=' , NSTEP
      WRITE(6,*) '       TIME= ',TIME(KK)
      WRITE(6,*) '       TEMP= ',XTEMP
C
 685  CONTINUE
      PRINT*
      PRINT*,' Do you want to adjust temperature path ?'
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) IADJ
C
      IF(IADJ.EQ.1)THEN
      PRINT*
      PRINT*,' Input  <0> for HEAD... <1> for TAIL: JHEADTAIL = ?'
      READ(5,*) JHEADTAIL
      PRINT*,' Input temperature to be adjusted; TADJ = '
      READ(5,*) TADJ
      CALL TEMPADJ (TEMP,TADJ,NSTEP,JHEADTAIL)
      ENDIF
C
      DP = 100.0
      IF(INV.EQ.0) DP=(TDROP-XTEMP)/TDROP
      DP=DABS(DP)*100
      PRINT*
      WRITE(6,*) '     SEED            TEMP.  = ',TEMP(1)
      WRITE(6,*) '     INITIAL DROP    TEMP.  = ',TEMP(NSTEP)
      WRITE(6,*) '     PREDICTED  I.D. TEMP.  = ',XTEMP
      WRITE(6,*) '     MINIMUM TIME STEP SIZE = ',CHKMIN
      PRINT*
      WRITE(6,*) '     % Difference           = ',DP
      PRINT*
      PRINT*
      PRINT*,' INPUT 0: To compute final stress:    '
      PRINT*,'       1:    continue   GUESS AND SHOT: '
      PRINT*,'       2:    start new game:'
      PRINT*,'       3:    print'
```

```
      PRINT*,'       4:    compute new stress function (temp): '
      PRINT*
      READ(5,*) ICONT
C
      IF(ICONT.EQ.4)THEN
      PRINT*
      PRINT*,' Input <1> for HEAD... <0> for TAIL: JHEADTAIL = ?'
      READ(5,*) JHEADTAIL
      PRINT*,' Input final temperature to be adjusted: TADJ = '
      READ(5,*) TADJ
      CALL TEMPADJ (TEM,TADJ,NST,JHEADTAIL)
C
      TEMPSTR(0)=TI
      TEMPSTR(NSTEP+1)=TF
      DO 6666 K= 1, NST
      TEMPSTR(K) = TEM(K)
6666  CONTINUE
      PRINT*
      PRINT*,' ....check if convergence is made. '
      PRINT*
      CALL  VSTRESS(NST,TIM,TEMPSTR,OPTSTR,DELTS,SXI,STRTF)
      DO 5555 I=1, NST
      STR(I)=OPTSTR(I)
5555  CONTINUE
C
         CALL STRFUNC (TEM,STR,ITERATION,NST,IFILE)
C
      IF(ITERATION.EQ.1)THEN
      TILIM =TI
                                                        ELSE
      TILIM = TEM(1)
      ENDIF
C
      PRINT*
      PRINT*,' NEW TILIM IS ',TILIM
      PRINT*
      PRINT*,' ... input 1 to change New TILIM.'
      READ(5,*) ITL
C
      IF(ITL.EQ.1)THEN
      READ(5,*) TILIM
      PRINT*
      PRINT*,' New TILIM is ',TILIM
      PRINT*
      ENDIF
C
      IF(IDBUG.EQ.1)THEN
      DO 3500 I=1, NST
      X=TEM(I)
      DERZER(I) = DCSDER(0, X,NINTV,BREAK,CSCOEF)
      DERFIR(I) = DCSDER(1, X,NINTV,BREAK,CSCOEF)
```

```
      DERSEC(I) = DCSDER(2, X,NINTV,BREAK,CSCOEF)
 3500 CONTINUE
      INTV=NST
      WRITE(IDT,1850)
      IF(NST.LE.INTV)INTV=NST
      CALL PDFILE(TIME,TEM,STR,DERZER,DERFIR,DERSEC,IDT,
     +     INTV,NST)
      INTV=6
      WRITE(6,2850)
      CALL PSCRFILE(TEM,STR,DERZER,DERSEC,6,
     +     INTV,NST)
      PRINT*
      PRINT*,' Input any number to continue'
      READ(5,*)IRESUME
      INTV=6
      WRITE(6,605) NST
      CALL PDBGFILE(T,TEM,STR,6,INTV,NST)
C
      PRINT*,' Input any number to continue'
      READ(5,*)IRESUME
      ENDIF
      GOTO 6
      ENDIF
      IF(ICONT.EQ.3) GOTO 990
      IF(ICONT.EQ.2) GOTO 6
C
      IF(ICONT.EQ.1)THEN
      PRINT*,' Do you want to print for debugging ? '
      PRINT *, ' INPUT 1:                    For   Yes '
      PRINT *, '       0:                           No '
      READ(5,*) IDBG
C
      IF(IDBG.EQ.1)THEN
      WRITE(IDATE,5900) ITERATION,TEMPF
      WRITE(IDATX,6000) ITERATION,TEMPF
      WRITE(IDATY,6100) ITERATION,TEMPF
      WRITE(IDATZ,6200) ITERATION,TEMPF
      WRITE(IDATV,6300) ITERATION,TEMPF
      ENDIF
      ENDIF
C
C
      IF(ICONT.EQ.1) THEN
C
      IF(XTEMP.GE.TDROP) THEN
C
      IF(ATEMF.GE.TDROP)THEN
      BTEMI=ATEMI
      BTEMF=ATEMF
      ATEMI=TEMPF
      ATEMF=XTEMP
```

```
                                ELSEIF(BTEMF.LE.TDROP)THEN
      ATEMI=BTEMI
      ATEMF=BTEMF
      BTEMI=TEMPF
      BTEMF=XTEMP
                                ELSEIF(BTEMF.GT.XTEMP)THEN
      BTEMI=TEMPF
      BTEMF=XTEMP
      ENDIF
                                                        ELSE
C
      IF(ATEMF.GE.TDROP)THEN
      BTEMI=ATEMI
      BTEMF=ATEMF
      ATEMI=TEMPF
      ATEMF=XTEMP
                                ELSEIF(BTEMF.LE.TDROP)THEN
      ATEMI=BTEMI
      ATEMF=BTEMF
      BTEMI=TEMPF
      BTEMF=XTEMP
                                ELSEIF(ATEMF.LT.XTEMP)THEN
      ATEMI=TEMPF
      ATEMF=XTEMP
      ENDIF
      ENDIF
C
      TEMPF=(BTEMI+ATEMI)/2.
      PRINT*
      WRITE(6,*) ' Initial guess temperature         = ',TEMPF
      WRITE(6,*) ' Lower bound of guess temperature  = ',ATEMI
      WRITE(6,*) ' and ending temperature            = ',ATEMF
      PRINT*
      WRITE(6,*) ' Upper bound of guess temperature  = ',BTEMI
      WRITE(6,*) ' and ending temperature            = ',BTEMF
      PRINT*
      PRINT*,' To continue, input any number '
      read(5,*)
      GOTO 16
                                     ELSEIF(ICON.EQ.2)THEN
      GOTO 6
                                                        ELSE
C
C --------->      Calculate final stress:
C
      DO 710 ID=1, NSTEP
      T(ID)=TIME(NSTEP-ID+1)
      TEMPSTR(ID)=TEMP(NSTEP-ID+1)
      SDELT(ID)=TDELT(NSTEP-ID+1)
 710  CONTINUE
C
```

```
      T(0)=0D0
      T(NSTEP+1)=TIMEF
      TEMPSTR(0)=TI
      TEMPSTR(NSTEP+1)=TF
      PRINT*
      PRINT*,' ....No Need to check if convergence is made. '
      PRINT*
      CALL  VSTRESS(NSTEP,T,TEMPSTR,OPTSTR,SDELT,SXI,STRTF)
      TD(ITD)=TEMP(NSTEP)
      FT(ITD)=TEMP(1)
      ST(ITD)=STRTF
C
      IF(ITD.GT.1)THEN
      CALL STRCOMP(ST(ITD-1),ST(ITD),S,ITD)
      SS(ITD)=S
      WRITE(6,850)ITD, ST(ITD),S
      ENDIF
C
      PRINT*
      WRITE(6,800)
C
      DO 730 I=1, ITD
      WRITE(6,825) I,TD(I),FT(I),ST(I)
 730  CONTINUE
C
      ITD=ITD+1
 735  PRINT*
      PRINT*,'Input 0: To  go for new iteration without record '
      PRINT*,'      1:     search for new I.D. Temperature '
      PRINT*,'      2:     print and stop '
      PRINT*,'      3:     go for new iteration with record  '
      PRINT*,'      4:     continue adjusting temperature path '
      PRINT*,'      5:     serch optimal path by SERCH '
      READ(5,*) IKCON
C
      IF(IKCON.EQ.5) THEN
      CALL SERCH
      GOTO 735
      ENDIF
C
      IF(IKCON.EQ.4)GOTO 685
      IF(IKCON.EQ.0.OR.IKCON.EQ.3)THEN
      INTV=10
      WRITE(6,6850)
      CALL PSCRFILE(T,SXI,TEMPSTR,OPTSTR,6,
     +            INTV,NSTEP)
      ENDIF
C
      IF(IKCON.EQ.1)THEN
      GOTO 6
      ELSE
```

```
      RETURN
      ENDIF
      ENDIF
      ENDIF
 990  CONTINUE
C
C --------->      FORMAT:
C
602  FORMAT('1',/,3X,'In OPTPATH ...',/,
     +3X,'Number of Iteration  = ',I5,3X,
     +3X,'Number of step       = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
 603  FORMAT('1',/,3X,'After adjusting ...',/,
     +3X,'Number of steps = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
1850  FORMAT('1',//,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',9X,'STRESS',9X,
     +'1ST DERV.',9X,'2ND DERV.'/,24X,'(NUMERICAL)',8X,'(NUMERICAL)',
     +8X,'(GENERATED)',8X,'(GENERATED)',8X,'(GENERATED)',/)
2850  FORMAT('1',//,
     +9X,'TEMPERATURE',9X,'STRESS',9X,'STRESS',9X,
     +'2ND DERV.'/,24X,'(NUMERICAL)',
     +8X,'(GENERATED)',8X,'(GENERATED)',/)
 604  FORMAT('1',/,3X,'After  STRFUNC in OPTPATH...',/,
     +3X,'Number of Iteration = ',I5,3X,
     +3X,'Number of steps     = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
      PRINT*,' Input any number to continue'
  660 FORMAT(/,5X,'EDERS = ',D16.6,5X,'EDERF = ',D16.6,
     +/,5X,'TOP=',D16.6,5X,'BOT=',D16.6,/,5X,'DRVT=',D16.6)
 605  FORMAT('1',/,3X,'After adjusting ...',/,
     +3X,'Number of steps = ',I5,/,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/)
 850  FORMAT(///,5X,'ITERATION',5X,'FINAL STRESS',5X,
     +       'PERCENT DIFF.',/,5X,I5,5X,F12.3,5X,F12.6)
 800  FORMAT('1',///,5X,'ITERATION',5X,'I.D. TEMP',9X,'F.D. TEMP',
     +12X,'FINAL STRESS',/)
 825  FORMAT(5X,I5,5X,F12.3,5X,F16.9,2(5X,D16.6))
6850  FORMAT('1',//,11X,'TIME       ',5X,'XI     ',8X,'TEMPERATURE',
     +              10X, 'STRESS ',/)
 601  FORMAT('1',///,5X,'NUMBER OF ITERATION = ',I5,//,
     +9X,'TIME',11X,'TEMPERATURE',9X,'STRESS',/,
     +               24X,'(NUMERICAL)',8X,'(NUMERICAL)',/)
 5900 FORMAT('1',///,5X,'NONLINEAR PARAMETER = ',F12.6,/,
     +5X,'ITERATION = ',I5,/,5X,'SEED TEMPERATURE = ',
     +F12.6,//,5X,'TIME',10X,'TEMP.',10X,'DRVT',12X,'SDRVT',
     +10X,'DELT',10X,'XI      ',12X,'X',/)
 6000 FORMAT('1',///,5X,'NONLINEAR PARAMETER = ',F12.6,/,
     +5X,'ITERATION = ',I5,/,5X,'SEED TEMPERATURE = ',
     +F12.6,//,5X,'TIME',5X,'EDERS',10X,'ADERF',10X,'TOP',
     +12X,'EDERF',10X,'ADERS',10X,'AFUNC',12X,'BOT',10X,'DRVT',/)
```

```
 6100 FORMAT('1',///,5X,'NONLINEAR PARAMETER = ',F12.6,/,
     +5X,'ITERATION = ',I5,/,5X,'SEED TEMPERATURE = ',F12.6,//,
     +5X,'TIME',10X,'TEMP',10X,'SIGX',10X,'DUMMY',10X,'AFUNC',/)
 6200 FORMAT('1',///,5X,'NONLINEAR PARAMETER = ',F12.6,/,
     +5X,'ITERATION = ',I5,/,5X,'SEED TEMPERATURE = ',F12.6,//,
     +5X,'TIME',10X,'TEMP',10X,'CONSF',10X,'AFUNC',10X,'ADERF',/)
 6300 FORMAT('1',///,5X,'NONLINEAR PARAMETER = ',F12.6,/,
     +5X,'ITERATION = ',I5,/,5X,'SEED TEMPERATURE = ',F12.6,//,
     +5X,'TIME',10X,'TEMP',10X,'CONSTF',10X,'CONSTS',7X,'AFUNC',
     +10X,'ADERF',10X,'ADERS',/)
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE CALTEMP(DELT,XI,TEMP,TIME,DRVT,TILIM,IFLAG,NSTEP,IDBG)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: performs the integral to get `` reduced time '' along       +
*           the temperature at each time step.                          +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON/CISML/ERRABS,ERRREL,IRULE
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      EXTERNAL CXI,DQDAG,DQDAGS
      COMMON/BBB/ TIMEF,TI,TF,TOL
C
      CALL TDER(XI,TB,XB,DRVT,SDRVT,IFLAG,IDBG)
      IF(IFLAG.EQ.1) RETURN
      DELT=DABS(DTEGAL/DRVT)
C
      IF(DELT.LT.CHKMIN)CHKMIN=DELT
      IF(DELTMAX.EQ.0D0) GOTO 100
      IF(DELT.GT.DELTMAX)DELT=DELTMAX
 100  CONTINUE
C
      IF(DELTMIN.EQ.0D0) GOTO 110
      IF(DELT.LT.DELTMIN)DELT=DELTMIN
 110  CONTINUE
C
      XA=XB-DELT
      TA=TB-DELT*DRVT
C
      IF(TA.GE.TILIM)THEN
      IFLAG=2
      RETURN
      ENDIF
CHG E
```

```
      IF(XA.LT.0D0)THEN
      TA=(TB-TA)*XB/(XB-XA)+TB
      XA=0.0
      ENDIF
C
      IF(XB.EQ.XA)THEN
      PRINT*,' ERROR...XA=XB. AT ROUTINE CALTEMP'
      PRINT*,' XB = ',XB,'  XA = ',XA
      PRINT*,' TB = ',TB,'  TA = ',TA
      PRINT*,' X  = ',X
      READ(5,*) ICONT
      ENDIF
C
      IF(IADP.EQ.1) THEN
      CALL DQDAG(CXI,XA,XB,ERRABS,ERRREL,IRULE,X,ERREST)
                                        ELSEIF(IADP.EQ.0) THEN
      CALL DQDAGS(CXI,XA,XB,ERRABS,ERRREL,X,ERREST)
      ENDIF
CHG E
      XI=XI+X
      IF(IDBG.EQ.1)WRITE(21,600) TIME,TEMP,DRVT,SDRVT,DELT,XI,X
C
      TB=TA
      XB=XA
      TEMP=TB
      TIME=XB
 600  FORMAT(7(3X,D12.6))
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
        SUBROUTINE TDER(RX,TEMP,TIME,DRVT,SDRVT,IFLAG,IDBG)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                            +
*  Purpose: computes the temperature slope at each time step                 +
*                                                                            +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=5000)
      PARAMETER(IDATW=30,IDATX=31,IDATY=32,IDATZ=33,IDATV=34)
      REAL*8  BREAK(2*NDATA), CSCOEF(4,2*NDATA)
      COMMON/CCC/ BREAK, CSCOEF, NINTV, JDBUG
      COMMON/BBB/ TIMEF,TI,TF,TOL
      COMMON/PPP/ EDERS,EDERF,TOP,BOT
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
C
      PARAMETER   (NDEG=3)
      REAL*8  B(NDEG+1),CSB(4,NDEG+1)
      COMMON/CURV/B,CSB,STIME
```

```
      EXTERNAL AT,DAT,DRCURV,CDER1,CDER2
C
C
C -------->     Evaluate first and second derivatives of modulus
C         using piecewise least square curve fitting:
C
      xx=RX
      if(xx.lt.STIME)xx=STIME
      x=dlog10(xx)

      CONST=dLOG10(dEXP(1D0))
      const=1./const
      dxy=const*xX
C
      if(x.ge.-6d0.and.x.le.-3d0) then
      do 32 k=1,ndeg+1
      b(k)=Csb(1,k)
  32  continue
      Y=CFIT(X)
      dy=cder1(x)
      ddy=cder2(x)
                      elseif(x.gt.-5d0.and.x.le.-3d0) then
      xk1=(-4-x)/2
      xk2=1d0-xk1
      do 133 k=1,ndeg+1
      b(k)=Csb(1,k)
 133  continue
      Y1=CFIT(X)
      dy1=cder1(x)
      ddy1=cder2(x)
      do 134 k=1,ndeg+1
      b(k)=  Csb(2,k)
134   continue
      Y2=CFIT(X)
      dy2=cder1(x)
      ddy2=cder2(x)
      y=y1*xk1+y2*xk2
      dy=dy1*xk1+dy2*xk2
      ddy=ddy1*xk1+ddy2*xk2
                       elseif(x.gt.-3d0.and.x.le.0d0) then

      do 232 k=1,ndeg+1
      b(k)=Csb(2,k)
  232 continue
      Y=CFIT(X)
      dy=cder1(x)
      ddy=cder2(x)
                        elseif(x.gt.0D0.and.x.le.2.0) then
      xk1=(2-x)/2
      xk2=1d0-xk1
      do 36 k=1,ndeg+1
```

```
      b(k)=Csb(2,k)
 36   continue
      Y1=CFIT(X)
      dy1=cder1(x)
      ddy1=cder2(x)
      do 37 k=1,ndeg+1
      b(k)=  Csb(3,k)
 37   continue
      Y2=CFIT(X)
      dy2=cder1(x)
      ddy2=cder2(x)
      y=y1*xk1+y2*xk2
      dy=dy1*xk1+dy2*xk2
      ddy=ddy1*xk1+ddy2*xk2
                        elseif(x.gt.2d0.and.x.le.6d0) then
      do 38 k=1,ndeg+1
      b(k)=Csb(3,k)
 38   continue
      Y=CFIT(X)
      dy=cder1(x)
      ddy=cder2(x)
                        elseif(x.gt.6d0.and.x.le.8d0) then
      xk1=(8-x)/2
      xk2=1d0-xk1
      do 39 k=1,ndeg+1
      b(k)=Csb(3,k)
 39   continue
      Y1=CFIT(X)
      dy1=cder1(x)
      ddy1=cder2(x)
      do 40 k=1,ndeg+1
      b(k)=  Csb(4,k)
 40   continue
      Y2=CFIT(X)
      dy2=cder1(x)
      ddy2=cder2(x)
      y=y1*xk1+y2*xk2
      dy=dy1*xk1+dy2*xk2
      ddy=ddy1*xk1+ddy2*xk2
                       elseif(x.gt.8d0.and.x.le.22d0) then

      do 41 k=1,ndeg+1
      b(k)=Csb(4,k)
 41   continue
      Y=CFIT(X)
      dy=cder1(x)
      ddy=cder2(x)
      endif
C
      tdy=dy/dxy
      top=ddy-tdy*xx*const*const
```

```
      tddy=top/dxy/dxy
      EDERF=tdy
      EDERS=tddy
      WRITE(IOUT,610) EDERF,EDERS
C
      AFUNCL= ATL(TEMP)
      AFUNC = AT (TEMP,AFUNCL)
C
      ADERFL= ATFL(TEMP,AFUNCL)
      ADERF = ATF (TEMP,AFUNCL,ADERFL)
C
      ADERSL= ATSL(TEMP,AFUNCL)
      ADERS = ATS (TEMP,AFUNCL,ADERFL,ADERSL)
C
C
      IF(IH.EQ.1)THEN
      TV   = TEMP-TI
      DINT = DTDHT(TEMP)*TV+HT(TEMP)
      DINTL=DTDHT(TEMP)*TV
      WRITE(94,6161)TEMP,DTDHT(TEMP),TV,DINTL,HT(TEMP),DINT
      ARGA = DT2DHT(TEMP)*TV+2*DTDHT(TEMP)
      ARGB = ADERS*DINT-ADERF*ARGA
      TOP = ADERF*DINT*EDERS
      BOT = EDERF*ARGB*AFUNC
      WRITE(92,6161)AFUNC,ADERF,ADERS,EDERF,EDERS,DINT,ARGB,
     +              TOP,BOT
      ALEFT=ADERS*DINT
      ARITE=ADERF*ARGA
                                                          ELSE
      TOP=EDERS*ADERF
      BOT=EDERF*ADERS*AFUNC
      ENDIF
CHG
      ABSBOT=DABS(BOT)
C
      IF(ABSBOT.LE.TOL)THEN
      IFLAG=1
      RETURN
      ENDIF
C
CHG      DRVT=-EDERS*TOP/EDERF/BOT
      DRVT=-TOP/BOT
C
      IF(ADERS.LT.0D0) THEN
      PRINT*
      PRINT*,' ADERS is positive ... ADERS is  ',ADERS
      ENDIF
C
      IF(DRVT.GT.0D0)THEN
      PRINT*
      PRINT*,' DRVT is positive ... DRVT is ',DRVT
```

```
      DRVT = SDRVT
      PRINT*,' Adjusted DRVT = ',DRVT
      RETURN
      ENDIF
C
      SDRVT=DRVT
      TD=DABS(DRVT)
C
 610  FORMAT(/,5X, 'First derivative ; EDERF =',D16.6,/,5X,
     &'Second derivative ; EDERS=',D16.6)
 6161 FORMAT(9(2X,D12.6))
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION CXI(X)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      COMMON/TDROP/ TEMPI,TEMPF,NSTEP
      COMMON/BBB/ TIMEF,TI,TF,TOL
C
      IF(XB.EQ.XA)THEN
      PRINT*,' Error ... Denominator is zero .'
      PRINT*,' XB = ',XB,'  XA = ',XA
      PRINT*,' TB = ',TB,'  TA = ',TA
      PRINT*,' X  = ',X
      READ(5,*) ICONT
      ENDIF
C
      T=(TB-TA)/(XB-XA)*(X-XA)+TA
CHG B
      AFUNCL= ATL(T)
      AFUNC = AT (T,AFUNCL)
C
      IF(AFUNC.EQ.0D0)THEN
      PRINT*,' Error ... Denominator is zero.'
      PRINT*,' TIME B = ',XB,'  TIME A = ',XA
      PRINT*,' TEMP B = ',TB,'  TEMP A = ',TA
      PRINT*,' TEMP X  = ',X
      PRINT*
      WRITE(6,600)T,AFUNCL,AFUNC,ARGA1,SIGX1,T
      READ(5,*) ICONT
C
      STOP
      ENDIF
C
      CXI=1/AFUNC
CHG E
```

```
 600  FORMAT(/,5X,'TEMPERATURE = ',D16.6,/,
     +5X,'ATL(T) = ',D16.6,5X,'AT(T) = ',D16.6,//,
     +5X,'AT(T) = ATL(T)*SAT(T,S);',/,
     +5X,'SAT = ',D16.6,5X,'S = ',D16.6,5X,'T = ',D16.6,/)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                 DOUBLE PRECISION FUNCTION ATL(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
CHG
      PARAMETER(M=4)
      REAL*8 ATLG(M),TRCP(M)
      DATA ATLG/6.,2.5,-8.,-10./
      DATA TRCP/3.3,2.69,2.1,1.6/
CHG      PARAMETER(M=4)
CHG      REAL*8 ATLG(M),TRCP(M)
CHG
CHGDATA      DATA ATLG/6.,2.5,-4.,-5.0/
CHGDATA      DATA TRCP/3.3,2.69,2.32,1.6/
C
      Y=1000./(T+273.3)
C
      IF(Y.GE.TRCP(1)) THEN
      ATL=(ATLG(1)-ATLG(2))/(TRCP(1)-TRCP(2))*(Y-TRCP(2))+ATLG(2)
      IF(ATL.GE.22D0)ATL=22.0
      ATL=10.**ATL
      RETURN
C
                                  ELSEIF(Y.LT.TRCP(M)) THEN
      ATL=ATLG(M)
      ATL=10.**ATL
      ENDIF
C
      DO 10 I=1, M-1
      IF(Y.LT.TRCP(I).AND.Y.GE.TRCP(I+1)) THEN
      ATL=(ATLG(I)-ATLG(I+1))/(TRCP(I)-TRCP(I+1))*(Y-TRCP(I+1))+
     +     ATLG(I+1)
      ATL=10.**ATL
      RETURN
      ENDIF
 10   CONTINUE
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE  PDFILE(T,ATEMP,TEMP,DERZER,DERFIR,DERSEC,IDAT,
     +            INTV,NSTEP)
```

```
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 T(0:NM),ATEMP(0:NM),TEMP(0:NM),DERZER(NM),DERFIR(NM),
     +       DERSEC(NM)
C
C  --------->      Print every  (NSTEP/INTV) Intervals:
C
      NPTS=INT(NSTEP/INTV)
C      WRITE(IDAT,601)
      WRITE(IDAT,400) T(0),ATEMP(0),TEMP(0)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) T(K),ATEMP(K),TEMP(K),DERZER(K),DERFIR(K),
     +DERSEC(K)
 350  CONTINUE
      WRITE(IDAT,400) T(NSTEP+1),ATEMP(NSTEP+1),TEMP(NSTEP+1)
 400  FORMAT(5X,F12.6,3X,F12.3,4(3X,D16.6))
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE  PSCRFILE(ATEMP,TEMP,DERZER,DERSEC,IDAT,INTV,NSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 ATEMP(NM),TEMP(NM),DERZER(NM),DERSEC(NM)
C
C  --------->      Print every  (NSTEP/INTV) Intervals:
C
      NPTS=INT(NSTEP/INTV)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) ATEMP(K),TEMP(K),DERZER(K),DERSEC(K)
 350  CONTINUE
 400  FORMAT(3X,F12.3,3(3X,D16.6))
      RETURN
      END
*

* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
           SUBROUTINE PFILE(T,ATEMP,TEMP,IDAT,INTV,NSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
```

```
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 T(0:NM),ATEMP(0:NM),TEMP(0:NM)
C
C  --------->      Print every  (NSTEP/INTV) Intervals:
C
      NPTS=INT(NSTEP/INTV)
C      WRITE(IDAT,601)
      WRITE(IDAT,400) T(0),ATEMP(0),TEMP(0)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) T(K),ATEMP(K),TEMP(K)
 350  CONTINUE
      WRITE(IDAT,400) T(NSTEP+1),ATEMP(NSTEP+1),TEMP(NSTEP+1)
 400  FORMAT(2X,F16.9,2X,5(F16.9,2X,F16.9,2X))
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          SUBROUTINE PPFILE(T,ATEMP,TEMP,DELT,IDAT,INTV,NSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 T(0:NM),ATEMP(0:NM),TEMP(0:NM),DELT(NM)
C
C  --------->      Print every  (NSTEP/INTV) Intervals:
C
      NPTS=INT(NSTEP/INTV)
C      WRITE(IDAT,601)
      K=0
      WRITE(IDAT,400) K,T(0),ATEMP(0),TEMP(0)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) K,T(K),ATEMP(K),TEMP(K),DELT(K)
 350  CONTINUE
      WRITE(IDAT,400) K,T(NSTEP+1),ATEMP(NSTEP+1),TEMP(NSTEP+1)
 400  FORMAT(2X,I5,2X,F16.9,2X,5(F16.9,2X,F16.9,2X))
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          SUBROUTINE PDBGFILE(T,ATEMP,TEMP,IDAT,INTV,NSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
```

```
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 T(0:NM),ATEMP(NM),TEMP(NM)
C
C  --------->      Print every  (NSTEP/INTV) Intervals:
C
      NPTS=INT(NSTEP/INTV)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) T(K),ATEMP(K),TEMP(K)
 350  CONTINUE
 400  FORMAT(2X,D16.6,2X,5(F16.9,2X,F16.9,2X))
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                    SUBROUTINE POSTPRO(IDATB,IDATR)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(M=3)
      REAL*8 ATLG(M),TRCP(M)
      COMMON/VATDATA/ATLG,TRCP
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,IADP,IANS,IH
CHGC
      IF(IH.EQ.1.OR.IH.EQ.2) THEN
      WRITE(IDATR,900)
      WRITE(IDATB,900)
      WRITE(IDATB,1850)
      DO 770 K=1, M
 770  WRITE(IDATB,1860)K, ATLG(K),K,TRCP(K)
C
                                                        ELSE
      WRITE(IDATR,905)
      WRITE(IDATB,905)
      ENDIF
C
      IF(IADP.EQ.1) THEN
      WRITE(IDATR,910)
      WRITE(IDATB,910)
                                                        ELSE
      WRITE(IDATR,915)
      WRITE(IDATB,915)
      ENDIF
C
      R    =  .9125
      TG   = 125.
      ULTSTRL=325
```

```
      ULTSTRT=11
      ULTSTNL=0.015
      ULTSTNT=0.008
      WRITE(IDATB,650)R,TG,ULTSTRL,ULTSTRT,ULTSTNL,ULTSTNT
      WRITE(IDATR,650)R,TG,ULTSTRL,ULTSTRT,ULTSTNL,ULTSTNT
      WRITE(IDATB,655)
      WRITE(IDATR,655)
C
      IF(IADP.EQ.1) THEN
      WRITE(IDATR,850)
                                                    ELSE
      WRITE(IDATR,860)
      ENDIF
C
C --------->      FORMAT STATEMENT;
C
 400  FORMAT(3X,D16.6,10X,F9.4,10X,6(D16.6,3X))
 450  FORMAT(3X,D16.6,10X,6(D16.6,3X))
1850  FORMAT(/,5X,'Data for vertical shift factors: ',/)
1860  FORMAT(  5X,'ATLG(',I3,' ) = ',F12.3,3X,'TRCP(',I3,' ) = ',
     +F12.3)
 900  FORMAT(5X,' * * * * * * * * * * * * * * * * * * * * * * * *',/,
     +       5X,' *                                               *',/,
     +       5X,' *  Viscoelastic stress analysis  for cross ply  *',/,
     +       5X,' *  composite under optimal temperature cool     *',/,
     +       5X,' *  down.                                        *',/,
     +       5X,' *  (with horizontal and vertical shift factors) *',/,
     +       5X,' *                                               *',/,
     +       5X,' * * * * * * * * * * * * * * * * * * * * * * * *',//)
905   FORMAT(5X,' * * * * * * * * * * * * * * * * * * * * * * * *',/,
     +       5X,' *                                               *',/,
     +       5X,' *  Viscoelastic stress analysis  for cross ply  *',/,
     +       5X,' *  composite under optimal temperature cool     *',/,
     +       5X,' *  down.                                        *',/,
     +       5X,' *  (with horizontal and vertical shift factors) *',/,
     +       5X,' *                                               *',/,
     +       5X,' * * * * * * * * * * * * * * * * * * * * * * * *',//)
 910  FORMAT(///,5X,' General Purpose Adaptive Rule ; DQDAG ')
 915  FORMAT(///,5X,' General Purpose Adaptive Rule End Point',
     +' Singularity ;DQDAGS ')
650   FORMAT(///,5X,'***  ENGINEERING DATA  ***',/,
     +5X,'R ( Material Constant ) = ',F16.6,/,
     +5X,'Glass transition temperature (deg)    = ',F16.6,/,
     +5X,'Ultimate stress, longitudinal (ksi)   = ',F16.6,/,
     +5X,'                 transverse   (ksi)   = ',F16.6,/,
     +5X,'Ultimate strain, longitudinal (in/in) = ',F16.6,/,
     +5X,'                 transverse   (in/in) = ',F16.6),/)
 655  FORMAT(//,5X,'* * *  ',
     +'For temperature range between 23 deg AND 125 deg ;',/,5X,
     +'Thermal expansion coefficient, longitudinal (/ deg C)= .5 E-6',/,5X,
     +'                               transverse   (/ deg C)= 30 E-6 ',/,
```

```
     +//,5X,'* * *',
     +'For temperature range between 125 deg AND 300 deg ;',/,5X,
     +'Thermal expansion coefficient, longitudinal (/ deg C)= 1.0 E-6',/,5X,
     +'                               transverse   (/ deg C)= 75. E-6 ',/ )
 850  FORMAT(/,5X,'General purpose adaptive rule ( DQDAG)',
     +' is used for integration of reduced times',//)
 860  FORMAT(/,5X,'General purpose adaptive rule end poing',
     +' singularity ( DQDAGS) is used for integration',
     +' of reduced times',/)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE VSTRESS(NSTEP,T,TEMP,STRESS,DELT,XI,STRTF)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: computes stress history along the optimal temperature       +
*           path                                                        +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000,IDATS=22)
      INTEGER INUM(NM)
      REAL*8 XI(NM),T(0:NM),TEMP(0:NM), STRESS(0:NM), DELT(NM)
      REAL*8 TIMEM(0:NM),TEMPM(0:NM),BSTRESS(0:NM),BSRSS(100)
      REAL*8 STRESSM(0:NM),FSTRESS(0:NM),FDELT(NM)
C
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      COMMON/BBB/ TIMEF,TI,TF,TOL
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
C
      CALL XQSTR(NSTEP,T,TEMP,STRESS,DELT,XI,STRTF)
      WRITE(6,*) '     FINAL STRESS  = ',STRTF
C
      PRINT*
      PRINT*,' Input 0,          to return'
      PRINT*,'       any number, to refine steps'
      READ(5,*) ICHG
      IF(ICHG.EQ.0)RETURN
C
      ISTR=1
C
      CALL  REPXY (DELT,  FDELT,  NSTEP )
      CALL  REPXY (STRESS,FSTRESS,NSTEP )
      CALL  REPXY (STRESS,BSTRESS,NSTEP )
C
C      CALL  BRUBUD (NSTEP,IPRT,DELTP,DELT,T,TEMP)
C
 100  PRINT*,' Input 1, to refine from original O.T.P '
      PRINT*,'       0,                previous O.T.P '
```

```
      READ(5,*) IPATH
C
      PRINT*,'       1,  to change time intervals for whole o.t.p '
      PRINT*,'       2,                            partly'
      READ(5,*) ICHG
C
      IF(IPATH.EQ.1)THEN
      CALL  BRUBUD (NSTEP,IPRT,DELTP,DELT,T,TEMP)
                                                        ELSE
      CALL  BRUBUD (KSTEP,IPRT,DELTP,DELT,TIMEM,TEMPM)
      ENDIF
C
      IF(IPATH.EQ.1)THEN
      NEWSTP=NSTEP
                                                        ELSE
      NEWSTP=KSTEP
      ENDIF
C
      WRITE(6,666)
      INTV=15
C
      IF(IPATH.EQ.1)THEN
      CALL PPFILE(T,TEMP,FSTRESS,FDELT,6,INTV,NEWSTP)
                                                        ELSE
      CALL PPFILE(TIMEM,TEMPM,STRESSM,DELT,6,INTV,NEWSTP)
      ENDIF
C
      PRINT*
      PRINT*,' Input any number to continue '
      READ(5,*) ICONT
C
      PRINT*
      PRINT*,' Input 1 to see Data for specific steps '
      READ(5,*) ISI
C
 777  IF(ISI.EQ.1)THEN
      PRINT*,' Input number of steps to see.'
      READ(5,*) NSTP
      PRINT*,' Input number of data to see. '
      READ(5,*) INTV
C
      WRITE(6,666)
C
      IF(IPATH.EQ.1)THEN
      CALL PPFILE(T,TEMP,FSTRESS,FDELT,6,INTV,NSTP)
                                                        ELSE
      CALL PPFILE(TIMEM,TEMPM,STRESSM,DELT,6,INTV,NSTP)
      ENDIF
C
      PRINT*
      PRINT*,' Input 1  continue to see Data any specific step  '
```

```
      READ(5,*) ISI
      IF(ISI.EQ.1) GOTO 777
      ENDIF
C
      IF(ICHG.EQ.1)THEN
      ISTEP=NSTEP
                                                        ELSE
      PRINT*
      PRINT*,' ..... Input proposed  Maximum Time Interval;  '
      READ(5,*) DELTP
      PRINT*,' ..... Input specific number of step to be refined ;'
      READ(5,*) ISTEP
      ENDIF
C
      IF(IPATH.EQ.0)THEN
      CALL REPXY (TIMEM,T,NEWSTP)
      CALL REPXY (TEMP,TEMPM,NEWSTP)
      ENDIF
C
      CALL  XQMOD  (NEWSTP,KSTEP,ISTEP,INUM,DELTP,T,TEMP,TIMEM,
     +              TEMPM,DELT )
C
      PRINT*
      PRINT*,' Previous total number of steps is  ',NEWSTP
      PRINT*,' New number of steps is             ',KSTEP
      PRINT*
C
C
      PRINT*
      PRINT*,' Input 1 to print data for modified time intervals.'
      READ(5,*) IPRT
C
      IF(IPRT.EQ.1)THEN
      INTV=ISTEP
      WRITE(IDATS,650)
      CALL PDSFILE(T,TIMEM,TEMP,TEMPM,INUM,IDATS,INTV,ISTEP,KSTEP)
C
      INTV=KSTEP
      WRITE(IDATS,660)
      CALL PSFILE(TIMEM,TEMPM,DELT,IDATS,INTV,KSTEP)
      ENDIF
CC
      PRINT*,' To try another, hit 1'
      READ(5,*) ICONT
      IF(ICONT.EQ.1)GOTO 100
C
      CALL XQSTR(KSTEP,TIMEM,TEMPM,STRESSM,DELT,XI,STRTF)
      PRINT*
      IF(ICHG.EQ.1) THEN
      WRITE(6,*) '     FINAL STRESS  = ',STRTF
                                                        ELSE
```

```
      IK= INUM(ISTEP)
      WRITE(6,669) ISTEP,STRESSM(IK)
      ENDIF
C
      PRINT*
      PRINT*,' Input 1 to see Stresses at specific steps '
      READ(5,*) ISI
C
 888  IF(ISI.EQ.1)THEN
      PRINT*,' NUMBER OF STEPS IS ', KSTEP
      PRINT*
      PRINT*,' Input Number of Steps to see.'
      READ(5,*) NSTP
      PRINT*,' Input Number of Data to see. '
      READ(5,*) INTV
C
      WRITE(6,666)
      CALL PPFILE(TIMEM,TEMPM,STRESSM,DELT,6,INTV,NSTP)
      PRINT*
      PRINT*,' To print SCREEN, input 1'
      READ(5,*) IP
C
      IF(IP.EQ.1)THEN
      WRITE(IDATS,666)
      CALL PPFILE(TIMEM,TEMPM,STRESSM,DELT,IDATS,INTV,NSTP)
      ENDIF
      PRINT*
      PRINT*,' Input 1 to see Stresses at specific steps '
      READ(5,*) ISI
      IF(ISI.EQ.1)GOTO 888
      ENDIF

C
      CALL  CHGSTR(STRESSM,STRESS,INUM,NSTEP,KSTEP)
C
      CALL XQSRSS (BSTRESS,STRESS,SRSS,NSTEP )
      BSRSS(ISTR)=SRSS
C
      WRITE(6,670) ISTR
      WRITE(IDATS,670) ISTR
      DO 701 K=1, ISTR
      WRITE(6,675 ) K, BSRSS(K)
      WRITE(IDATS,675) K,BSRSS(K)
 701  CONTINUE
      PRINT*
      PRINT*,' Input any number to continue '
      READ(5,*) ICONT
C
      INTV=10
      WRITE(6,601)  ISTR
      WRITE(IDATS,601) ISTR
C
```

```
      IF(ICHG.EQ.1)THEN
      NSTEPB=NSTEP
                                                          ELSE
      NSTEPB=ISTEP-1
      ENDIF
C
      IF(INTV.GT.NSTEPB)INTV=NSTEPB
      CALL PFILE(T,FSTRESS,STRESS,6,INTV,NSTEPB)
      CALL PFILE(T,FSTRESS,STRESS,IDATS,INTV,NSTEPB)
C
 778  PRINT*
      PRINT*,' Input 1 to see specific steps '
      READ(5,*) ISI
C
      IF(ISI.EQ.1)THEN
      PRINT*,' Input Number of Steps to see.'
      READ(5,*) NSTP
      PRINT*,' Input Number of Data to see. '
      READ(5,*) INTV
C
      WRITE(6,666)
      CALL PPFILE(T,TEMP,STRESS,DELT,6,INTV,NSTP)
      PRINT*
      PRINT*,' Input 1  continue to see any specific step  '
      READ(5,*) ICONT
      IF(ICONT.EQ.1)GOTO 778
      ENDIF
C
      PRINT*
      PRINT*,' Input 0,  to return'
      PRINT*,'       1,  to change time intervals '
      READ(5,*) ICHG
      IF(ICHG.EQ.0)RETURN
      CALL REPXY (STRESS,BSTRESS,NSTEP)
      ISTR=ISTR+1
      GOTO 100
C
C --------->      FORMAT
C
 601  FORMAT(/,10X,'NUMBER  OF ITERATION = ',I5,//,
     +10X,'TIME', 13X,'STRESS',10X,'STRESS',/)
 620  FORMAT(/,10X,'TIME', 10X,'TEMPARATURE',10X,'STRESS',/)
 630  FORMAT(/,5X,'NOS',10X,'TIME', 10X,'TEMPARATURE'/)
 650  FORMAT(/,10X,'TIME',13X,'TIME (MOD)',5X,'TEMPERATURE',
     +6X,'TEMPERATURE (MOD)',/)
 660  FORMAT(/,10X,'TIME', 10X,'TEMPARATURE',10X,'DELT',/,
     +25X,'MODIFIED')
 666  FORMAT(/,5X,'NOS',10X,'TIME', 10X,'TEMPARATURE',
     +10X,'STRESS',10X,'DELT',/)
 669  FORMAT(/,5X,'* Stress at ',I5,'th step  is ',F12.3,//)
 670  FORMAT(/,5X,' Number of iteration = ',I5,//,
```

```
     +5X,' ITERATION',10X,'SQ. ROOT OF SQ. SUM ',/)
 675  FORMAT(3X,I5,15X,D12.6)
C
C      RETURN
       END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
         SUBROUTINE XQSTR(NSTEP,T,TEMP,STRESS,DELT,XI,STRTF)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
       IMPLICIT REAL*8(A-H,O-Z)
       PARAMETER(NM=5000)
       REAL*8 XI(NM),T(0:NM),TEMP(0:NM), STRESS(0:NM), DELT(NM)
C
       COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
       COMMON/BBB/ TIMEF,TI,TF,TOL
       COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
C
       SIGMA=0D0
       STRESS(0)=SIGMA
C
       TEMPA=TI
       TEMPB= TEMP(1)
       EPSTF= 1D-6
C
       SIGMA=R*ET(EPSTF)*(75D-6-1D-6)*(TEMPA-TEMPB)*1000./6.895
       STRESS(1)=SIGMA
       SA=SIGMA
C
       XI(1)=0D0
C
       DO 40 I=1, NSTEP-1
       XA=T(I)
       XB=T(I+1)
       TA=TEMP(I)
       TB=TEMP(I+1)
C
       CALL CXISTR(XXX)
       XI(I+1)=XI(I)+XXX
       N=I
       ENTEGRAL=0.0
C
       DO 30 J=1, N
       X = XI(N)-XI(J)
       XK=TEMP(J+1)
       XJ=TEMP(J)
       TK=T(J+1)
       TJ=T(J)
       DELTJ=TK-TJ
       DELT(J)=DELTJ
```

```
C
      IF(X.LT.1D-6) X=1D-6
      DTDT=(XK-XJ)/(TK-TJ)
      SLOPE=-DTDT
C
      IF(IH.EQ.1) THEN
        VDEV=(1./VAT(XK)-1./VAT(XJ))/(TK-TJ)
        SLOPE=-(VDEV*(XK-TEMPA)+DTDT/VAT(XK))
      ENDIF
C
  30  ENTEGRAL = ENTEGRAL +ET(X)*DELTJ*SLOPE*(ALPAT(XJ)-ALPAL(XJ))
C
      IF(IH.EQ.2) THEN
      ENTEGRAL = ENTEGRAL/VAT(TB)
      ENDIF
C
      SIGMA=ENTEGRAL
      X=XI(N)
      IF(X.LE.1D-6) X=1D-6
      SIGMA=ET(X)*(75D-6-1D-6)*(TEMPA-TEMPB)+SIGMA
      SIGMA=R*SIGMA*1000./6.895
C
      STRESS(I+1)=SIGMA
 40   CONTINUE
C
      TEMPA= TEMP(NSTEP)
      TEMPB=30.
      EPSTF=1D-6
      STRTF=STRESS(NSTEP)
      STR=R*ET(EPSTF)*(30D-6-.5D-6)*(TEMPA-TEMPB)*1000./6.895
      ETF=ET(EPSTF)
      STRTF=STRTF+STR
      STRESS(NSTEP+1)=STRTF
      T(NSTEP+1)=T(NSTEP)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                  DOUBLE PRECISION FUNCTION ALPAT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: interpolate  the temperature dependent coefficient of       +
*           temperature expansion at 90 degree direction                +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      TG=125D0
      ALPATN=75D-6
      ALPATP=30D-6
```

```
      ALPATM=(ALPATP+ALPATN)/2.
      ALPATO=ALPATM-ALPATN
      CONSTA=-3.*ALPATO/2.
      CONSTB=    ALPATO/2.
C
      IF(T.LE.124D0) THEN
      ALPAT=ALPATP
                                  ELSEIF(T.GE.126D0) THEN
      ALPAT=ALPATN
                                                     ELSE
      ALPAT=ALPATM+CONSTA*(T-TG)+CONSTB*(T-TG)*(T-TG)*(T-TG)
      ENDIF
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION ALPAL(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: interpolate  the temperature dependent coefficient of       +
*           temperature expansion at 90 degree direction                +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      TG=125D0
      ALPALN=1D-6
      ALPALP=.5D-6
      ALPALM=(ALPALP+ALPALN)/2.
      ALPALO=ALPALM-ALPALN
      CONSTA=-3.*ALPALO/2.
      CONSTB=    ALPALO/2.
C
      IF(T.LE.124D0) THEN
      ALPAL=ALPALP
                                  ELSEIF(T.GE.126D0) THEN
      ALPAL=ALPALN
                                                     ELSE
      ALPAL=ALPALM+CONSTA*(T-TG)+CONSTB*(T-TG)*(T-TG)*(T-TG)
      ENDIF
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                      SUBROUTINE CXISTR(X)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
```

```
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      EXTERNAL CAT,AT,DQDAG,DQDAGS,VAT
      EXTERNAL DCSVAL,SAT
      IRULE=2
      ERRABS=0D0
      ERRREL=1D-3
C
      IF(XA.EQ.XB) THEN
      WRITE(6,*) ' .... ERROR in CXISTR Routine ; XA = XB '
      WRITE(6,*) '   XA = ',XA, ' XB= ',XB
      STOP
                                    ELSEIF(TA.EQ.TB) THEN
      WRITE(6,*) ' .... ERROR in CXISTR Routine ; TA = TB '
      WRITE(6,*) '   TA = ',TA, ' TB= ',TB
      STOP
      ENDIF
C
C -----> CALCULATE THE REDUCED TIME
C
      IF(IADP.EQ.1) THEN
      CALL DQDAG(CAT,XA,XB,ERRABS,ERRREL,IRULE,X,ERREST)
                                                      ELSE
      CALL DQDAGS(CAT,XA,XB,ERRABS,ERRREL,X,ERREST)
      ENDIF
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                  DOUBLE PRECISION FUNCTION CAT(X)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      COMMON/DATA/TA,TB,XA,XB,DTEGAL,DELTMAX,DELTMIN,CHKMIN,
     +      IADP,IANS,IH
      COMMON/BBB/ TIMEF,TI,TF,TOL
      EXTERNAL AT,SAT,DCSVAL
C
      T=(TB-TA)/(XB-XA)*(X-XA)+TA
C
      AFUNCL= ATL(T)
      AFUNC = AT (T,AFUNCL)
      RCAT=AFUNC
      CAT=1./RCAT
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
               DOUBLE PRECISION FUNCTION ET(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
```

```
      IMPLICIT REAL*8(A-H,O-Z)
C      PARAMETER(MM=29)
      PARAMETER(MM=25)
C
      REAL*8 E(MM),TLG(MM)
C
C      DATA E/9.17,9.13,9.10,9.06,9.02,8.98,8.77,8.44,7.89,7.32,
C     +6.76,6.37,5.98,5.74,5.70,5.65,5.60,5.40,5.10,4.60,4.20,3.70,
C     +3.20,2.70,2.30,1.80,1.35,.80,0.0/
C      DATA TLG/-6.,-5.,-4.,-3.,-2.,-1.,0.,1.,2.,3.,4.,5.,6.,7.,
C     +8.,9.,10.,11.,12.,13.,14.,15.,16.,17.,18.,19.,20.,21.,22./
C
      DATA E/9.17,9.13,9.10,9.06,9.02,8.98,8.77,8.44,7.89,7.32,
     +6.76,6.37,5.98,5.74,5.40,5.22,5.00,4.70,4.40,3.95,3.40,2.25,
     +2.00,1.10,0.0/
      DATA TLG/-6.,-5.,-4.,-3.,-2.,-1.,0.,1.,2.,3.,4.,5.,6.,7.,
     +8.,9.,10.,11.,12.,13.,14.,15.,16.,17.,18./
CC
      TT=DLOG10(T)
C
      IF(TT.LE.TLG(1)) THEN
      ET=E(1)
      RETURN
                                ELSEIF(TT.GE.TLG(MM)) THEN
      ET=E(MM)
      RETURN
      ENDIF
C
      DO 10 I = 2, MM
C
      IF(TT.GT.TLG(I-1).AND.TT.LE.TLG(I)) THEN
      ET=(E(I)-E(I-1))/(TLG(I)-TLG(I-1))*(TT-TLG(I-1))+E(I-1)
      RETURN
      ENDIF
C
  10  CONTINUE
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            SUBROUTINE ALPACS(CSCOEFN,BREAKN,IBREAKN)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: generate cubic spline interpolation function of temperature +
*           dependent parameter alpa                                    +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=6)
      REAL*8 BREAKN(2*NDATA),CSCOEFN(4,2*NDATA),FDATA(NDATA),
```

```
     +       XDATA(NDATA)
      EXTERNAL DCSCON
      DATA XDATA/24.411,44.264,74.412,91.323,103.088,120.00/
      DATA FDATA/.118,.118,.118,.146,.171,.239/
C
      CALL DCSCON(NDATA,XDATA,FDATA,IBREAKN,BREAKN,CSCOEFN)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            SUBROUTINE  OCTACS(CSCOEFM,BREAKM,IBREAKM)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: generate cubic spline interpolation function of temperature +
*           dependent parameter octa                                    +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(MDATA=3)
      REAL*8 BREAKM(2*MDATA),CSCOEFM(4,2*MDATA),FDATA(MDATA),
     +       XDATA(MDATA)
      EXTERNAL DCSCON
      DATA XDATA/20.00,86.176,120.00/
      DATA FDATA/31.42,30.00,17.857/
C
      CALL DCSCON(MDATA,XDATA,FDATA,IBREAKM,BREAKM,CSCOEFM)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                  DOUBLE PRECISION FUNCTION VAT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                       +
*  Purpose: interpolate the vertical shift factor                       +
*                                                                       +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(M=3)
      REAL*8 ATLG(M),TRCP(M)
      COMMON/VATDATA/ATLG,TRCP
      COMMON/DEBUG1/ TEMP1,VATL1,VAT1
      Y = T
C
      IF(Y.GE.TRCP(1)) THEN
      VAT=(ATLG(1)-ATLG(2))/(TRCP(1)-TRCP(2))*(Y-TRCP(2))+ATLG(2)
      TEMP1=Y
      VATL1=VAT
      VAT=10.**VAT
      VAT1=VAT
```

```
      RETURN
                                    ELSEIF(Y.LT.TRCP(M)) THEN

      VAT=ATLG(M)
      TEMP1=Y
      VATL1=VAT
      VAT=10.**VAT
      VAT1=VAT
      RETURN
      ENDIF
C
      DO 10 I=1, M-1
      IF(Y.LT.TRCP(I).AND.Y.GE.TRCP(I+1)) THEN
      VAT=(ATLG(I)-ATLG(I+1))/(TRCP(I)-TRCP(I+1))*(Y-TRCP(I+1))+
     +ATLG(I+1)
C
      TEMP1=Y
      VATL1=VAT
      VAT=10.**VAT
      VAT1=VAT
      RETURN
      ENDIF
  10  CONTINUE
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION HT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL VAT
      COMMON/DEBUG2/TEMP2,VAT2,HT2
      HT=1./VAT(T)
C
      TEMP2=T
      VAT2=VAT(T)
      HT2=HT
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION DTDHT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL VAT,DTDVAT
      COMMON/DEBUG3/TEMP3,DTDHT3,VAT3,DTDVAT3
C
      DTDHT=-1./VAT(T)/VAT(T)*DTDVAT(T)
C
      TEMP3=T
```

```
      DTDHT3=DTDHT
      VAT3=VAT(T)
      DTDVAT3=DTDVAT(T)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION DTDVAT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      EXTERNAL DTDLVT,VAT
      COMMON/DEBUG4/TEMP4,DTDVAT4,DTDLVT4,VAT4
C
      DTDVAT=DTDLVT(T)*VAT(T)*DLOG(10D0)
C
      TEMP4=T
      DTDVAT4=DTDVAT
      DTDLVT4=DTDLVT(T)
      VAT4=VAT(T)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                DOUBLE PRECISION FUNCTION DTDLVT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(M=3)
      REAL*8 ATLG(M),TRCP(M)
      COMMON/VATDATA/ATLG,TRCP
      COMMON/DEBUG5/TEMP5,DTDLVT5
C
      Y = T
C
      IF(Y.GE.TRCP(1)) THEN
      DTDLVT=(ATLG(1)-ATLG(2))/(TRCP(1)-TRCP(2))
      TEMP5=Y
      DTDLVT5=DTDLVT
      RETURN
C
                                  ELSEIF(Y.LT.TRCP(M)) THEN
      DTDLVT=(ATLG(M-1)-ATLG(M))/(TRCP(M-1)-TRCP(M))
      TEMP5=Y
      DTDLVT5=DTDLVT
      RETURN
      ENDIF
C
      DO 10 I=1, M-1
      IF(Y.LT.TRCP(I).AND.Y.GE.TRCP(I+1)) THEN
      DTDLVT=(ATLG(I)-ATLG(I+1))/(TRCP(I)-TRCP(I+1))
```

```
      TEMP5=Y
      DTDLVT5=DTDLVT
      RETURN
      ENDIF
  10  CONTINUE
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION DT2DHT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/DEBUG6/TEMP6,VAT6,DT2DVT6,DTDVAT6,DT2DHT6,DLEFT,DRITE
C
      DLEFT=2./VAT(T)/VAT(T)/VAT(T)*DTDVAT(T)
      DRITE=-1./VAT(T)/VAT(T)*DT2DVT(T)
      DT2DHT=DLEFT+DRITE
C
      TEMP6=T
      VAT6=VAT(T)
      DT2DVT6=DT2DVT(T)
      DTDVAT6=DTDVAT(T)
      DT2DHT6=DT2DHT
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                  DOUBLE PRECISION FUNCTION DT2DVT(T)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      COMMON/DEBUG7/TEMP7,DT2DVT7,DTDLVT7,DTDVAT7
C
      DT2DVT=DTDLVT(T)*DTDVAT(T)*DLOG(10D0)
C
      TEMP7=T
      DT2DVT7=DT2DVT
      DTDLVT7=DTDLVT(T)
      DTDVAT7=DTDVAT(T)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
       DOUBLE PRECISION FUNCTION SATS(T,SIGMA,SIGMAF,SIGMAS)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*                                                                      +
*  Purpose: interpolate the stress-induced nonlinear parameter ''as''  +
*                                                                      +
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
```

```
*
      IMPLICIT REAL*8(A-H,O-Z)
CHG
      COMMON/OCT/OCTSHR,OCTA
C
      IF(OCTSHR.GT.OCTA) THEN
      SATS = GS(T,SIGMA,SIGMAF,SIGMAS) * SAT(T,SIGMA)+
     &       GF(T,SIGMA,SIGMAF) * SATF(T,SIGMA,SIGMAF)
      RETURN
                                                      ELSE
      SATS = 0D0
      RETURN
      ENDIF
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          DOUBLE PRECISION FUNCTION SATF(T,SIGMA,SIGMAF)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
CHG
      COMMON/OCT/OCTSHR,OCTA
CHGC
      IF(OCTSHR.GT.OCTA) THEN
      SATF= GF(T,SIGMA,SIGMAF) * SAT(T,SIGMA)
C
      RETURN
                                                      ELSE
      SATF=0D0
      RETURN
      ENDIF
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
             DOUBLE PRECISION FUNCTION SAT(T,SIGMA)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=6,MDATA=3)
      REAL*8 BREAKN(2*NDATA),CSCOEFN(4,2*NDATA)
      REAL*8 BREAKM(2*MDATA),CSCOEFM(4,2*MDATA)
      COMMON/CSN/CSCOEFN,BREAKN,IBREAKN
      COMMON/CSM/CSCOEFM,BREAKM,IBREAKM
CHG
      COMMON/OCT/OCTSHR,OCTA
CHG
      EXTERNAL DCSVAL
      NINTVN=IBREAKN-1
      NINTVM=IBREAKM-1
      SATOL=-5
C
```

```
C ---------->     CONVERT STRESS UNIT TO MPA FROM KSI
C
      OCTSHR=DABS(SIGMA)*6.895
C      OCTSHR=2.00**.5*OCTSHR/3.
      OCTSHR=6.00**.5*OCTSHR/3.
C
      IF(T.GT.120.0) THEN
      ALPA=(.239-.119)*(T-74.412)/(120.-74.412)+.119
      IF(ALPA.GE.0.65) ALPA=0.65
C
      OCTA=(17.857-30.00)*(T-86.176)/(120.-86.176)+30.000
      IF(OCTA.LE.17.857) OCTA=17.857

                                    ELSEIF(T.LT.20.0) THEN
      ALPA=.118
      OCTA=31.42
                                                       ELSE
      ALPA=DCSVAL(T,NINTVN,BREAKN,CSCOEFN)
      OCTA=DCSVAL(T,NINTVM,BREAKM,CSCOEFM)
      ENDIF
C
      IF(OCTSHR.GT.OCTA) THEN
      CSAT=-ALPA*(OCTSHR-OCTA)
      PARTA6 =OCTSHR-OCTA
      IF(CSAT.GE.85.00) THEN
      WRITE(11,670) CSAT
      STOP
      ENDIF
      SAT=DEXP(CSAT)
C
      RETURN
      ELSE
      SAT=1D0
      RETURN
      ENDIF
C
C ---------->      FORMAT:
C
 670  FORMAT(5X,'* WARNING---CSAT IS GREATER THAN 87.49 ')
 650  FORMAT(5X,'CSAT=',D16.6,5X,'SAT=',D16.6)
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            DOUBLE PRECISION FUNCTION GF(T,SIGMA,SIGMAF)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=6,MDATA=3)
      REAL*8 BREAKN(2*NDATA),CSCOEFN(4,2*NDATA)
      REAL*8 BREAKM(2*MDATA),CSCOEFM(4,2*MDATA)
      COMMON/CSN/CSCOEFN,BREAKN,IBREAKN
```

```
      COMMON/CSM/CSCOEFM,BREAKM,IBREAKM
CHG
      EXTERNAL DCSVAL, DCSDER
      NINTVN=IBREAKN-1
      NINTVM=IBREAKM-1
C
C --------->     CONVERT STRESS UNIT TO MPA FROM KSI
C
C      OCTSHR=2.00**.5*OCTSHR/3.
      OCTSHR =DABS(SIGMA)*6.895
      OCTSHR =6.00**.5*OCTSHR/3.
C
      OCTSHRF =DABS(SIGMAF)*6.895
      OCTSHRF =6.00**.5*OCTSHRF/3.
C
      IF(T.GT.120.0) THEN
      ALPA=(.239-.119)*(T-74.412)/(120.-74.412)+.119
      ALPAF = (.239-.119)/(120.-74.412)
      IF(ALPA.GE.0.65) ALPA=0.65
C
      OCTA=(17.857-30.00)*(T-86.176)/(120.-86.176)+30.000
      OCTAF = (17.857-30.00)/(120.0-86.176)
C
      IF(OCTA.LE.17.857) THEN
      OCTA=17.857
      OCTAF=0D0
      ENDIF
                                    ELSEIF(T.LT.20.0) THEN
      ALPA=.118
      OCTA=31.42
      ALPAF=0D0
      OCTAF=0D0
                                                      ELSE
      ALPA=DCSVAL(T,NINTVN,BREAKN,CSCOEFN)
      ALPAF=DCSDER(1,T,NINTVN,BREAKN,CSCOEFN)
      OCTA=DCSVAL(T,NINTVM,BREAKM,CSCOEFM)
      OCTAF=DCSDER(1,T,NINTVM,BREAKM,CSCOEFM)
      ENDIF
C
      IF(OCTSHR.GT.OCTA) THEN
      GF=-(ALPAF*OCTSHR+ALPA*OCTSHRF)+(ALPAF*OCTA+ALPA*OCTAF)
      RETURN
      ELSE
      GF=0D0
      RETURN
      ENDIF
C
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          DOUBLE PRECISION FUNCTION GS(T,SIGMA,SIGMAF,SIGMAS)
```

```
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=6,MDATA=3)
      REAL*8 BREAKN(2*NDATA),CSCOEFN(4,2*NDATA)
      REAL*8 BREAKM(2*MDATA),CSCOEFM(4,2*MDATA)
      COMMON/CSN/CSCOEFN,BREAKN,IBREAKN
      COMMON/CSM/CSCOEFM,BREAKM,IBREAKM
CHG
      EXTERNAL DCSVAL, DCSDER
      NINTVN=IBREAKN-1
      NINTVM=IBREAKM-1
C
C----> CONVERT STRESS UNIT TO MPA FROM KSI HERE !
C      OCTSHR=2.00**.5*OCTSHR/3.
      OCTSHR =DABS(SIGMA)*6.895
      OCTSHR =6.00**.5*OCTSHR/3.
C
      OCTSHRF=DABS(SIGMAF)*6.895
      OCTSHRF=6.00**.5*OCTSHRF/3.
C
      OCTSHRS=DABS(SIGMAS)*6.895
      OCTSHRS=6.00**.5*OCTSHRS/3.
C
      IF(T.GT.120.0) THEN
      ALPA=(.239-.119)*(T-74.412)/(120.-74.412)+.119
      ALPAF = (.239-.119)/(120.-74.412)
      ALPAS =0D0
      IF(ALPA.GE.0.65)ALPA=0.65
C
      OCTA=(17.857-30.00)*(T-86.176)/(120.-86.176)+30.000
      OCTAF = (17.857-30.00)/(120.0-86.176)
      OCTAS = 0D0
C
      IF(OCTA.LE.17.857) THEN
      OCTA=17.857
      OCTAF=0D0
      OCTAS=0D0
      ENDIF

                                    ELSEIF(T.LT.20.0) THEN
      ALPA=0.118
      OCTA=31.42
      ALPAF=0D0
      OCTAF=0D0
C
      ALPAS=0D0
      OCTAS=0D0
                                                      ELSE
      ALPA=DCSVAL(T,NINTVN,BREAKN,CSCOEFN)
      OCTA=DCSVAL(T,NINTVM,BREAKM,CSCOEFM)
```

```
C
      ALPAF=DCSDER(1,T,NINTVN,BREAKN,CSCOEFN)
      OCTAF=DCSDER(1,T,NINTVM,BREAKM,CSCOEFM)
C
      ALPAS=DCSDER(2,T,NINTVN,BREAKN,CSCOEFN)
      OCTAS=DCSDER(2,T,NINTVM,BREAKM,CSCOEFM)
      ENDIF
C
      IF(OCTSHR.GT.OCTA) THEN
      GS=-ALPAS*(OCTSHR-OCTA)-2.*ALPAF*(OCTSHRF-OCTAF)-
     +    ALPA*(OCTSHRS-OCTAS)
      RETURN
                                                      ELSE
      GS=0D0
      RETURN
      ENDIF
C
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION AT(TEMP,AFUNCL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=5000)
      REAL*8     BREAK(2*NDATA), CSCOEF(4,2*NDATA)
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      COMMON/CCC/ BREAK, CSCOEF, NINTV, JDBUG
      EXTERNAL   DCSVAL
C
      IF(ITERATION.EQ.1)THEN
      AT = AFUNCL
      RETURN
                                                      ELSE
      SIGX = DCSVAL(TEMP, NINTV, BREAK, CSCOEF)
      ARGA = SAT(TEMP,SIGX)
      AT   = AFUNCL*ARGA
      ENDIF
C
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
              DOUBLE PRECISION FUNCTION ATFL(TEMP,AFUNCL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      ATFL=DAT(TEMP)*FDKT(TEMP)*AFUNCL*DLOG(10D0)
C
      RETURN
```

```
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          DOUBLE PRECISION FUNCTION ATF(TEMP,AFUNCL,ADERFL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      PARAMETER(NDATA=5000)
      REAL*8     BREAK(2*NDATA), CSCOEF(4,2*NDATA)
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      COMMON/CCC/ BREAK, CSCOEF, NINTV, JDBUG
      EXTERNAL   DCSDER, DCSVAL
C
      IF(ITERATION.EQ.1)THEN
      ATF = ADERFL
      RETURN
                                                        ELSE
      SIGX  = DCSVAL(TEMP, NINTV, BREAK, CSCOEF)
      SIGXF = DCSDER(1, TEMP,NINTV,BREAK,CSCOEF)
      ARGA = SAT(TEMP,SIGX)
      ARGB = SATF(TEMP,SIGX,SIGXF)
C
      ATF  = ADERFL*ARGA + AFUNCL*ARGB
      ENDIF
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
           DOUBLE PRECISION FUNCTION ATSL(TEMP,AFUNCL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      ARGA=SDKT(TEMP)+FDKT(TEMP)*FDKT(TEMP)*DAT(TEMP)*DLOG(10D0)
      ATSL=DAT(TEMP)*DLOG(10D0)*AFUNCL*ARGA
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
        DOUBLE PRECISION FUNCTION ATS(TEMP,AFUNCL,ADERFL,ADERSL)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NDATA=5000)
      REAL*8     BREAK(2*NDATA), CSCOEF(4,2*NDATA)
      COMMON/STRDATA/TO,SLOPE,TG,R,ITERATION
      COMMON/CCC/ BREAK, CSCOEF, NINTV, JDBUG
C
      COMMON/DBG3/SIGX3,ARGA3,DSIGX3,ARGB3,FACT3,SSIGX,ARGC
```

```
      COMMON/DBG3A/ PARTA3,PARTB3,PARTC3,FRA3,FRB3,FRC3
      EXTERNAL   DCSDER,DCSVAL

C
      IF(ITERATION.EQ.1)THEN
      ATS   =  ADERSL
      RETURN
                                                      ELSE
      SIGX  =  DCSVAL(TEMP, NINTV, BREAK, CSCOEF)
      SIGXF =  DCSDER(1, TEMP,NINTV,BREAK,CSCOEF)
      SIGXS =  DCSDER(2, TEMP,NINTV,BREAK,CSCOEF)
C
      ARGA  = SAT(TEMP,SIGX)
      ARGB  = SATF(TEMP,SIGX,SIGXF)
      ARGC  = SATS(TEMP,SIGX,SIGXF,SIGXS)
C
      ATS  = ADERSL*ARGA + 2.*ADERFL*ARGB + AFUNCL*ARGC
      ENDIF
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
               DOUBLE PRECISION FUNCTION FDKT(TEMP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      FDKT = -1000.0/(273.+TEMP)/(273.+TEMP)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
               DOUBLE PRECISION FUNCTION SDKT(TEMP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
C
      SDKT= 2.*1000.0/(273.+TEMP)/(273.+TEMP)/(273.+TEMP)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                       SUBROUTINE SERCH
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER    NTOT
      PARAMETER  (NTOT=33)
C
      COMMON/RED/ ZDATA(NTOT), XYDATA(2,NTOT), MDATA
```

```
      EXTERNAL    DSURF
C
      COMMON/BRK/       FDATA(NTOT), XDATA(NTOT), NDATA
      EXTERNAL   DQDVAL,DUMCGF
C
 800  PRINT*,' For one-dimensional minimization, input  1'
      PRINT*,'     two-dimensional minimizaiton,        2'
      READ(5,*) IANS
C
      IF(IANS.EQ.1) THEN
C
C --------->     Read data points:
C
 25     PRINT*,' Input number of data, NDATA =?'
      READ(5,*) NDATA
      DO 10  I=1, NDATA
      PRINT *, ' Input  ', I,' th XDATA and FDATA '
      READ(5,*)   XDATA(I), FDATA(I)
   10 CONTINUE
C
      CALL OPTONE(XOPT,YOPT)
      PRINT*,' To continue with added data, input 1'
      READ(5,*) IAD
C
      IF(IAD.EQ.1) THEN
      GOTO 25
      ENDIF
                                                      ELSE
C
C --------->     Set up X, Y, and F data:
C
      PRINT*,' Input MDATA : '
      READ(5,*) MDATA
      DO 100  I=1, MDATA
      PRINT*,' Input  ', I, 'th TF, TI AND  SIGMAF:  '
      READ(5,*) XYDATA(1,I), XYDATA(2,I), ZDATA(I)
  100 CONTINUE
C
      CALL OPTSEC
      ENDIF
C
      PRINT*,' For new game, input  1'
      READ(5,*) IG
      IF(IG.EQ.1)GOTO 800
      RETURN
      END
*
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                    SUBROUTINE ROSBRK(N,X,F)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
```

```
*
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER    NTOT
      PARAMETER  (NTOT=33)
C
      COMMON/BRK/ FDATA(NTOT), XDATA(NTOT), NDATA
      REAL*8 X(N)
      LOGICAL    CHECK
      EXTERNAL   DQDVAL
C
C --------->     Check XDATA:
C
      CHECK = .TRUE.
C
      XX=X(N)
      F = DQDVAL(XX,NDATA,XDATA,FDATA,CHECK)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                 SUBROUTINE OPTONE(XOPT, YOPT)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER    N
      PARAMETER  (N=1)
C
      REAL*8       DFPRED, FVALUE, G(N), GRADTL, X(N), XGUESS(N),
     &           XS(N)
      EXTERNAL   ROSBRK,  DUMCGF
C
      DATA XGUESS/ 1.0D0/, XS/1.0D0/
C
C


      DFPRED = 0.2
      GRADTL = 1.0D-6
      MAXFN  = 100
      WRITE(6,600) XGUESS,XS,DFPRED,GRADTL,MAXFN
C
      PRINT*,' To supply data, input 1'
      READ(5,*) ISUP
C
      IF(ISUP.EQ.1)THEN
      PRINT*,' Input XGUESS and XS: '
      READ(5,*)  XGUESS,XS
C
      PRINT*,' Input DFPRED and GRADTL:'
      READ(5,*)DFPRED ,GRADTL
C
```

```
      PRINT*,' Input MAXFN: '
      READ(5,*) MAXFN
C
      WRITE(6,600) XGUESS,XS,DFPRED,GRADTL,MAXFN
     ENDIF
C
C --------->     Minimize the Rosenbrock function:
C
      CALL DUMCGF (ROSBRK, N, XGUESS, XS, GRADTL, MAXFN, DFPRED, X, G,
     &             FVALUE)
C
      XOPT=X(1)
      YOPT=FVALUE
C
C --------->     Print the results:
C
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (X(I),I=1,N), FVALUE, (G(I),I=1,N)
C
C --------->      FORMAT:
C
 600  FORMAT(5X,'XGUESS = ',F12.3,6X,'XS = ',D12.6,/,
     +       5X,'DFPRED = ',D16.6,6X,'GRADTL = ',D16.6,/,
     +       5X,'MAXFN  = ',I6,/)
99999 FORMAT ('  The sclution is ', F8.3, //, '  The function ',
     &       'evaluated at the solution is ', F8.3, //, '  The ',
     &       'gradient is ', F8.3, /)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                        SUBROUTINE OPTSEC
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER    N
      PARAMETER  (N=2)
C
      REAL*8       DFPRED, FVALUE, G(N), GRADTL, X(N), XGUESS(N),
     &           XS(N)
      EXTERNAL   ROSRED,  DUMCGF
C
      DATA XGUESS/ 2*1.0D0/, XS/2*1.0D0/
C
C


      DFPRED = 0.2
      GRADTL = 1.0E-6
      MAXFN  = 100
      WRITE(6,600) XGUESS(1),XGUESS(2),XS(1),XS(2),DFPRED,GRADTL,MAXFN
```

```
C
      PRINT*,' To supply data, input 1'
      READ(5,*) ISUP
C
      IF(ISUP.EQ.1)THEN
      PRINT*,' Input TF guess and TI guess: '
      READ(5,*)  XGUESS(1),XGUESS(2)
C
      PRINT*,' Input DFPRED and GRADTL:'
      READ(5,*)DFPRED ,GRADTL
C
      PRINT*,' Input MAXFN: '
      READ(5,*) MAXFN
C
      WRITE(6,600) XGUESS(1),XGUESS(2),XS(1),XS(2),DFPRED,GRADTL,MAXFN
      ENDIF
C
C --------->      Minimize the Rosenbrock function:
C
      CALL DUMCGF (ROSRED, N, XGUESS, XS, GRADTL, MAXFN, DFPRED, X, G,
     &            FVALUE)
C
C --------->      Print the results:
C
      CALL UMACH (2, NOUT)
      WRITE (NOUT,99999) (X(I),I=1,N), FVALUE, (G(I),I=1,N)
C
C --------->      FORMAT:
C
 600  FORMAT(5X,' TF GUESS  = ',F12.3,6X,' TI GUESS  = ',F12.3,/,
     +       5X,' XS(1)     = ',D12.6,6X,' XS(2)     = ',D12.6,/,
     +       5X,'DFPRED = ',D16.6,6X,'GRADTL = ',D16.6,/,
     +       5X,'MAXFN  = ',I6,/)
99999 FORMAT ('  The solution is ', 2F8.3, //, '  The function ',
     &       'evaluated at the solution is ', F8.3, //, '  The ',
     &       'gradient is ', F8.3, /)
C
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                     SUBROUTINE ROSRED(N,X,F)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER  (NTOT=33, NXOUT=1, NYOUT=1, LDSUR=NXOUT)
C
      REAL*8 X(N)
      REAL*8        SUR(LDSUR,NYOUT), XOUT(NXOUT), YOUT(NYOUT)
C
      COMMON/RED/ ZDATA(NTOT), XYDATA(2,NTOT), MDATA
```

```
      EXTERNAL    DSURF, UMACH
C
      XOUT(1) = X(1)
      YOUT(1) = X(2)
C   20 CONTINUE
C
C
C ---------->      Interpolate scattered data:
C
      CALL DSURF (MDATA, XYDATA, ZDATA, NXOUT, NYOUT, XOUT, YOUT, SUR,
     &           LDSUR)
C
      F = SUR(1,1)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
      SUBROUTINE PSTRFILE(TIMEM,TEMPM,INUM,IDATD,INTV,NSTEP,KSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      INTEGER INUM(NM)
      REAL*8 TIMEM(0:NM),TEMPM(0:NM)
C
C  ---------->      PRINT EVERY (NSTEP/INTV) INTERVALS :
C
      NPTS=INT(NSTEP/INTV)
      IK=0
      WRITE(IDATD,400) IK,TIMEM(0),TEMPM(0)
      DO 350 K=1, NSTEP
      IK=INUM(K)
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDATD,400) IK,TIMEM(IK),TEMPM(IK)
 350  CONTINUE
      WRITE(IDAT,400) IK,TIMEM(KSTEP+1),TEMPM(KSTEP+1)
 400  FORMAT(2X,I5,5X,2(F16.9,2X,F16.9,2X))
      RETURN
      END
*
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
          SUBROUTINE PDSFILE(TIME,TIMEM,TEMP,TEMPM,INUM,IDATD,
     +               INTV,NSTEP,KSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      PARAMETER(NM=5000)
      INTEGER INUM(NM)
      REAL*8 TIME(0:NM),TEMP(0:NM),TIMEM(0:NM),TEMPM(0:NM)
C
C  --------->      PRINT EVERY (NSTEP/INTV) INTERVALS :
C
      NPTS=INT(NSTEP/INTV)
      WRITE(IDATD,400) TIME(0),TIMEM(0),TEMP(0),TEMPM(0)
      DO 350 K=1, NSTEP
      IK=INUM(K)
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDATD,400) TIME(K),TIMEM(IK),TEMP(K),TEMPM(IK)
 350  CONTINUE
      WRITE(IDATD,400) TIME(NSTEP+1),TIMEM(KSTEP+1),TEMP(NSTEP+1),
     +                 TEMPM(KSTEP+1)
 400  FORMAT(2X,4(F16.9,2X))
      RETURN
      END

*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            SUBROUTINE PSFILE(T,TEMP,DELT,IDAT,INTV,NSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER(NM=5000)
      REAL*8 T(0:NM),TEMP(0:NM),DELT(NM)
C
C  --------->      PRINT EVERY (NSTEP/INTV) INTERVALS :
C
      NPTS=INT(NSTEP/INTV)
      WRITE(IDAT,400) T(0),TEMP(0)
      DO 350 K=1, NSTEP
      IF(K.EQ.1.OR.K.EQ.NSTEP) GOTO 325
      ICHK=INT(K/NPTS)
      ICHK=ICHK*NPTS-K
      IF(ICHK.LT.0) GOTO 350
 325  WRITE(IDAT,400) T(K),TEMP(K),DELT(K)
 350  CONTINUE
      WRITE(IDAT,400) T(NSTEP+1),TEMP(NSTEP+1)
 400  FORMAT(2X,F16.9,2X,F16.9,2X,F16.9)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
            SUBROUTINE REDBUD (NSTEP,DELTMAX,DELTMIN,TIME)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
```

```
      PARAMETER (NM=5000)
      REAL*8 TIME(0:NM)
C
      DELTMAX=0D0
      DELTMIN=100D0
      DO 100 K=1, NSTEP-1
      BRICK = TIME(K+1)-TIME(K)
      IF(BRICK.GT.DELTMAX) DELTMAX=BRICK
      IF(BRICK.LT.DELTMIN) DELTMIN=BRICK
 100  CONTINUE
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
         SUBROUTINE BRUBUD (NSTEP,IPRT,DELTP,DELT,TIME,TEMP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NM=5000,IDATS=22)
      INTEGER IROW(NM),INUM(20),NUMS(10),NUME(10)
      REAL*8 TIME(0:NM),TEMP(0:NM),DELT(NM)
      DATA IROW/NM*0D0/

C
      CALL REDBUD (NSTEP,DELTMX,DELTMN,TIME)
      PRINT*
      PRINT*,' Number of steps = ', NSTEP
      PRINT*,' Maximum time interval = ',DELTMX
      PRINT*,' Minimum time interval = ',DELTMN
      PRINT*
      PRINT*,' Input proposed  Maximum Time Interval;  '
      READ(5,*) DELTP
C
C
      IK=0
      DO 100 K=1, NSTEP-1
      BRICK = TIME(K+1)-TIME(K)
      IF(BRICK.GT.DELTP) IROW(K)=1
C
      IF(K.GE.2)THEN
      CHECK = IROW(K) - IROW(K-1)
      CHECK = DABS(CHECK)
C
      IF(CHECK.GT.0.5) THEN
      IK=IK+1
      INUM(IK)=K
      ENDIF
      ENDIF
C
 100  CONTINUE
C
```

```
      IKH=IK/2
      DO 150 I=1, IKH
      NUMS(I)=INUM(2*I-1)
      NUME(I)=INUM(2*I)
 150  CONTINUE
C
      IF(IKH.EQ.0) RETURN
C
      PRINT*
      PRINT*,' Total number of intervals = ',IKH
      DO 175 K=1, IKH
      PRINT*
      WRITE(6, 635) K
C
      NUMSB=NUMS(K)-1
      NUMSN=NUMS(K)
      NUMSA=NUMS(K)+1
      NUMEB=NUME(K)-1
      NUMEN=NUME(K)
      NUMEA=NUME(K)+1
C
      PRINT*
      PRINT*,' NUMSN = ', NUMSN,'  AND NUMEN = ',NUMEN
      PRINT*,' time step proposed is ', DELTP
      WRITE(6,600)
      WRITE(6,650) NUMSB,TIME(NUMSB),DELT(NUMSB)
      WRITE(6,650) NUMSN,TIME(NUMSN),DELT(NUMSN)
      WRITE(6,650) NUMSA,TIME(NUMSA)
      PRINT*
      WRITE(6,650) NUMEB,TIME(NUMEB),DELT(NUMEB)
      WRITE(6,650) NUMEN,TIME(NUMEN),DELT(NUMEN)
      WRITE(6,650) NUMEA,TIME(NUMEA)
      PRINT*
      PRINT*,' To continue, input any number. '
      READ(5,*) ICONT
 175  CONTINUE
C
      PRINT*
      PRINT*,' Input 1 to print data for modified time intervals.'
      READ(5,*) IPRT
C
      IF(IPRT.EQ.1)THEN
      WRITE(IDATS, 625) IKH
      DO 185 K=1, IKH
      WRITE(IDATS, 635)  K
      NUMSB=NUMS(K)-1
      NUMSN=NUMS(K)
      NUMSA=NUMS(K)+1
      NUMEB=NUME(K)-1
      NUMEN=NUME(K)
      NUMEA=NUME(K)+1
```

```
C
      WRITE(IDATS,610) NUMSN,NUMEN,DELTP
      WRITE(IDATS,600)
      WRITE(IDATS,650) NUMSB,TIME(NUMSB),DELT(NUMSB)
      WRITE(IDATS,650) NUMSN,TIME(NUMSN),DELT(NUMSN)
      WRITE(IDATS,650) NUMSA,TIME(NUMSA)
      WRITE(IDATS,*)
      WRITE(IDATS,650) NUMEB,TIME(NUMEB),DELT(NUMEB)
      WRITE(IDATS,650) NUMEN,TIME(NUMEN),DELT(NUMEN)
      WRITE(IDATS,650) NUMEA,TIME(NUMEA)
 185  CONTINUE
C
      INTV=NSTEP
      WRITE(IDATS,660)
      CALL PSFILE(TIME,TEMP,DELT,IDATS,INTV,NSTEP)
      ENDIF
C
C --------->      FORMAT:
C
 600  FORMAT(/,5X,'NOS',10X,'TIME',10X,'DELT',/)
 610  FORMAT(/,5X,' NUMS = ', I5,'  AND NUME = ',I5,/,
     +         5X,' time step proposed is ', F12.6)
 625  FORMAT(/,5X,' TOTAL NUMBER OF INTERVALS  = ',I5)
 635  FORMAT(/,5X,I5,'TH  INTERVALS: ',//)
 650  FORMAT(I5,5X,F12.3,5X,D16.6 )
 660  FORMAT(/,10X,'TIME', 10X,'TEMPARATURE',10X,'DELT',/,
     +25X,'MODIFIED')
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
        SUBROUTINE XQMOD (NSTEP,KSTEP,ISTEP,INUM,DELTP,TIME,
     +             TEMP ,TIMEM,TEMPM,DELT)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NM=5000)
      INTEGER INUM(NM)
      REAL*8 TIME(0:NM),TEMP(0:NM),TIMEM(0:NM),TEMPM(0:NM),DELT(NM)
C
      TIMEM(0)=TIME(0)
      TEMPM(0)=TEMP(0)
C
      KSTEP=0
      DO 100 K=1, ISTEP-1
      BRICK = TIME(K+1)-TIME(K)
C
      IF(BRICK.LE.DELTP) THEN
      KSTEP=KSTEP+1
      INUM(K)=KSTEP
```

```
      TIMEM(KSTEP)=TIME(K)
      TEMPM(KSTEP)=TEMP(K)
                                                        ELSE
      IDIV=(TIME(K+1)-TIME(K))/DELTP
      IDIV=IDIV+1
      DELTM=(TIME(K+1)-TIME(K))/IDIV
      DELPM=(TEMP(K+1)-TEMP(K))/IDIV
      DO 50 J=1, IDIV
      KSTEP=KSTEP+1
      IF(J.EQ.1)INUM(K)=KSTEP
      TIMEM(KSTEP)=TIME(K)+DELTM*(J-1)
      TEMPM(KSTEP)=TEMP(K)+DELPM*(J-1)
 50   CONTINUE
      ENDIF
C
 100  CONTINUE
C
      DO 150 K=ISTEP, NSTEP
      KSTEP=KSTEP+1
      INUM(K)=KSTEP
      TIMEM(KSTEP)=TIME(K)
      TEMPM(KSTEP)=TEMP(K)
 150  CONTINUE
C
      DO 200 I=1, KSTEP
      DELT(I)=TIMEM(I+1)-TIMEM(I)
 200  CONTINUE
C
      TIMEM(KSTEP+1)=TIME(NSTEP+1)
      TEMPM(KSTEP+1)=TEMP(NSTEP+1)
C
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
               SUBROUTINE XQSRSS (X,Y,SQSS,NSTEP )
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NM=5000)
      REAL*8 X(0:NM),Y(0:NM)
C
      SQSS=0D0
C
      DO 100 I=1, NSTEP
      SQSS=(X(I)-Y(I))*(X(I)-Y(I))+SQSS
 100  CONTINUE
      SQSS=SQSS**0.5
      SQSS=SQSS/NSTEP
      RETURN
      END
```

```
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
         SUBROUTINE  CHGSTR(STRESSM,STRESS,INUM,NSTEP,KSTEP)
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NM=5000)
      INTEGER INUM(NM)
      REAL*8 STRESSM(0:NM),STRESS(0:NM)
C
      DO 100 I=1, NSTEP
      IK=INUM(I)
      STRESS(I)=STRESSM(IK)
 100  CONTINUE
      STRESS(NSTEP+1)=STRESSM(KSTEP+1)
      RETURN
      END
*
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
                     SUBROUTINE REPXY (X,Y,NSTEP )
* + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + + +
*
      IMPLICIT REAL*8(A-H,O-Z)
      PARAMETER (NM=5000)
      REAL*8 X(0:NM),Y(0:NM)
C
      DO 100 I=1, NSTEP+1
      Y(I)=X(I)
 100  CONTINUE
      RETURN
      END
```

Appendix B

Expressions for $\bar{N}x$, $\bar{N}y$, $\bar{M}x$, $\bar{M}y$ and their derivatives with respect to ϵ_x^0, ϵ_y^0, Kx, and Ky

Substitution of Equation 6.4 into Equation 6.5 yields

$$N_x = \sum_{i=-n}^{n} (P_x)_i \tag{B.1}$$

$$N_y = \sum_{i=-n}^{n} (P_y)_i \tag{B.2}$$

$$M_x = \sum_{i=-n}^{n} (T_x)_i \tag{B.3}$$

$$M_y = \sum_{i=-n}^{n} (T_y)_i \tag{B.4}$$

where

$$\begin{aligned} ((P_x)_i, (T_x)_i) &= \int_{h_{i-1}}^{h_i} \left[Q_{11}^{(i)}(\epsilon_x^o + zK_x + Q_{12}^{(i)}(\epsilon_y^o + zK_y) \right] (1, z)dz \\ ((P_y)_i, (T_y)_i) &= \int_{h_{i-1}}^{h_i} \left[Q_{12}^{(i)}(\epsilon_x^o + zK_x + Q_{22}^{(i)}(\epsilon_y^o + zK_y) \right] (1, z)dz \end{aligned} \tag{B.5}$$

Introduce now the nondimensional quantities $k_x = hK_x$, $k_y = hK_y$, $l_k = h_k/h$ and $\rho = E_T/E_L^o$, where h is half the laminate's thickness.

Employing expressions 6.1 and 6.3 and the definition for r and performing the integrations indicated in Equation B.5 we obtain :

B.1 (a) For the 0^0 plies:

$$\frac{1}{r(E_L^0 h)}(P_x)_k^0 = 3(A_4\chi_k + A_2)K_x{}^3(l_k^4 - l_{k-1}^4) + 4(3(A_4\chi_k + A_2)\epsilon_x^0 +$$

$$A_3\chi_k + A_1)K_x{}^2(l_k^3 - l_{k-1}^3) + 6(\rho\nu_{LT}K_y + (3(A_4\chi_k + A_2)\epsilon_x^{0\,2} +$$
$$2(A_3\chi_k + A_1)\epsilon_x^0 + A_0\chi_k + 1)K_x)(l_k^2 - l_{k-1}^2) + 12(\rho\nu_{LT}\epsilon_y^0 + (A_4\chi_k + A_2)\epsilon_x^{0\,3} +$$
$$(A_3\chi_k + A_1)\epsilon_x^{0\,2} + (A_0\chi_k + 1)\epsilon_x^0)(l_k - l_{k-1})/12$$

$$\frac{1}{r(E_L^0 h)}(P_y)_k^0 = \rho\left((K_y + \nu_{LT}K_x)(l_k^2 - l_{k-1}^2) + 2\rho(\epsilon_y^0 + \nu_{LT}\epsilon_x^0)(l_k - l_{k-1})\right)/2$$

$$\frac{1}{r(E_L^0 h^2)}(T_x)_K^0 = (12(A_4\chi_k + A_2)K_x{}^3(l_k^5 - l_{k-1}^5) +$$
$$15(3(A_4\chi_k + A_2)\epsilon_x^0 + A_3\chi_k + A_1)K_x{}^2(l_k^4 - l_{k-1}^4) +$$
$$20(\nu_{LT}\rho k_y + (3(A_4\chi_k + A_2)\epsilon_x^{0\,2} + 2(A_3\chi_k + A_1)\epsilon_x^0 +$$
$$A_0\chi_k + 1)K_x(l_k^3 - l_{k-1}^3) + 30(\nu_{LT}\rho\epsilon_y^0 + (A_4\chi_k + A_2)\epsilon_x^{0\,3} +$$
$$(A_3\chi_k + A_1)\epsilon_x^{0\,2} + (A_0\chi_k + 1)\epsilon_x^0)(l_k^2 - l_{k-1}^2))/60$$

$$\frac{1}{r(E_L^0 h^2)}(T_y)_K^0 = (2\rho(K_y + \nu_{LT}K_x)(l_k^3 - l_{k-1}^3) +$$
$$3\rho(\epsilon_y^0 + \nu_{LT}\epsilon_x^0)(l_k^2 - l_{k-1}^2))/6$$

$$\frac{1}{r(E_L^0 h)}\left(\frac{\partial P_x}{\partial \epsilon_x^0}\right)_k^0 = (A_4\chi_k + A_2)K_x{}^2(l_k^3 - l_{k-1}^3) + (3(A_4\chi_k + A_2)\epsilon_x^0 + A_3\chi_k +$$
$$A_1)K_x(l_k^2 - l_{k-1}^2) + (3(A_4\chi_k + A_2)\epsilon_x^{0\,2} + 2(A_3\chi_k + A_1)\epsilon_x^0 + A_0\chi_k + 1)$$
$$(l_k - l_{k-1})$$

$$\frac{1}{(E_L^0 h)}\left(\frac{\partial P_x}{\partial \epsilon_y^0}\right)_k^0 = \frac{1}{(E_L^0 h)}\left(\frac{\partial P_y}{\partial \epsilon_x^0}\right)_k^0 = r\nu_{LT}\rho(l_k - l_{k-1})$$

$$\frac{1}{r(E_L^0 h^2)}\left(\frac{\partial P_x}{\partial K_x}\right)_k^0 = 9(A_4\chi_k + A_2)K_x{}^2(l_k^4 - l_{k-1}^4) + 8(3(A_4\chi_k + A_2)\epsilon_x^0 +$$
$$A_3\chi_k + A_1)K_x(l_k^3 - l_{k-1}^3) + 6(3(A_4\chi_k + A_2)\epsilon_x^{0\,2} + 2(A_3\chi_k + A_1)\epsilon_x^0 +$$
$$A_0\chi_k + 1)(l_k^2 - l_{k-1}^2))/12$$

$$\frac{1}{r(E_L^0 h)}\left(\frac{\partial P_y}{\partial \epsilon_y^0}\right)_k^0 = \rho(l_k - l_{k-1})$$

$$\frac{1}{(E_L^0 h^2)}\left(\frac{\partial P_y}{\partial K_x}\right)_k^0 = \frac{1}{(E_L^0 h^2)}\left(\frac{\partial T_y}{\partial \epsilon_x^0}\right)_k^0 = \frac{1}{(E_L^0 h^2)}\left(\frac{\partial P_x}{\partial K_y}\right)_k^0 =$$
$$\frac{1}{(E_L^0 h^2)}\left(\frac{\partial T_x}{\partial \epsilon_y^0}\right)_k^0 = r\nu_{LT}\rho(l_k^2 - l_{k-1}^2)/2$$

$$\frac{1}{(E_L^0 h^2)}\left(\frac{\partial P_y}{\partial K_y}\right)_k^0 = \frac{1}{(E_L^0 h^2)}\left(\frac{\partial T_y}{\partial \epsilon_y^0}\right)_k^0 = r\rho(l_k^2 - l_{k-1}^2)/2$$

$$\frac{1}{r(E_L^0 h^2)}\left(\frac{\partial T_x}{\partial \epsilon_x^0}\right)_k^0 = (9(A_4\chi_k + A_2)K_x{}^2(l_k^4 - l_{k-1}^4) + 8(3(A_4\chi_k + A_2)\epsilon_x^0 + A_3\chi_K + A_1)K_x(l_k^3 - l_{k-1}^3) + 6(3(A_4\chi_k + A_2)\epsilon_x^{0\,2} + 2(A_3\chi_k + A_1)\epsilon_x^0 + A_0\chi_k + 1)(l_k^2 - l_{k-1}^2))/12$$

$$\frac{1}{r(E_L^0 h^3)}\left(\frac{\partial T_x}{\partial K_x}\right)_k^0 = (18(A_4\chi_k + A_2)K_x{}^2(l_k^5 - l_{k-1}^5) + 15(3(A_4\chi_k + A_2)\epsilon_x^0 + 3A_3\chi_k + A_1)K_x(l_k^4 - l_{k-1}^4) + 10(3(A_4\chi_k + A_2)\epsilon_x^{0\,2} + 2(A_3\chi_k + A_1)\epsilon_x^0 + A_0\chi_k + 1)(l_k^3 - l_{k-1}^3))/30$$

$$\frac{1}{(E_L^0 h^3)}\left(\frac{\partial T_x}{\partial K_y}\right)_k^0 = \frac{1}{(E_L^0 h^3)}\left(\frac{\partial T_y}{\partial K_x}\right)_k^0 = r\nu LT\rho(l_k^3 - l_{k-1}^3)/3$$

$$\frac{1}{r(E_L^0 h^3)}\left(\frac{\partial T_y}{\partial K_y}\right)_k^0 = \rho(l_k^3 - l_{k-1}^3)/3$$

B.2 (b) For the 90^0 plies:

$(P_y)_k^{90}$, $(P_X)_k^{90}$, $(T_y)_k^{90}$ and $(T_x)_k^{90}$ are obtained from $(P_x)_k^0$, $(P_y)_k^0$, $(T_x)_k^0$ and $(T_y)_k^0$, respectively, by the interchanges $\epsilon_x^0 \leftrightarrow \epsilon_y^0$ and $k_x \leftrightarrow k_y$.

The same interchange yields $\left(\frac{\partial P_y}{\partial \epsilon_y^0}\right)_k^{90}$ from $\left(\frac{\partial P_x}{\partial \epsilon_x^0}\right)_k^0$, $\left(\frac{\partial P_y}{\partial K_y}\right)_k^{90}$ from $\left(\frac{\partial P_x}{\partial K_x}\right)_k^0$, $\left(\frac{\partial T_y}{\partial \epsilon_y^0}\right)_k^{90}$ from $\left(\frac{\partial T_x}{\partial \epsilon_x^0}\right)_k^0$ and $\left(\frac{\partial T_y}{\partial K_y}\right)_k^{90}$ from $\left(\frac{\partial T_x}{\partial K_x}\right)_k^0$.

In addition,

$$\left(\frac{\partial P_x}{\partial \epsilon_x^0}\right)_k^{90} = \left(\frac{\partial P_y}{\partial \epsilon_y^0}\right)_k^0$$

$$\left(\frac{\partial P_x}{\partial \epsilon_y^0}\right)_k^{90} = \left(\frac{\partial P_y}{\partial \epsilon_x^0}\right)_k^{90} = = \left(\frac{\partial P_x}{\partial \epsilon_y^0}\right)_k^0$$

$$\left(\frac{\partial P_x}{\partial K_x}\right)_k^{90} = \left(\frac{\partial T_x}{\partial \epsilon_x^0}\right)_k^{90} = \left(\frac{\partial P_y}{\partial K_y}\right)_k^0$$

$$\left(\frac{\partial P_x}{\partial K_y}\right)_k^{90} = \left(\frac{\partial P_y}{\partial K_x}\right)_k^{90} = \left(\frac{\partial T_x}{\partial \epsilon_y^0}\right)_k^{90} = \left(\frac{\partial T_y}{\partial \epsilon_x^0}\right)_k^{90} = \left(\frac{\partial P_y}{\partial K_x}\right)_k^0$$

$$\left(\frac{\partial T_y}{\partial K_x}\right)_k^{90} = \left(\frac{\partial T_x}{\partial K_y}\right)_k^{90} = \left(\frac{\partial T_x}{\partial K_y}\right)_k^0$$

$$\left(\frac{\partial T_x}{\partial K_x}\right)_k^{90} = \left(\frac{\partial T_y}{\partial K_y}\right)_k^0$$

In all the above expressions, χ_k indicates the degree of crystallinity at the center of the $K^{\underline{th}}$ ply.

VITA

Kyun Lee was born in Seoul, Korea on October 24, 1952. He attended Chung-Un elementary school and graduated from Choong-Ang high school in Seoul, Korea in 1971. He received the Bachelor of Science with a major in Civil Engineering at Seoul National University in 1975, and the Master of Engineering with a major in Civil Engineering at Texas A & M University in 1985. He will receive the Ph.D degree with a major in Engineering Science at University of Tennessee, Knoxville in December, 1991.

He served the military service as a duty of the citizen in Korea to 1977 from 1975 and discharged honorably as a Lieutenant in 1977. He was employed at the Dae-Lim Inustry to 1979 from 1977 as a civil engineer and at the Korea Nuclear Engineering to 1980 from 1979 as a structural engineer. He is married to Hae-Ok and has one son, Jung-Woo.