NUMERICAL DETERMINATION OF FORCES ACTING ON MATERIAL INTERFACES: AN APPLICATION TO RAFTING IN Ni-SUPERALLOYS

by

Simona Socrate

Laurea in Nuclear Engineering
University of Rome
(Rome, Italy, 1984)

Submitted in partial fulfillment of the
requirements for the degree of Master of Science in
Mechanical Engineering

at the

Massachusetts Institute of Technology
August, 1990

Signature of Author____

Department of Mechanical Engineering
August, 1990

Certified by____

David M. Parks
Associate Professor of Mechanical Engineering
Thesis Supervisor

Accepted by____

Ain A. Sonin,Chairman
Departmental Committee on Graduate Studies

NUMERICAL DETERMINATION OF FORCES ACTING ON MATERIAL INTERFACES: AN APPLICATION TO RAFTING IN Ni-SUPERALLOYS

by

Simona Socrate

Submitted to the Department of Mechanical Engineering
on August 10, 1990 in partial fulfillment of the requirements
for the Degree of Master of Science in Mechanical Engineering

ABSTRACT

Numerical techniques have been developed to evaluate local driving forces acting on material interfaces. Since migrations of the interfaces are associated with modifications of the microstructure, these methods can be applied to predict and model morphological evolution in multi-phase materials. Here this methodology has been applied to the study of an evolution of the morphology of γ' precipitates in Ni-superalloys, which has been termed *rafting*.

It is shown how predictions of the model agree with available experimental data, and a comprehensive treatment of the rafting phenomenon is proposed.

Thesis Supervisor: Dr. David M. Parks
Title: Associate Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I would like to express my sincerest gratitude to Professor David M. Parks for his guidance and support during this project. Not only his uncommon knowledge and understanding have always been offered when needed, but, most important, his generosity and friendship have been an invaluable help through any difficult situation.

I also extend my thanks to Professors Ali Argon, Rohan Abeyaratne and Mary Boyce for their help through numerous discussions.

Thanks to my office mates, and to all the Mechanics of Materials research group, for creating a nice working environment, for being always willing to lend a hand, for keeping together our computer facility and for watering my plants when I forget to.

A special thank to Mary Toscano, without whom this manuscript wouldn't exist. She is nice, patient and amazingly efficient: three qualities that rarely coexist in one person.

I would also like to thank Leslie Regan and Joan Kravit for having always been extremely friendly and helpful.

I think I should also thank Dr. R. G. S. P. Stringfellow. Thank you Richy for having been always at my side, for your *paramount* help and support and for having made my life worth living.

Finally, I would like to thank my mother and sister. I will have to do this in Italian since my mother is still at her third English lesson and my sister has just got a "C" in her latest English exam.

Grazie mamma e Carla per avermi sempre aiutato e incoraggiato anche quando le mie scelte hanno portato dolore e sacrifici. Siete sempre nel mio cuore e questo Master é per voi (e per papa se mi può vedere in qualche modo).

This work was supported by the Ida Green Fellowship and by the M.I.T. Center for Material Science and Engineering under National Science Foundation Grant No. DMR-87-19217.

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

Page

LIST OF FIGURES

No. Page

INTRODUCTION

The mechanical and physical properties of any material depend critically on two parameters. The first is its *constitution*: the overall composition, the number of phases and their relative volume fractions and compositions. The second is its *microstructure*: the shape, size and distribution of each phase.

The application of external loads to a multi-phase crystalline solid can significantly alter its microstructure.

Limiting our attention to the most common case of a two-phase (matrix/precipitate) alloy, an external stress can modify the morphology of the precipitates, influence the precipitate coarsening kinetics and alter the relative stability of the two phases.

The influence of the applied loading conditions on the morphology and stability of the microstructure depends on the material parameters of the precipitate and of the matrix, and on the magnitude and nature of the applied loads.

Since many of the properties of the alloy are determined by its microstructure, it is of technological relevance to be able to predict microstructural development. The microstructure evolution of the precipitate phase can be understood on the basis of simple energy arguments. When the morphology of the microstructure evolves, it is because there is a *driving force* for the change: the system is trying to lower its total energy by modifying its configuration.

Obviously, the mere presence of a driving force does not guarantee that a change will occur: the kinetics of the process will dictate the pace at which the structural change will eventually take place. Nevertheless, quantifying the driving force for the morphological evolution is still the essential first step toward a complete modeling of the phenomenon.

In analyzing the evolution of the crystal microstructure, we can focus our attention on the migration of the interfaces between different phases: an evolution of the microstructure corresponds to a migration of the interfaces. Thus we can think of the driving forces for microstructure development as forces acting on the interfaces and driving their migration.

The objective of this research has been to develop numerical techniques for the determination of the forces acting on material interfaces, and to apply these techniques to the analysis of morphological evolution in $\gamma - \gamma'$ Nickel superalloys.

We will first describe, in chapter 1, the phenomenon that we intend to analyze: an evolution of the morphology of γ' precipitates in Ni-superalloys, which has been termed *rafting*. We will present a brief historical review of the experimental observations and discuss the models which have been proposed in the literature.

In chapter 2 we will define, in a more rigorous context, the notion of *force on a material interface*, and in chapter 3 we will introduce numerical techniques to evaluate it.

In chapter 4 we will apply these techniques to the analysis of rafting in $\gamma - \gamma'$ Ni-superalloys; we will compare the predictions of our model with available experimental data and present a general discussion of the rafting process.

Finally, in chapter 5, we will draw the conclusions of this research and discuss some suggestions for future studies.

CHAPTER 1
RAFTING IN $\gamma - \gamma'$ Ni-SUPERALLOYS

1.1 Introduction

Nickel-base superalloy single crystals are a class of two-phase precipitation-strengthened materials.

The microstructure consists of an fcc nickel solid solution matrix, γ, with Ni_3Al precipitates, γ'. The precipitate phase, an fcc intermetallic compound, is ordered and coherent with the matrix and can constitute up to 70 percent of the volume of the crystal. After complete aging the precipitates are distributed in the γ matrix as a stable periodic array of $< 100 >$ aligned cuboids of fairly uniform size, usually in the range of 0.2 - 0.5 μm in diameter (Figure 1.1)

Typically, a lattice parameter mismatch, δ, exists between the γ and γ' phases:

$$\delta = \frac{a_{\gamma'} - a_{\gamma}}{a_{\gamma}}, \tag{1.1}$$

where $a_{\gamma'}$ and a_{γ} are the lattice parameters of the two phases.

In most commercial alloys the magnitude of the misfit is minimal ($|\delta|$ less than 0.5%); nevertheless, since the matrix-precipitate interface is coherent, the lattice misfit introduces a significant internal stress in the crystal (Figure 1.2).

The $\gamma - \gamma'$ misfit, which is temperature-dependent due to the differing coefficients of thermal expansion in the two phases, has remarkable effects on the evolution of the morphology of the γ' precipitates.

Nathan [1] observed that, during the aging treatment, the morphology change, from spherical to cuboid, occurs at smaller sizes of the precipitates for higher levels of misfit.

But the most dramatic effect of the magnitude and sense of the misfit has been recognized in a phenomenon, called *rafting*, which has been observed by various researchers during stress-annealing experiments or during creep tests at elevated temperatures.

At high temperatures – above 900°C for most commercial alloys – the γ' cuboidal precipitates become unstable. The cubic γ' particles link together to form rods and/or plates (rafts).

When the crystal is annealed in the absence of an applied stress, the new lamellar structure is randomly oriented along the three $< 100 >$ cube directions, without showing a preferential orientation (Figure 1.3(a)).

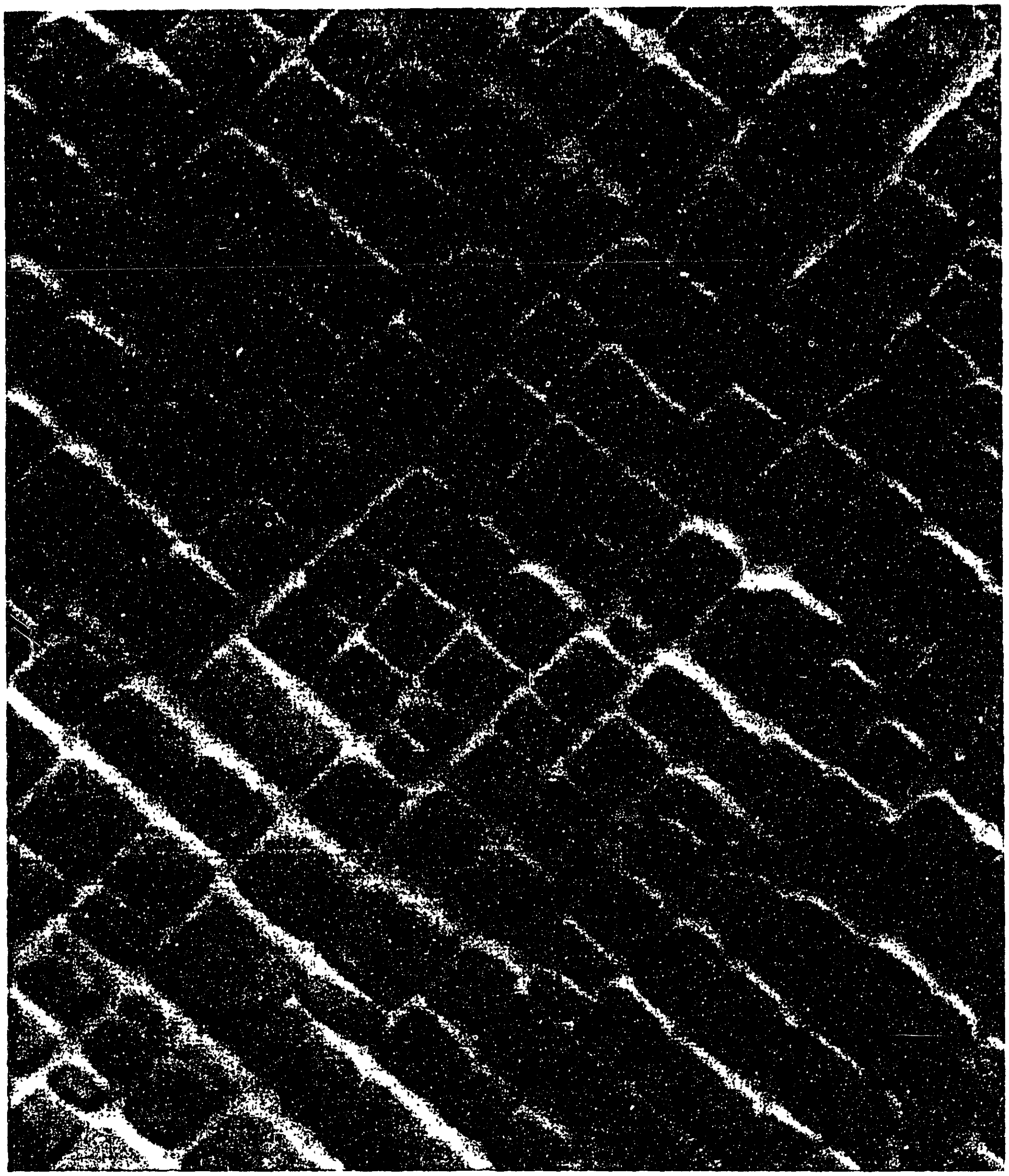


Figure 1.1 Typical microstructure of the fully heat treated single crystal [30].

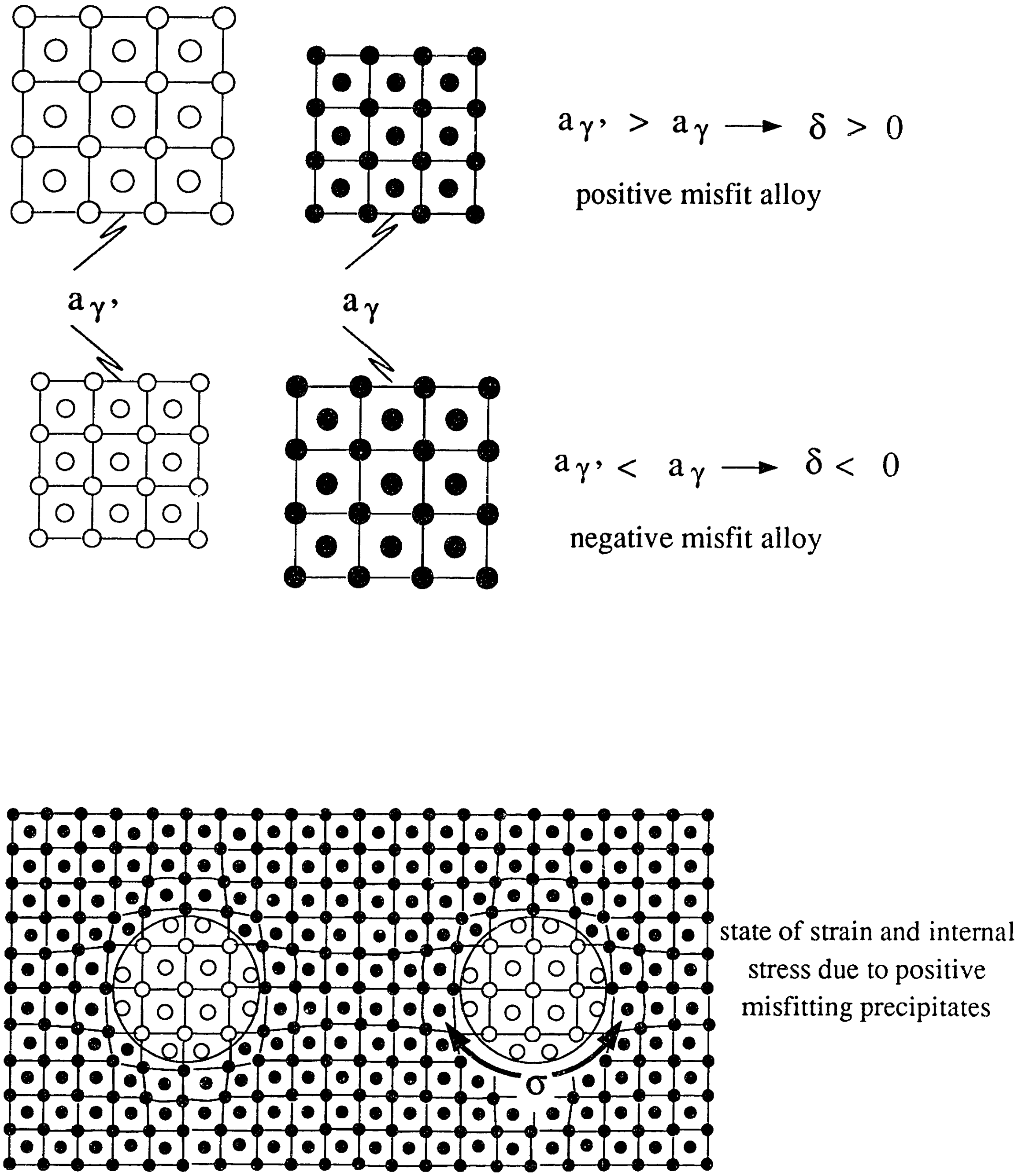

$a_{\gamma'} > a_\gamma \longrightarrow \delta > 0$
positive misfit alloy
$a_{\gamma'}$
a_γ
$a_{\gamma'} < a_\gamma \longrightarrow \delta < 0$
negative misfit alloy
σ
state of strain and internal stress due to positive misfitting precipitates

Figure 1.2 Lattice misfit and its effect on the internal stress state.

When a tensile or compressive stress is applied along the $< 001 >$ direction, the morphological evolution of the precipitates exhibits a marked *directionality.* Two different types of behavior for the coarsening of the precipitates have been observed:

- Type P (Parallel): the cuboids coarsen preferentially along the direction of the applied stress: the precipitates assume the form of plates which lie parallel to the stress direction. In alloys with small volume fraction of γ', strings of γ' cubes can coarsen along the $< 001 >$ direction and form rods (Figure 1.3(b)).
- Type N (Normal): the cuboids coarsen preferentially along the directions normal to the applied stress: the precipitates assume the form of broad flat plates with their faces *normal* to the stressed $< 001 >$ direction (Figure 1.3(c))

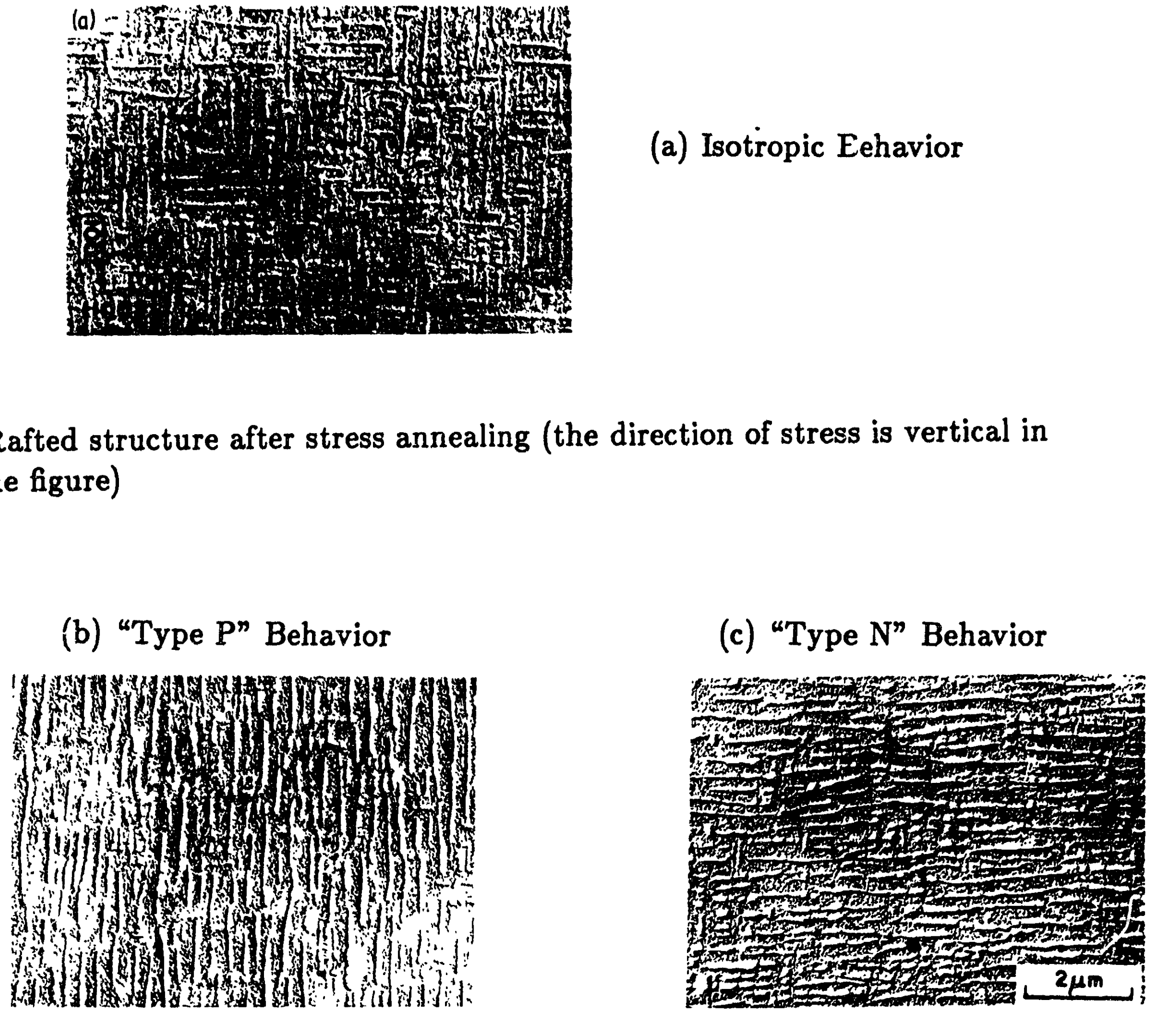
Several rafting observations have been reported over the last two decades [2-9,12,13,15-25]. According to these observations, different materials can coarsen in opposite directions under the same loading conditions.

The lattice misfit was soon recognized as a key parameter controlling the rafting behavior of the alloys.

Since the morphology of the precipitates strongly affects the mechanical properties of the alloy, several studies have been carried out in order to model, predict and control the γ' morphology evolution.

In the following paragraphs, we give a brief survey of rafting observations, identify possible approaches to analyze the phenomenon, and discuss the models proposed in the literature.

(1) Rafted structure after annealing in absence of an applied stress.



(a) Isotropic Eehavior

(2) Rafted structure after stress annealing (the direction of stress is vertical in the figure)

(b) "Type P" Behavior

(c) "Type N" Behavior

Figure 1.3 Directional coarsening of γ' precipitates [15].

1.2 Experimental Observations

Directional coarsening of the γ' precipitates was first observed in commercial Ni-superalloys after prolonged creep exposure [2-9].

These instabilities of the γ' phase were viewed initially with some concern since it was observed that they were generally associated with a reduction in the creep resistance of the alloy [5,9,10,11]. This led to a number of theoretical and experimental studies, toward a better understanding of the rafting phenomenon [12-15]. These studies, which will be discussed in greater detail in paragraph 1.4, identified some of the major factors related to the stress coarsening behavior, and suggested that rafting of the γ' precipitates could be essentially eliminated by reducing the $\gamma - \gamma'$ lattice misfit through careful alloy design.

In the early '80s, Pearson, Kear and Lemkey [18,19] presented an innovative study where they demonstrated that directional coarsening significantly *enhanced* the high temperature creep properties of an experimental alloy with an unusually high negative value of the misfit ($\delta = -0.78\%$).

These results brought about a wave of renewed attention for the rafting phenomenon. A number of investigations were undertaken concerning the actual development, under different testing conditions, of the γ' rafts and their influence on the creep properties of the crystal [16-25]. The creep loads were generally applied along the $< 001 >$ crystal direction. While in most studies only tensile loads were considered, in some experiments the effect of compressive loads was also investigated.

Here we will briefly review some of the results of these observations, together with the models proposed to explain the observed variation in creep resistance.

1. The rafts begin to form early in primary creep [17-24,27]. The time needed to attain a fully-developed lamellar structure appears to be influenced by:

   (a) the test temperature: the rate of directional coarsening increases when the test temperature is raised [24];

   (b) the applied load : upon increasing the magnitude of the applied load a hastening of the rafting process is observed [24];

   (c) the lattice misfit: under the same conditions of applied load and test temperature, alloys with larger magnitude of misfit exhibit a higher rate of directional coarsening [27];

   (d) the initial microstructure: a fine microstructure with closely-spaced, small-sized γ' precipitates considerably hastens the development of rafts [22].

2. The rafted configuration is very stable [18,20,22,24]. The average thickness of the rafts is initially very close to the original γ' particle size [17-25]. As the creep

transient progresses, contradictory observations have been reported concerning the evolution of raft thickness and interlamellar spacing.

Some researchers [22,24,25] report that the thickness of the rafts remains constant up to the onset of tertiary creep and the interlamellar spacing also shows a similar behavior, while other research groups [18,20,23] have observed a gradual thickening of the rafted lamellae during steady state creep.

Differences in the alloy composition could be responsible for these discrepancies: alloys with higer levels of refractory elements are characterized by reduced diffusion rates so that, for these alloys, the thickening of the lamellae might be hindered [24].

In tertiary creep the rafts become irregularly shaped , lose their perfect alignment and coarsen considerably prior to failure.

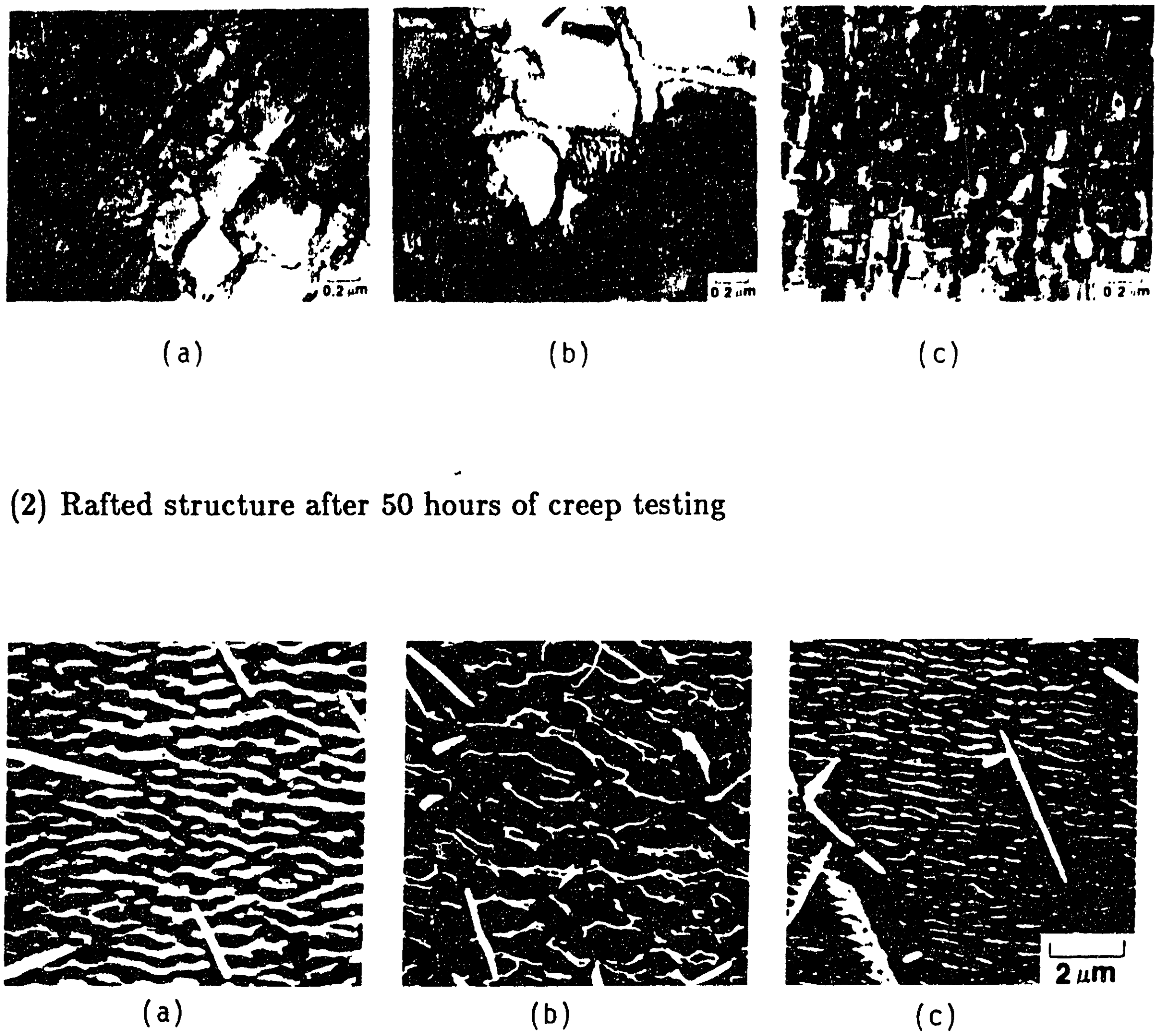
3. The initial microstructure prior to testing can drastically affect the resulting rafted morphology [18, 19, 21, 22]. An ordered, perfectly-aligned structure of γ' cuboids promotes a rapid formation of more perfect platelets with a high aspect ratio. Since the initial thickness of the rafts basically coincides with the original dimension of the γ' cuboids, a finer initial microstructure will produce a finer rafted structure. In contrast, if the initial structure is overaged, and is characterized by irregularly-shaped γ' particles, the raft morphology will appear very irregular as well, with a very low average aspect ratio (Fig. 1.4).

4. A characteristic feature of the stress-coarsened structure is the presence of networks of dislocations at the $\gamma - \gamma'$ interfaces [17, 18, 19, 21, 23, 25]. These dislocations are true misfit dislocations since their Burgers' vectors are appropriate for relaxing the internal stress due to the misfit. Merging interfaces of coarsening cuboids are usually deprived of dislocations [23, 25].

5. Under *low stress*, at high temperatures, the operative creep mechanism involves dislocation motion primarily in the γ-matrix with the mobile dislocations circumventing the γ' particles which remain virtually dislocation-free [18, 19, 21, 22, 25]. Thus, γ'-rafts with a high aspect ratio provide an ideal structure for creep resistance because circumvention of the γ' phase is eliminated. Significant creep can occur only by insertion of dislocations through the ordered intermetallic γ' phase, resulting in dramatically improved creep resistance [18, 19, 22]. Furthermore, the misfit dislocation networks at the $\gamma - \gamma'$ interfaces act as obstacles to the penetration of the dislocations inside the precipitates.

   Under *high stresses*, at high temperatures, γ' particle shearing tends to become the prevailing creep mode [18, 19]. For these loading conditions, a directionally-coarsened structure may not be beneficial since one set of $\gamma - \gamma'$ interfaces is essentially eliminated [24].

(1) Morphology of the precipitates prior to creep test



(2) Rafted structure after 50 hours of creep testing

Figure 1.4 Effect of the initial microstructure on the coarsening process [24].

We can thus think of two possible explanations for the earlier observations [5, 9, 10, 11] in which rafted crystals did not exhibit an improvement in their creep resistance. First, in some studies the yield behavior of the alloy was investigated [11] so that the tests were conducted at stress levels well inside the range in which γ' particle shearing is the predominant mechanism of creep. Second, for the tests conducted in the low stress regimes, the overaged, irregular and coarser microstructure of the γ' rafts in these early alloys was easily circumvented by the dislocations in the γ phase [22].

Finally, a number of observations concerning the direction of coarsening of the γ' precipitates under tensile and compressive creep loads applied along the $< 001 >$ crystal direction are schematically summarized in Table 1.1.

Here, assuming that the direction of the $< 001 >$ applied load is as shown, we graphically identify a "type N" behavior with a horizontal rectangle and a "type P" behavior with a vertical rectangle.

The sign of the misfit of the alloy is also indicated. Note that in three cases (a, c, f), two different signs of the misfit are given. For these cases, the first sign corresponds to the value given by the authors in the referenced paper, while the second sign corresponds to the actual sign of the misfit for the alloy at test temperature. In particular, for the three cases:

(a) Tien and Copley [12, 13] state the value of the misfit of their alloy, Udimet-700, as +0.02%, as measured by Oblack and Kear [16] at room temperature. This value should be therefore corrected to obtain the value of the misfit at test temperature.

 In several studies [23, 27, 28], the expansion coefficients of the γ' phase have been found to be lower than those of the γ phase; this is consistent with the long-range ordered structure of γ' [29]. According to the data of Grose and Ansell [28], we can infer that the value of the misfit of Udimet-700 at the test temperature of 954°C is of the order of -0.3%.

(c) The value of the misfit given by Caron and Khan in [21] (+0.14%) has also been measured at room temperature. Fredholm and Strudel have subsequently determined that the actual value of the misfit at test temperature is -0.33% [23].

(f) In [17] Carry and Strudel give a negative value for the misfit (- 0.4%). This value has been corrected in a subsequent study [23] where the actual value of the misfit has been found to be + 0.38%. The erroneous value reported in [17] was probably due to an incorrect Burgers' vector sign convention.

Since the sign of the misfit at test temperature plays a fundamental role in the rafting behavior of the alloys, these misleading indications in the literature have brought about a substantial confusion both in the interpretation of the experimental results and in the modeling of the rafting phenomenon.

Researchers	References	Year	Misfit sign in paper/actual	Tension ↑□↓	Compression ↓□↑
(a) Tien and Copley	12, 13	1971	+ / -	▭	▯
(b) Miyazaki, Nakamura, Mori	15	1979	+	▯	▭
(c) Carry and Strudel	16, 17	1979	- / +	▯	
(d) Pearson, Lemkey, Kear	18, 19	1980	-	▭	▯
(e) Nathal and Ebert	20	1983	-	▭	▯
(f) Caron and Khan	21	1983	+ / -	▭	
(g) Mackay and Ebert	22	1984	-	▭	
(h) Fredholm and Strudel	23	1984	-	▭	
(i) Mackay and Ebert	24	1985	-	▭	
(j) Pollock	25	1989	-	▭	
(k) Pollock	25	1989	+	▯	

Table 1.1 Rafting Observations

1.3 Energy Approaches

The rafting phenomenon is an interesting example of how an externally applied loading condition can influence the morphology evolution of the microstructure in a multi-phase material.

The most puzzling point in this phenomenon is the marked directionality of the coarsening process in presence of an applied stress. We can ask ourselves two basic questions:

> What are the parameters that affect the direction of rafting?
>
> How can we predict, knowing the value of these parameters, the rafting behavior of the material?

The evolution of precipitate morphology can be qualitatively understood by using an energy analysis. Rafting is a spontaneous evolution of the microstructure; thus we can conclude that it must be associated with a decrease in the energy level of the system.

If we consider the original cuboidal morphology and the final rafted configuration, the variation in energy, ΔE^T, associated with this variation in the microstructure, can be roughly broken down into three terms:

$$\Delta E^T = \Delta E_{chem} + \Delta E_{int} + \Delta E^T_{EL}. \tag{1.2}$$

Here, the first two terms, ΔE_{chem} and ΔE_{int}, account for, respectively, the variation in chemical energy and interfacial energy. They are most certainly important terms that play a fundamental role in the coarsening process, but they "cannot discern among the three < 100 > crystal directions". In other words, the value of these two terms will be the same for all the rafted configurations shown in Fig. 1.3: they are, giving to the word a broader meaning, *isotropic* terms and cannot account for the directionality of coarsening under stress annealing. Thus the key to the problem must lie in the third term, ΔE^T_{EL}: the variation of the total elastic energy of the system.

Since this is the *only* term which is sensitive to the direction of the applied stress, we can infer that the directionality of rafting is controlled by a tendency to decrease the total elastic energy E^T_{EL} which is given by the sum of the elastic energy of the crystal plus the potential energy of the loading system.

If we express the total energy of the crystal as a function of the precipitate morphology, the misfit, the applied stress, σ_∞, and all the other parameters that characterize the system:

$$E^T = \hat{E}^T(\gamma'\_\text{morphology}, \delta, \sigma_\infty \ldots), \tag{1.3}$$

then we can think of two possible approaches to predict the morphology evolution of the precipitates (Fig. 1.5).

A first approach can be termed an *energy minimization* approach: we calculate the finite energy levels for different morphologies of the precipitates relative to some reference state such as a homogeneous Ni-Al solid solution crystal. We then compare the energy levels of the different morphologies under a certain loading condition and infer that the system will evolve toward the morphology that corresponds to the lowest value of the total energy.

An alternative approach can be termed an *energy perturbation* approach: we consider the actual initial configuration with γ' cuboids, and investigate the effect of a perturbation of the morphology of the precipitates on the total energy. We can thus determine the driving force:

$$f = \frac{-\delta E^T}{\delta\gamma'\_\mathrm{morphology}} \tag{1.4}$$

which is acting on the system, and infer the most likely direction for morphology evolution.

Note that, in both approaches, we implicitly assume that there exists a mechanism, characterized by suitable kinetics, to accomplish the morphology evolution.

If we compare the two approaches on a schematic "total energy vs. γ'-morphology" graph (Fig. 1.5), the first approach corresponds to the determination of the minimum of the curve, while the second approach corresponds to the determination of the slope of the curve for the initial morphology.

The first approach seems to be more suitable to study displacive transformations, where the microstructure instantaneously switches to the lowest energy configuration, while the second approach appears to be more easily incorporated in a kinetic model to study diffusive transformations, where the morphology evolution is controlled by the instantaneous value of the driving force as well as by the kinetics of the diffusion process.

Historically, in the study of rafting, the energy minimization approach has been more widely used, as we will briefly discuss in the next paragraph.

In our study we will instead follow an energy perturbation approach.

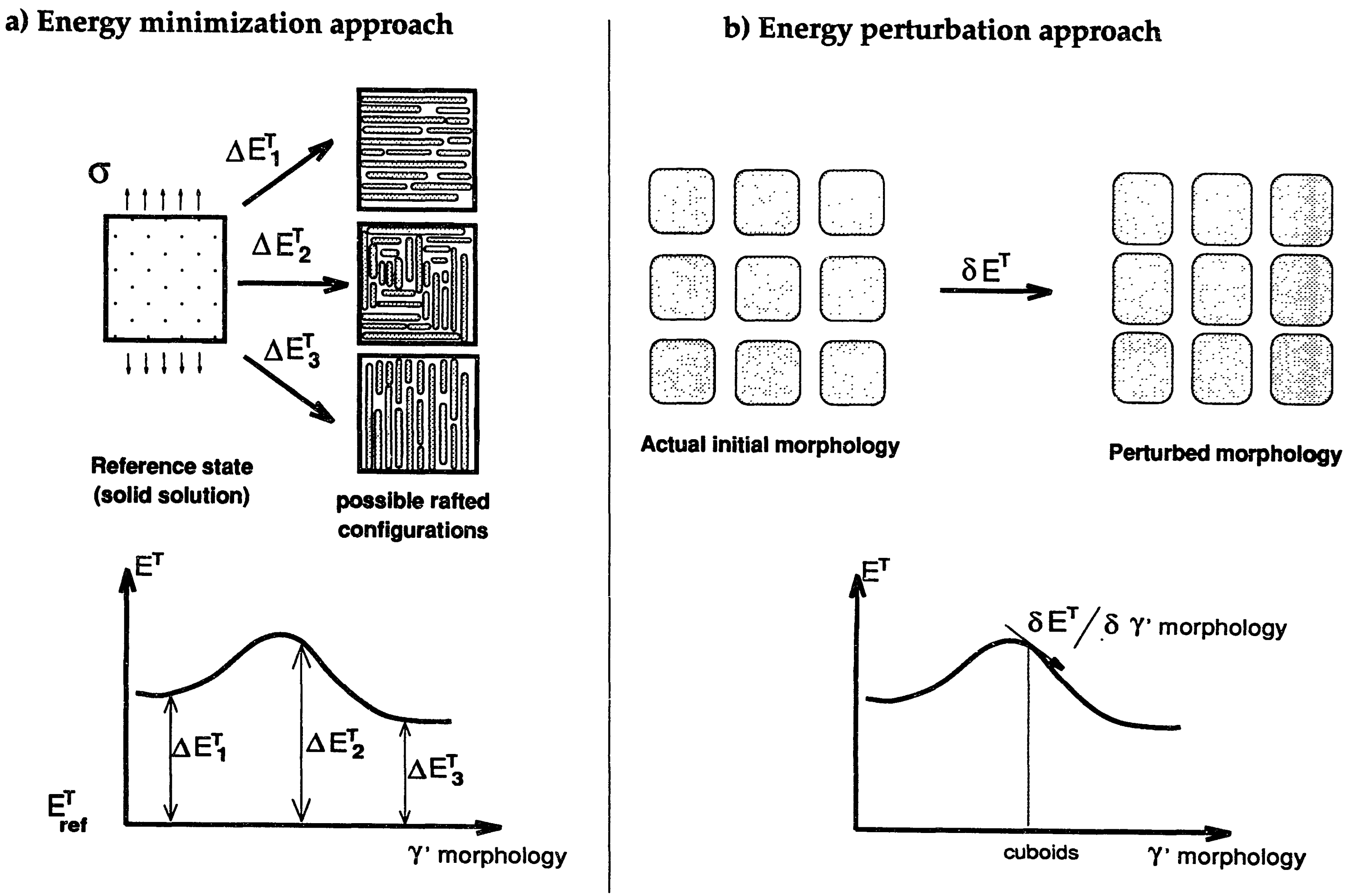


Figure 1.5 Energy approaches .

1.4 Historical Review of Rafting Models

The first attempt to explain the rafting behavior through energy considerations is due to Tien and Copley [13]. They observed the structure of the γ' precipitates in Udimet-700 and found that the sense and crystallographic orientation of the external stress influenced the final rafted morphology of the γ' precipitates. Their observations are graphically summarized in Fig. 1.6. The dotted cubes show the initial orientation of the γ' cuboids with respect to the direction of stress (vertical in the figure). The shapes bounded by solid lines represent the aligned plates, parallelepipeds and cuboids that result from stress annealing.

In the discussion of their observation, Tien and Copley gave only a qualitative theoretical analysis and could not draw quantitative conclusions regarding the influence of elastic energy on the final particle shape.

A significant step in the understanding of the rafting phenomenon is due to Pineau [14]. Since the evaluation of the total energy for the actual $\gamma - \gamma'$ microstructure is extremely arduous, the calculations are drastically simplified by considering a parallel model problem in which a single, ellipsoidal inclusion, representing the γ' phase, is embedded in an infinite γ-matrix. The fundamental idea behind this approach is that there would exist a direct correspondence between the actual problem and the model problem so that the results of the model problem could be directly extrapolated to the real microstructure (Fig. 1.7). In other words, if for the model problem an ellipsoid prolate in the direction of the applied stress minimizes the energy, then a "type P" behavior is inferred for the actual microstructure, while if an oblate ellipsoid minimizes the energy, a "Type N" behavior is inferred.

Pineau systematically applies Eshelby's theory and calculates the energy of a solid containing misfitting coherent precipitates of various shapes subjected to applied stresses. Three shapes are considered: spheres, plates perpendicular to the stress axis and needles aligned with the stress axis. Matrix and precipitates are assumed to be elastically isotropic.

Based on these calculations, the most stable shapes (corresponding to the lowest elastic energy), have been determined, and the final results are presented in the graphical form shown in Fig. 1.8.

According to this map, the major factors which affect stress coarsening behavior are the direction and the value of the applied stress, normalized by the elastic modulus of the matrix, the ratio between the elastic moduli of the two phases, and the γ - γ' misfit. The derivations of Pineau have been generally accepted, for a certain time, as the most satisfactory treatment of the rafting process.

Unfortunately, as noted by Fredholm and Strudel [23], the model considered by Pineau leads to predictions that agree with the experimental observations summarized in Table 1.1, only if it is assumed that the γ' particles are stiffer than the matrix for all the tested alloys. But for at least two cases [13, 15], we have positive evidence

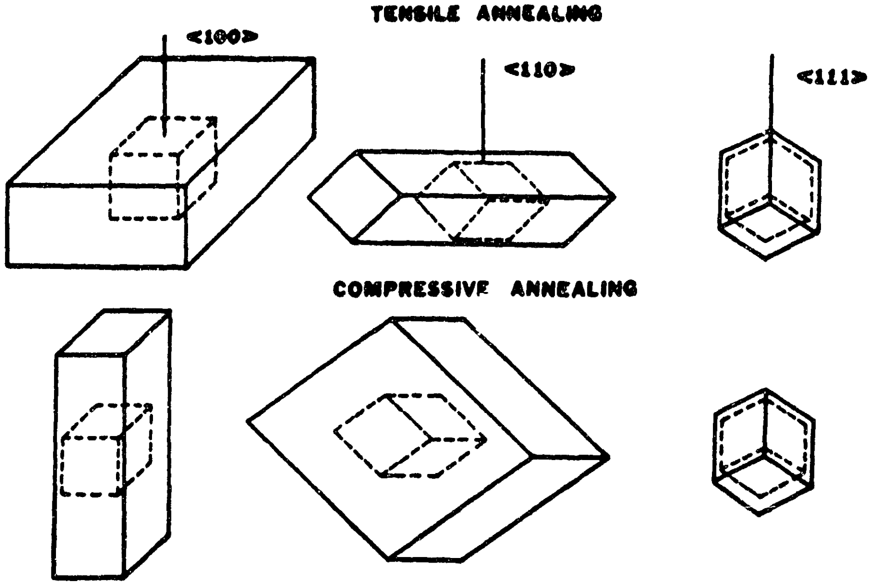


Figure. 1.6 A schematic summarizing the effects of stress orientation on the stress annealed shape of the γ' precipitates [13].

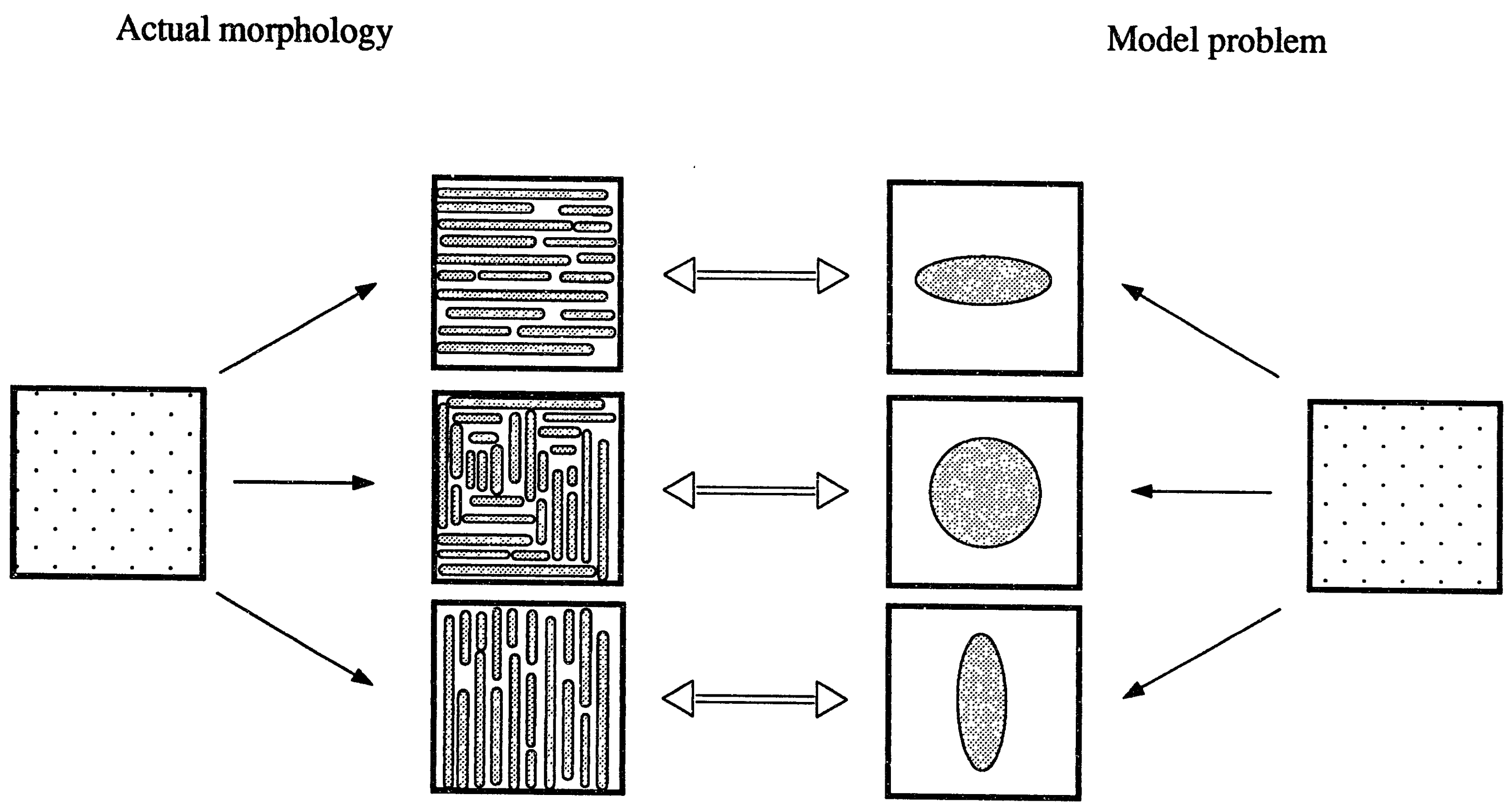


Figure 1.7 Correspondence between the model problem and the actual morphology.

that the precipitate phase is actually softer than the matrix so that Pineau's map predictions do not agree with the experimental data.

In order to obtain results more consistent with the experimental observations, subsequent studies have tried to reduce the number of simplifications in the Pineau model.

The Eshelby equivalent inclusion approach can be successfully applied to the study of ellipsoidal inclusions in anisotropic media, and models based on anisotropic elasticity have been derived in a number of studies [15, 31, 35].

The complexity of the problem increases by an order of magnitude when a cuboidal shape for the precipitate is considered. Faivre [33] has treated the problem of a single cuboidal inclusion for isotropic media and Chang [35] introduced a method based upon Green's function techniques to study single cuboidal precipitates in anisotropic media.

Regarding the actual periodic structure of the γ - γ' morphology, and the effect of interaction energy between particles, Johnson [34] has considered the problem of two spherical inhomogeneities under the influence of an applied stress and studied the coarsening kinetics of the two particles; Jankowski, Wingo and Tsakalakos [32] have modeled the periodicity of the microstructure by using a space and time-dependent Fourier series; finally, Chang [35] has attempted, with partial success, to apply the Mura trigonometric series method and the finite element method to treat arrays of inclusions.

Still, no comprehensive model is available that accounts for the actual structure of the crystal and gives predictions which explain all the experimental results.

The reasons for this failure lie not only in the inherent limitations of the Eshelby inclusion approach: even a perfect elastic model of the real microstructure would not yield correct predictions for the rafting behavior.

The essential common flaw for all the model proposed in the literature is that the *inelastic* response of the γ-matrix, due to dislocation motion, is always *neglected* in the evaluation of the stress and strain fields. As mentioned in paragraph 1.2, a network of misfit dislocations is always observed at the γ - γ' interfaces during rafting. The presence of these dislocations suggest that a localized "creep process" is relieving the γ - γ' misfit, altering the state of stress in the crystal. As will be shown in Chapter 4, only a model that takes into account this effect can successfully account for the rafting behavior of Ni-superalloys.

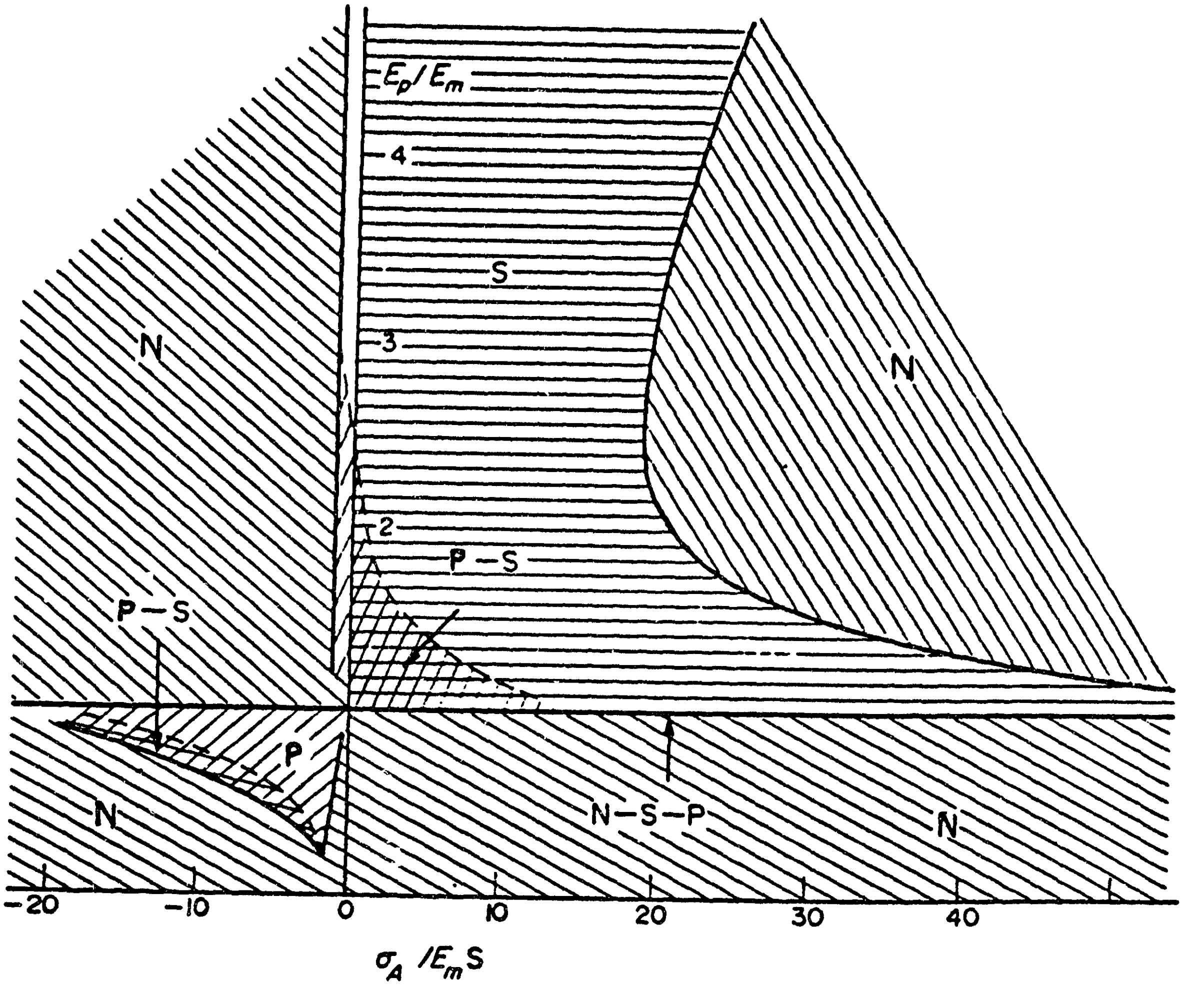


Figure. 1.8 Map giving the conditions which lead to the lowest total elastic energy for spheres (S), plates normal to the stress axis (N) and needles parallel to the stress axis (P). E_p/E_m is the ratio of the Young's modulus of the precipitate and that of the matrix, σ_A is the applied stress and δ the misfit between the precipitates and the matrix. The domains where two or three shapes are indicated for the particles are those where the difference in the corresponding elastic energies is less than 0.1 $E_m\delta^2$. [11]

CHAPTER 2
THE FORCE ON A MATERIAL INTERFACE

2.1 Introduction

Consider a generic system whose configuration can be identified by a suitable number of parameters $\beta_1, \beta_2 \ldots \beta_n$. We can express the *total energy* of the system, E^T, as a function of these parameters:

$$E^T = \hat{E}^T(\beta_1, \beta_2, \ldots \beta_i, \ldots \beta_n). \tag{2.1}$$

Following the terminology of analytical mechanics and thermodynamics, we can introduce the notion of *generalized force conjugate to the i^{th} parameter, f_i*, defined as

$$f_i = -\frac{\partial E^T}{\partial \beta_i}. \tag{2.2}$$

Thus f_i, which we may regard as the force *acting* on β_i, is the rate of decrease of the total energy with respect to the parameter β_i.

It is important to note that, in this definition, E^T is the *total* energy, i.e., the energy of the system we are studying plus the energy of the environment with which it interacts.

We will now restrict our attention to solid bodies with prescribed boundary conditions.

We can expect, in the material of the body, departures from uniformity on various scales which we may call *imperfections.* Examples on a microscopic scale are dislocations, foreign atoms, vacant lattice points and grain boundaries. On a macroscopic scale, there might be inclusions of one phase in another, cavities and cracks.

If the boundary conditions are held constant, the total energy of the system (the energy of the body plus the potential energy of the loading device) is a function of the set of parameters necessary to specify the configuration of the imperfections.

Thus, following the general derivation, we can define the force on a gliding dislocation, on an extending crack, or on a growing cavity.

For the particular case of inclusions and precipitates of one phase in another, such as martensitic plates in ferrite or γ' precipitates in a γ matrix in Ni-superalloys, if the two phases are uniform within themselves, we can consider the *interface itself to be the imperfection.*

If $\mathcal{C}^*$ is a reference configuration for the interface, we can characterize any other configuration, $\mathcal{C}$, in terms of the normal displacement of the interface, ξ_{n}, with respect to $\mathcal{C}^*$ (Fig. 2.1).

Thus the energy level E^T is determined by the parameter, $\xi_{\mathbf{n}}$, which is a function of the position $\vec{x}$ along the interface:

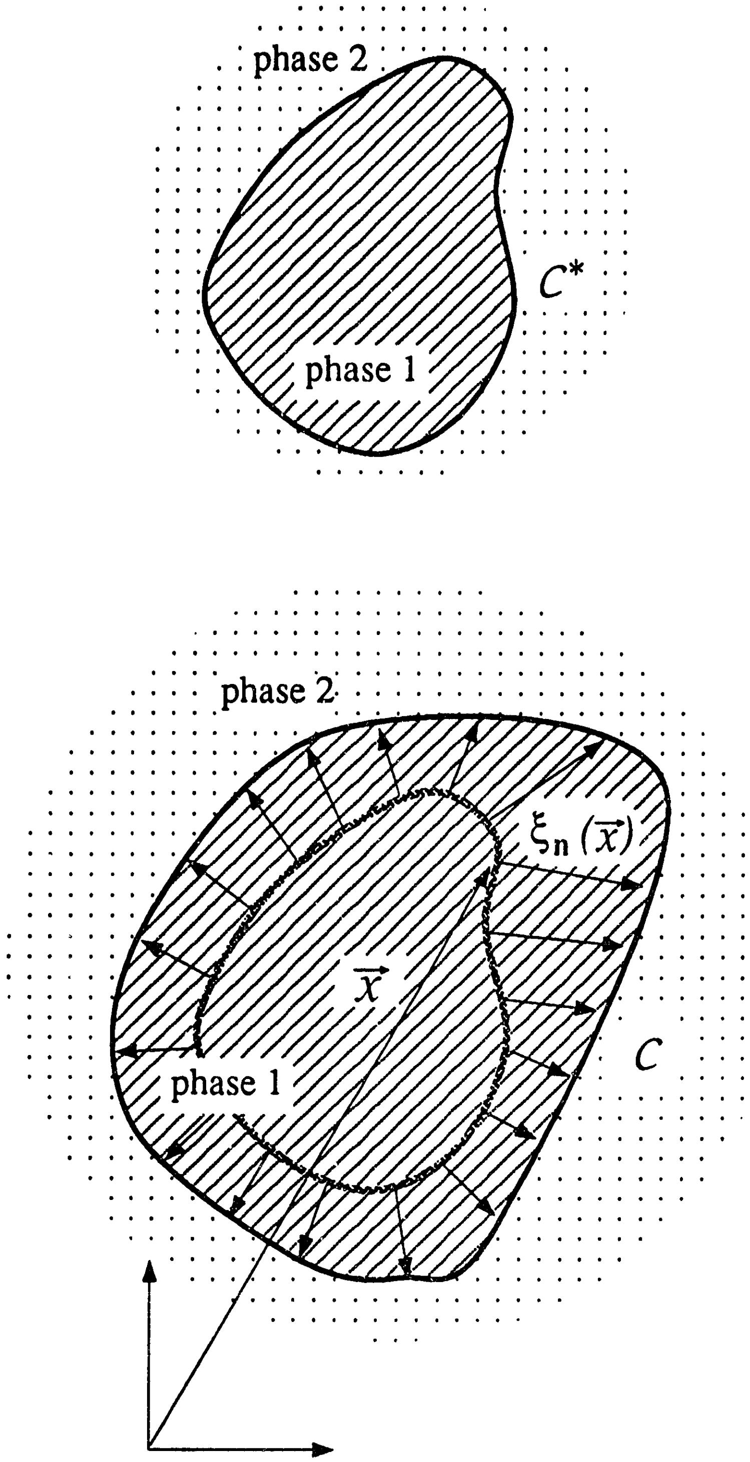
$$E^T = \hat{E}^T(\xi_{\mathbf{n}}(\vec{x})) \tag{2.3}$$

It is then straightforward to define a normal force acting on the interface, $\tau_{\mathbf{n}}$, which is work-conjugate to $\xi_{\mathbf{n}}$, as:

$$\tau_{\mathbf{n}}(\vec{x}) = \frac{-\partial E^T}{\partial \xi_{\mathbf{n}}(\vec{x})}. \tag{2.4}$$

It is possible to express this generalized force, $\tau_{\mathbf{n}}$, in an extremely convenient form using a quantity whose interesting features were first recognized by Eshelby [36], and which is therefore known as *Eshelby's Energy Momentum Tensor* (EMT).

In the following paragraphs, which closely follow the derivation presented by Eshelby in [37], we will first describe the mathematical process which generates the EMT, then we will discuss some physical interpretations and derive an expression for the force on a material interface.



Reference configuration C^*
$E^T = E^T_0 = E^T(\xi_n = 0)$

Current configuration C
$E^T = \hat{E}^T(\xi_n(\vec{x}))$

Figure 2.1 Interface configuration.

2.2 The Energy Momentum Tensor in Continuum Mechanics

We will derive an expression for the EMT in the framework of finite deformation theory with a hyperelastic stress-strain relation.

Rectangular Cartesian coordinates X_i are used to label the position of material particles in the initial state.

If u_i is the displacement field, the final position of the particle, in the same coordinate system, will be x_i so that

$$u_i = x_i - X_i. \tag{2.5}$$

As a stress measure, we use the nominal Piola Kirchhoff stress, p_{ij}, so that, if W is the Lagrangian strain energy density,

$$p_{ij} = \frac{\partial W}{\partial u_{i,j}}. \tag{2.6}$$

The equilibrium condition is

$$\frac{\partial p_{ij}}{\partial X_j} + \rho b_i = 0_i, \tag{2.7}$$

where b_i are the body forces per unit mass (which will include the D'Alembert inertial force in dynamic problems), and ρ is the mass density referred to the initial volume.

We will consider the general case in which the strain energy density, W, depends on the gradient of the displacement field $u_{i,j}$ and also *explicitly* on the X_m:

$$W = \hat{W}(u_{i,j}; X_m). \tag{2.8}$$

This explicit dependency on X_m allows us to consider material inhomogeneities (regions with varying elastic constants) as well as states of internal stress characterized by the presence of *eigenstrains* $\epsilon_{ij}^T(X_m)$.

Here, with the term eigenstrains we indicate all such nonelastic strains as those associated with thermal expansion, phase transformation, creep, plastic flow, and lattice misfit [38].

For the elastic calculations of the force on a material interface, we will consider these strain fields as "frozen" in the material, i.e., the eigenstrain field will depend *only* on the location X_m.

Under these conditions it is perfectly equivalent to regard ϵ_{ij}^T as an extra field variable or to absorb the ϵ_{ij}^T-dependence of W into its explicit dependence on X_m.

If we substitute Eq. (2.6) in (2.7), we obtain an alternative form of the equilibrium equations

$$\frac{\partial}{\partial X_j}\frac{\partial W}{\partial u_{i,j}} = -\rho b_i. \tag{2.9}$$

At this point we need to distinguish the gradient of $W, \partial W/\partial X_i$, defined by

$$W(\vec{X} + d\vec{X}) = W(\vec{X}) + \frac{\partial W}{\partial X_i} dX_i + o(dX_i)^2, \tag{2.10}$$

from the explicit partial derivative of W with respect to X_i,

$$(\frac{\partial W}{\partial X_i})_{\text{exp}} = \frac{\partial W(u_{i,j}; X_m)}{\partial X_i}\Bigg|\begin{array}{l} u_{i,j} = \text{const.} \\ X_m = \text{const.}\ m \neq i. \end{array} \tag{2.11}$$

The gradient of W is thus given by

$$\frac{\partial W}{\partial X_k} = \frac{\partial W}{\partial u_{i,j}}\frac{\partial u_{i,j}}{\partial X_k} + (\frac{\partial W}{\partial X_k})_{\text{exp}}. \tag{2.12}$$

Noting that $u_{i,jk} = u_{i,kj}$ and using the rule for differentiating a product, we obtain

$$\frac{\partial W}{\partial X_k} = \frac{\partial}{\partial X_j}(\frac{\partial W}{\partial u_{i,j}} u_{i,k}) - \frac{\partial}{\partial X_j}(\frac{\partial W}{\partial u_{i,j}}) u_{i,k} + (\frac{\partial W}{\partial X_k})_{\text{exp}}. \tag{2.13}$$

Substituting (2.9) and (2.6) in (2.13), we have

$$\frac{\partial}{\partial X_j}(W\delta_{jk} - p_{ij}u_{i,k}) = (\frac{\partial W}{\partial X_k})_{\text{exp}} + \rho b_i u_{i,k}. \tag{2.14}$$

We can now introduce the Energy Momentum Tensor, P, whose components are given by

$$P_{kj} = W\delta_{kj} - p_{ij}u_{i,k}, \tag{2.15}$$

so that eq. (2.14) reads

$$\frac{\partial P_{kj}}{\partial X_j} = (\frac{\partial W}{\partial X_k})_{\text{exp}} + \rho b_i u_{i,k}. \tag{2.16}$$

Thus the divergence of P vanishes wherever W does not explicitly depend on X_m and there are no body forces.

Note how, for a more concise notation, here we have incorporated the ϵ^T_{ij} -dependence of W into its dependence on X_m; if we regard ϵ^T_{ij} as an extra field variable, then

$$W = \hat{W}(u_{i,j}; \epsilon^T_{ij}(X_m); X_m). \tag{2.17}$$

We can expand $(\frac{\partial W}{\partial X_k})_{\text{exp}}$ into two contributions:

$$(\frac{\partial W}{\partial X_k})_{\text{exp}} = (\frac{\partial W}{\partial X_k})_{inhom} + \frac{\partial W}{\partial \epsilon^T_{mn}}\frac{\partial \epsilon^T_{mn}}{\partial X_k}, \tag{2.18}$$

where the first term accounts for material inhomogeneities as well as, in a more general context, for any effect due to space gradients of elastic constants (e.g., temperature dependence of elastic constants associated with a temperature gradient).

An interesting interpretation of the physical meaning of P can be derived if we consider an elastic body subject only to surface loading (with no body forces), containing a certain number of imperfections $I_1, I_2, \ldots I_n$. These imperfections might be inhomogeneities, inclusions with an eigenstrain, point defects, etc. We assume that, apart from these imperfections, the remaining material is "good" elastic material where

$$(\frac{\partial W}{\partial X_k})_{exp} = 0_k. \tag{2.19}$$

If the generic defect I suffers a small translation $\delta\vec{\xi}$, there will be a variation in the total energy of the system

$$\delta E^T = -\delta\xi_k f_k, \tag{2.20}$$

where $\vec{f}$ is the generalized force acting on I.

It can be proved [37] that the components of $\vec{f}$ are given by:

$$f_k = \int_S P_{kj} dS_j, \tag{2.21}$$

where S is any surface surrounding I and isolating it from all the other imperfections, and $d\vec{S} = dS\vec{n}$ is the oriented surface element with $\vec{n}$ being the unit *outward* normal to the surface. Note that S can be chosen arbitrarily, without altering $\vec{f}$, only as long as it remains in "good" material where (2.19) holds.

2.3 An Expression for the Force on an Interface

We are now concerned with the special case where the imperfection is represented by the interface between an *inclusion*, Ω, and an otherwise homogeneous medium which we will call the *matrix*, D (Fig. 2.2). We limit our attention to coherent interfaces, for which the displacement field $\vec{u}$ is continuous at the interface. The inclusion may have elastic constants differing from those of the matrix, and the body may be subjected to an eigenstrain field, ϵ_{ij}^T, which might be discontinuous across the interface.

We will derive an expression for the generalized normal force acting on the interface as a result both of the internal state of stress, due to the ϵ_{ij}^T-field, and of the externally applied loads. This quantity will be a measure of the force driving the migration of the interface.

Consider a reference configuration for the interface, S^*, and an infinitesimal migration to a current configuration S (Fig. 2.2). The migration can be specified by a small vector $\delta\vec{\xi}$ at each point of S^*.

We want to evaluate the change in the total energy of the system, δE^T, as a result of the migration $S^* \longrightarrow S$. We can obtain our result with the help of a sequence of imaginary steps which simulate the migration.

We start with the interface in the S^* configuration, a displacement field $\vec{u}^*$ in the body ($\vec{u}^{*D}$ in the matrix, $\vec{u}^{*\Omega}$ in the inclusion), and we want to end up with the interface in the S configuration and a displacement field $\vec{u}$ in the body ($\vec{u}^D$ in the matrix, $\vec{u}^\Omega$ in the inclusion).

Step 1. We cut out and remove the "D-material" which lies in the region between S^* and S, and apply suitable surface tractions to the "hole" to prevent relaxation.

The variation in energy for step 1 is

$$\delta E^T_{(1)} = -\int \delta\xi_j W^D dS_j, \tag{2.22}$$

where $d\vec{S} = dS\vec{n}$, with $\vec{n}$ being the unit outward normal to the interface (from Ω to D: see Fig. 2.2), and W^D is the elastic energy of the "D-material" that we are removing.

The displacement on the boundary, S, of D is now $u_i^{*D} + \delta\xi_k u_{i,k}^{*D}$ and the traction is $p_{ij}^D(-dS_j) + o(d\xi_j)$.

Step 2. We bring the displacement field at the boundary of D to its final value $\vec{u}^D$. The variation in energy is:

$$\delta E^T_{(2)} = -\int [u_i^D - (u_i^{*D} + \delta\xi_k u_{i,k}^{*D})] p_{ij}^D dS_j. \tag{2.23}$$

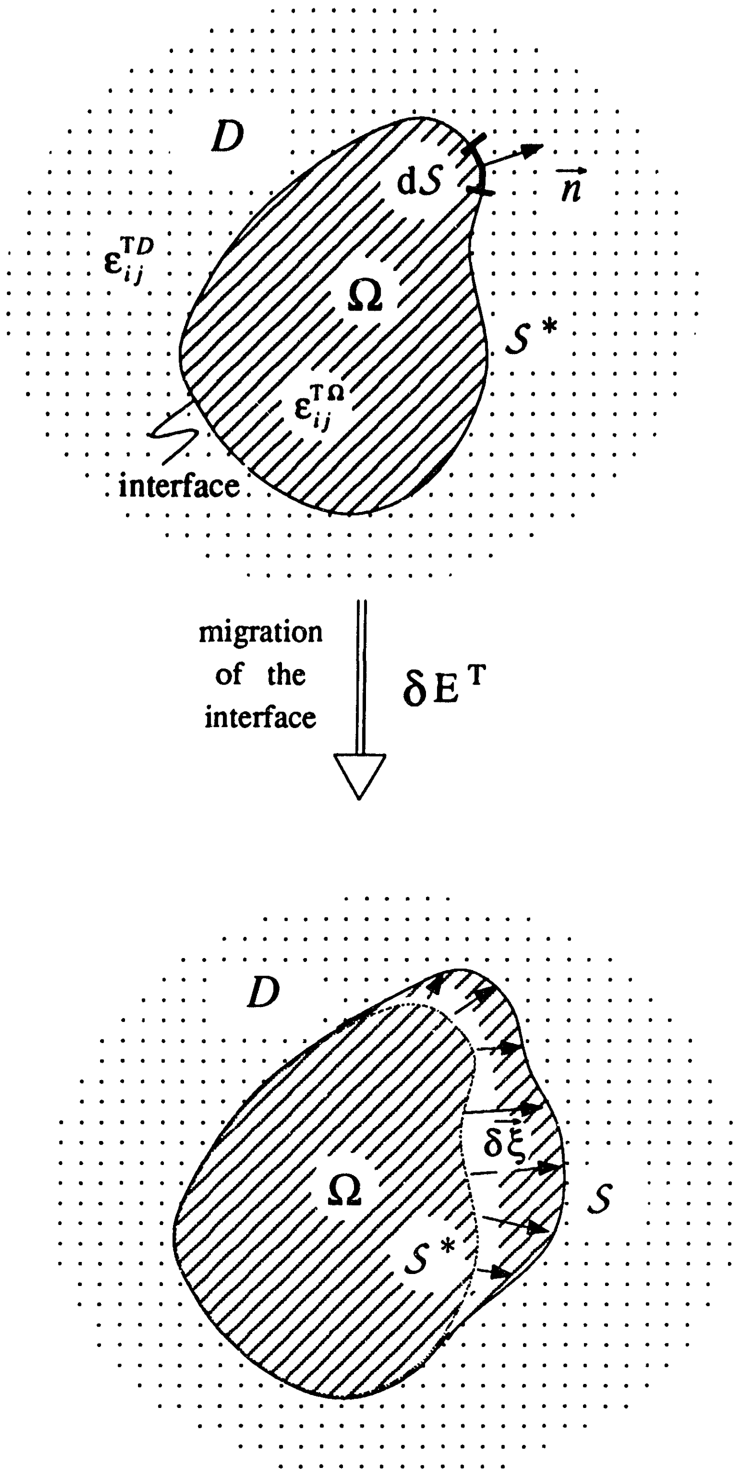


Figure 2.2 Variation in energy associated with a migration of the interface .

Step 3. We allow the material that we have removed to transform from "D-material" to "Ω-material" and we put it back in place.

The variation in energy is then

$$\delta E^T_{(3)} = \int \delta\xi_j W^\Omega dS_j. \tag{2.24}$$

The displacement on the boundary S of Ω is now $u_i^{*\Omega} + \delta\xi_k u^{*\Omega}_{i,k}$ and the traction is $p^\Omega_{ij} dS_j + o(d\xi_j)$.

Step 4. We alter the displacement of the boundary of Ω to its final value, $\vec{u}^\Omega$, with a variation in energy:

$$\delta E^T_{(4)} = \int [u_i^\Omega - (u_i^{*\Omega} + \delta\xi_k u^{*\Omega}_{i,k}) p^\Omega_{ij} dS_j]. \tag{2.25}$$

Note that if the transformation from "D-material" to "Ω-material" brings about a variation in chemical energy, we should also include a term

$$\delta E^T_{chem} = \int \delta\xi_j (W^\Omega_o - W^D_o) dS_j, \tag{2.26}$$

where $(W^\Omega_o - W^D_o)$ is the work required to transform a unit volume of unstressed D into an equal mass of unstressed Ω.

Furthermore, if a surface energy γ_{int} is associated with the interface, the displacement of the boundary brings about a variation in energy:

$$\delta E^T_{int} = \int \delta\xi_j \gamma_{int} K dS_j, \tag{2.27}$$

with $K = 1/R_1 + 1/R_2$, where R_1, R_2 are the local principal radii of curvature of the surface.

As previously mentioned (paragraph 1.3), in order to give an explanation for the directionality of rafting we can actually restrict our attention to the variation of the total elastic energy $\delta E^T_{EL} (= \delta E^T_{(1)} + \delta E^T_{(2)} + \delta E^T_{(3)} + \delta E^T_{(4)})$ so that we will neglect the chemical and interface energy terms in the following derivation. Thus, the expression for the force acting on the interface $\tau_{\mathbf{n}}$, that we will derive in the following section, actually gives only the "elastic contribution" to the force on the interface.

Chemical and interface energy terms should, however, be included in more general applications.

Since $\vec{u}$ is continuous across the interface, we have $u_i^D = u_i^\Omega$ and $u_i^{*D} = u_i^{*\Omega}$ for each element dS_j. Also, the traction vector $p_{ij} dS_j$ is continuous at the interface, so that on adding the four contributions (2.22, 2.23, 2.24, 2.25), we obtain the total variation:

$$\delta E^T_{EL} = -\int_{S^*} \delta\xi_k \{(W^D \delta_{kj} - p_{ij} u^D_{i,k}) - (W^\Omega \delta_{kj} - p_{ij} u^\Omega_{i,k})\} dS_j, \tag{2.28}$$

and, using the definition (2.15),

$$\delta E^T_{EL} = -\int_{S^*} \delta\xi_k \{P^D_{kj} - P^\Omega_{kj}\} dS_j. \tag{2.29}$$

It is natural to choose the direction of $\delta\vec{\xi}$ normal to the interface: $\delta\vec{\xi} = \delta\xi_{\mathbf{n}}\vec{n}$. If we use the notation $[A]$ to denote the discontinuity of the generic quantity A across the interface, we can recast Eq. (2.29) in the form

$$\delta E^T_{EL} = -\int_S \delta\xi_{\mathbf{n}}(n_k[P_{kj}]n_j)dS. \tag{2.30}$$

If we compare Eq. (2.30) and Eq. (2.4) we can easily see that Eq. (2.30) is equivalent to the statement that there is an effective normal force per unit area of the interface of magnitude

$$\tau_{\mathbf{n}} = n_k[P_{kj}]n_j, \tag{2.31(a)}$$

or, if we substitute expression (2.15) for P,

$$\tau_{\mathbf{n}} = [W] - t_i[\frac{\partial u_i}{\partial n}], \tag{2.31(b)}$$

where $t_i = p_{ij}n_j$ are the components of the traction vector $\vec{t}$ at the interface and $\partial/\partial n$ denotes spatial differentiation along the normal direction.

Note how, throughout our derivation of $\tau_{\mathbf{n}}$, we have never needed to invoke condition (2.19). This means that expressions (2.31) for $\tau_{\mathbf{n}}$ hold also in the presence of eigenstrains ϵ^T_{ij} and body forces b_i. In fact, we will actually use these expressions to evaluate the elastic driving force for morphology evolution of γ' precipitates in Ni-superalloys.

We will first perform our calculations for merely elastic fields in which the only eigenstrain is due to the $\gamma - \gamma'$ lattice misfit, then we will follow the evolution of the force on the interface during a stress-annealing transient where creep strains set in, so that the eigenstrain field will be the superposition of the initial misfit field and of the creep field.

Before concluding this paragraph, it is perhaps appropriate to emphasize that $\tau_{\mathbf{n}}$ is an extremely convenient measure of the elastic driving force for the evolution of the inclusion shape.

A positive value of $\tau_{\mathbf{n}}$ (Fig. 2.3) means that the total elastic energy will be reduced if the interface migrates outward, thus indicating a tendency for the inclusion to grow. Conversely, a negative value of $\tau_{\mathbf{n}}$ indicates a tendency for the interface to reduce its volume.

We can construct graphs of the distribution of $\tau_{\mathbf{n}}$ over the interface; in particular, if we limit ourselves to 2D-problems, we can plot the value of $\tau_{\mathbf{n}}$ as a function of the arclength along the interface (Fig. 2.4).

We will introduce numerical methods to calculate $\tau_{\mathbf{n}}$ in Chapter 3, and in Chapter 4 we will apply these methods to the analysis of rafting in $\gamma - \gamma'$ superalloys.

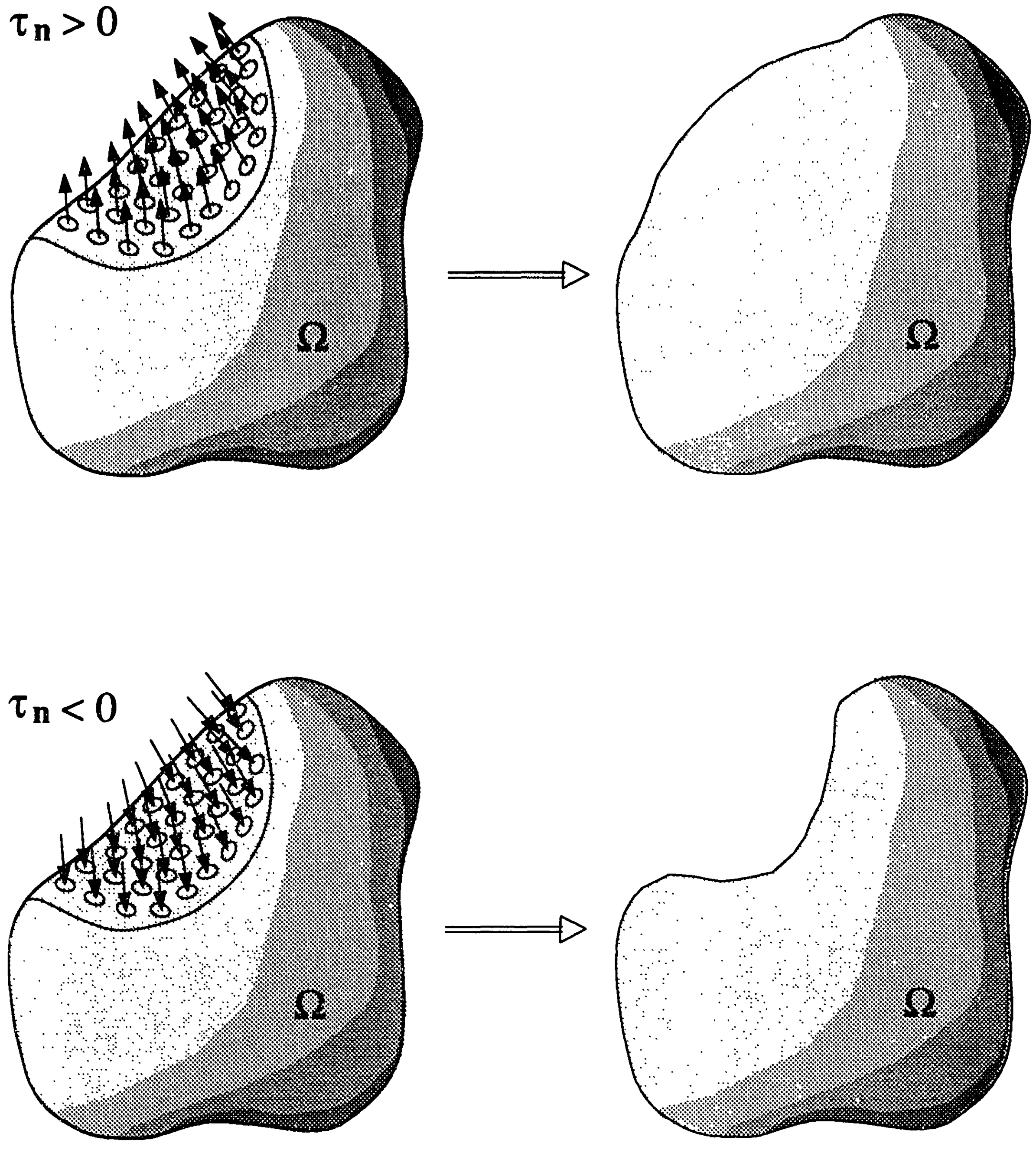


Figure 2.3 τ_n as a driving force for shape evolution.

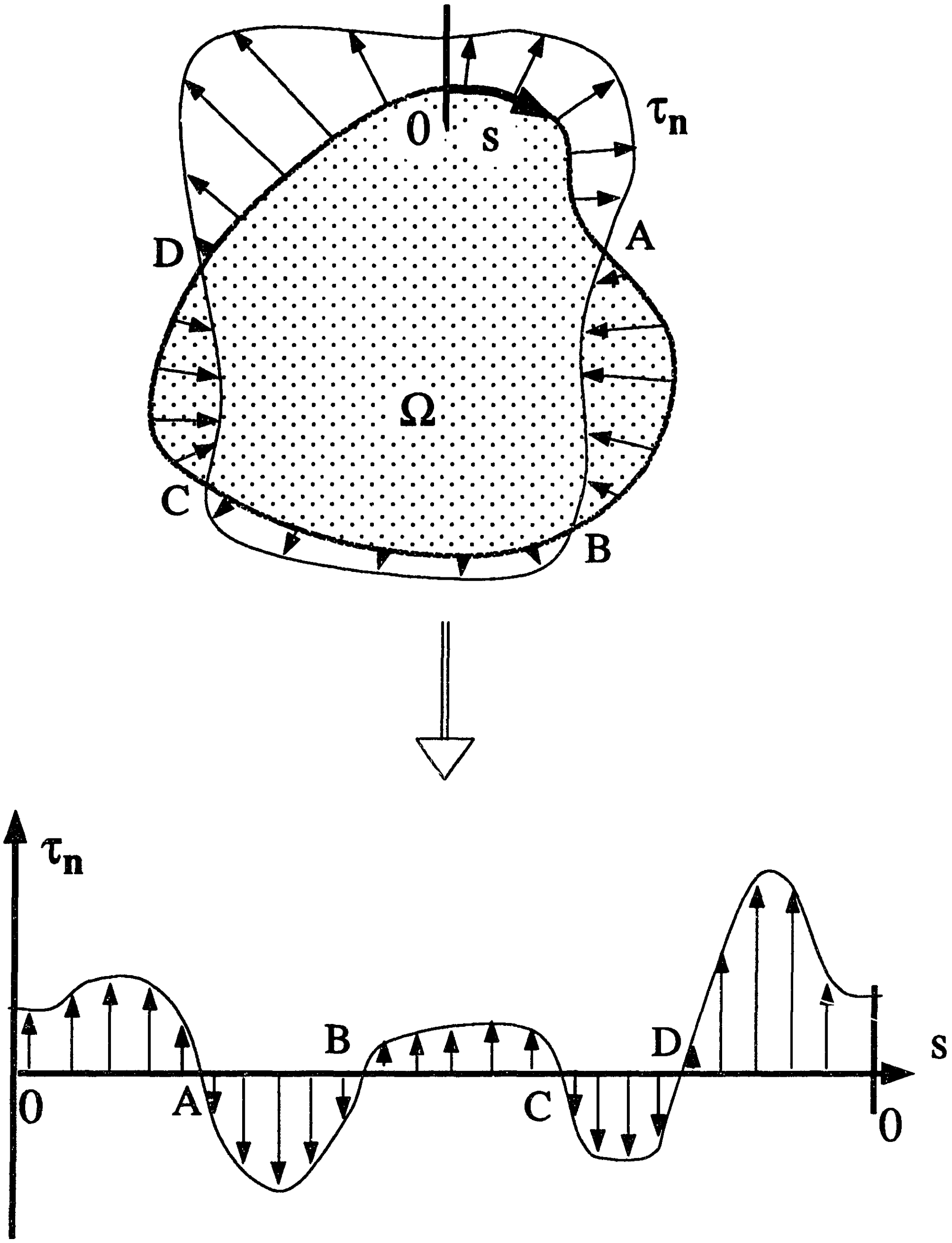


Figure 2.4 Schematic plot of the force along the interface for 2-D models .

CHAPTER 3
NUMERICAL METHODS

3.1 Introduction

In Chapter 2 we have defined a quantity, $\tau_{\mathbf{n}}$, which provides direct information on the driving force for morphology evolution of multi-phase materials. If we examine expressions (2.31), we can notice how the force on the interface can be obtained directly in terms of elastic field quantities evaluated at the material interface. This is indeed an extremely convenient feature of this approach, since we have an instrument, the Finite Element Method (FEM), that allows us to evaluate the elastic field around a material interface, for virtually any kind of morphology of the system, material constants, applied loads and internal stress state (eigenstrain field).

Throughout this thesis, the finite element program ABAQUS [39] has been used, so that, even if the methods are general, the software has been specifically developed to postprocess ABAQUS results.

For the present applications, we can limit our attention to small-deformation analysis, so that we can substitute the nominal Piola Kirchhoff stress, p_{ij}, with the Cauchy stress σ_{ij} and the expression for the EMT becomes

$$P_{kj} = W\delta_{kj} - \sigma_{ij}u_{i,k}. \tag{3.1}$$

The elastic strain energy density will be given by

$$W = \hat{W}(\epsilon^e_{ij}) = \int_0^{\epsilon^e} \sigma_{ij} d\epsilon^e_{ij}, \tag{3.2}$$

where

$$\epsilon^e_{ij} = \epsilon_{ij} - \epsilon^T_{ij} \tag{3.3}$$

is the elastic part of the strain tensor ϵ_{ij}:

$$\epsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}). \tag{3.4}$$

In this chapter we will first describe an extremely direct method to evaluate $\tau_{\mathbf{n}}$, then we will illustrate an alternative method which utilizes a domain integral representation for the energy perturbation δE^T_{EL} of exp. (2.30), and discuss its numerical implementation.

3.2 A Direct Approach: The Computer Program POSTABQ

Consider a generic boundary value problem (BVP) for which we intend to evaluate $\tau_{\mathbf{n}}$ at the interface of an inclusion Ω (Fig. 3.1(a)). We first build a finite element model for the BVP, discretizing the domain under examination and applying appropriate boundary conditions (Fig. 3.1(b)). This model represents the input for the structural finite element code, ABAQUS, which will solve the BVP. ABAQUS provides, in output, values at nodes and integration points for the elastic energy density W, the displacement field $\vec{u}$ and the stress tensor σ. Thus, one simple way to calculate the force on the interface would be to postprocess the ABAQUS output file, evaluating the quantities $[W], \vec{t}, [\frac{\partial \vec{u}}{\partial n}]$, and substitute these values in expression (2.31(b)) to directly obtain $\tau_{\mathbf{n}}$.

This method has been implemented in the computer program POSTABQ. The program POSTABQ has been conceived as a multi-purpose postprocessor for ABAQUS. The current version of the program can handle only 2-D problems with second-order eight-node isoparametric elements, but has been structured so that its applicability can be easily extended to cope with 3-D geometries. The user supplies a set of nodes (by way of their number in the F.E. model) which define a "path" — in our case the path will be the interface — and the program evaluates the path geometry: curvilinear coordinate and normal vector to the path, topology and connectivity of the elements around the path, etc. Then the user asks the program to perform a sequence of operations on a certain number of selected field variables along the path. These operations can include reading the quantities from the ABAQUS file and storing them in local arrays, evaluating gradients of scalar and vector fields, evaluating the components of vectors and second order tensors in a rotated reference frame, evaluating the dot product of vector and tensor fields with the normal to the path (thus calculating the normal component of these quantities), and printing the results of such operations in an ASCII file.

The user can also program a user-subroutine where he can process the quantities resulting from the previous operations. Thus, in order to calculate $\tau_{\mathbf{n}}$ on the interface, the user will

(a) provide an ordered list of the nodes along the interface;

(b) require the program to:

- read and store the values of the strain energy density W;
- read the values of the stress field and evaluate the components of the traction vector

$$t_i = \sigma_{ij} n_j;$$

- read the value of the displacement field $\vec{u}$, evaluate the gradient of the displacement field $\nabla\vec{u}$, and evaluate the derivative of $\vec{u}$ along the normal direction:

$$\frac{\partial u_i}{\partial n} = u_{i,j}n_j;$$

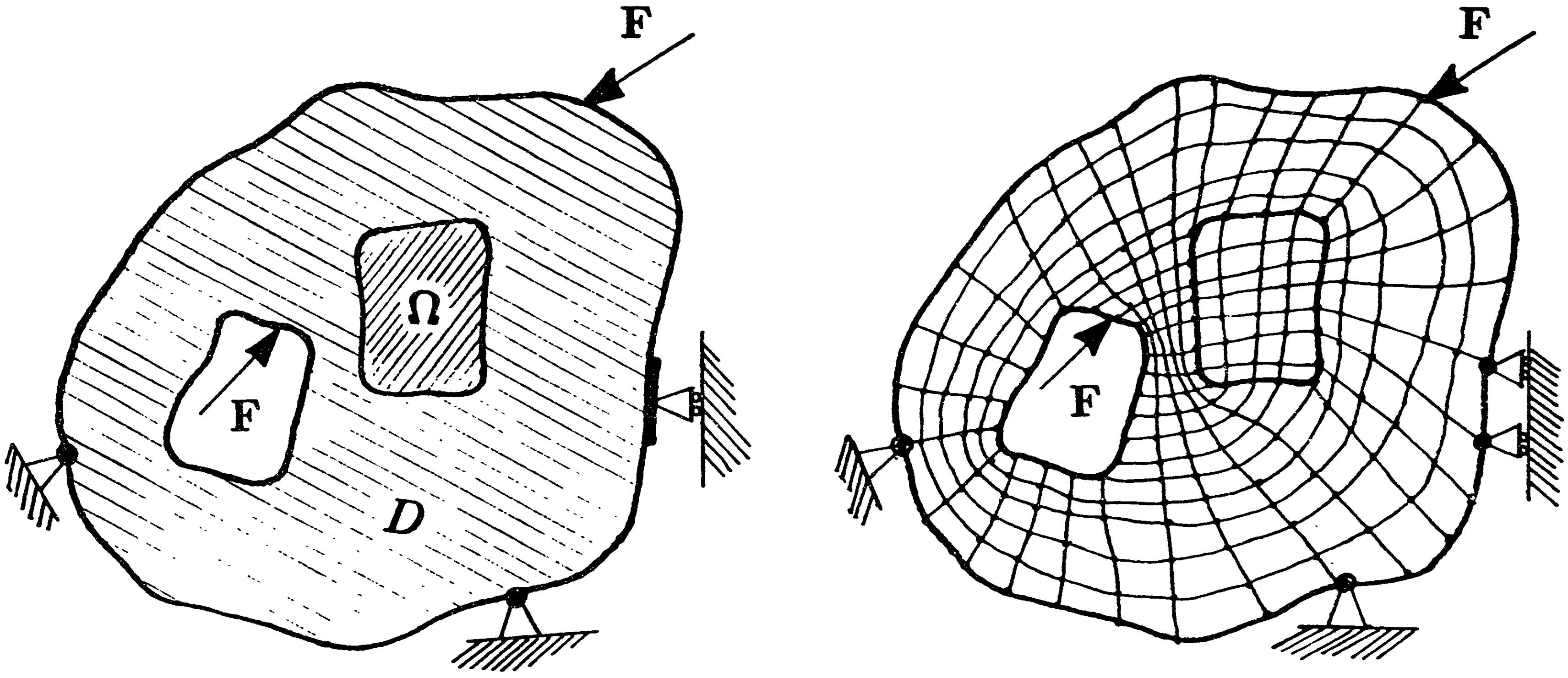
(c) write a user subroutine where the results of the previous operations are combined to obtain

$$\tau_{\mathbf{n}} = [W] - t_i[\frac{\partial u_i}{\partial n}]$$

at all nodes along the interface.

The listing of the program POSTABQ is given in APPENDIX I.

The advantage of this method lies in its simplicity and therefore in the limited amount of calculations required. Unfortunately, it has one main drawback: since the evaluation of $\tau_{\mathbf{n}}$ directly relies on the determination of elastic field *at* the interface, we can expect that our results will be affected by some numerical noise. The area adjacent to the interface is in fact directly affected by numerical disturbances arising from the discontinuity of the elastic field. Extremely refined meshes, therefore, are required to limit this effect. An alternative approach which can overcome this kind of difficulty will be presented in the next section.



(a) Boundary value problem

(b) Finite element model

Figure 3.1 Finite element model of a boundary value problem

3.3 A Domain Integral Approach: The Computer Program DOMAIN

3.3.1 Historical perspective

The J-integral, introduced by Eshelby [45] together with a number of other path-independent contour integrals, has quickly become one of the most-used crack parameters in fracture mechanics.

Its role in the context of nonlinear fracture mechanics was introduced by Rice [40, 41] who has interpreted J as a measure of the intensity of the crack tip field: if evaluated on a contour placed within the region of HRR dominance [42, 43], the value of J serves as a unique scaling amplitude for the HRR singular fields. The initiation of crack growth in plastically deforming bodies can thus be correlated with a critical value of the J-integral.

The reason for the "success" of the J-integral lies mainly in the mentioned property of path-independence: the integrand is divergence-free for a material that admits a strain energy function, so that the J-integral has the same value for all open paths beginning on one face of the crack and ending on the opposite face. This feature plays an important role since it allows a direct computation of the strength of crack tip singularities by evaluating the integral in regions remote from the crack tip, where the numerical field solution is more reliable.

With the help of weighting functions, the J contour integral can be recast into a finite *domain* volume *integral.* This alternative formulation is naturally compatible with the finite element method.

In fact, the domain-approach first appeared already cast in its finite element implementation: the Virtual Crack Extension (VCE) technique, introduced by Parks [47], actually corresponds to a finite element formulation of the domain integral approach. By the VCE technique [48, 49, 50], accurate pointwise values of the energy release rate can be obtained.

This method has been interpreted in a somewhat more general context by deLorenzi [51, 52], who has obtained a more compact continuum formulation of the VCE technique.

The use of domain integral techniques in fracture mechanics has been subsequently addressed in several studies [44, 46, 53, 54], where thermal loads and kinetic energy have been rigorously incorporated in the formulation.

Here we present a straightforward extension of this methodology, which will allow us to evaluate the force acting on a material interface.

We are led to this approach by the same reasons which have brought about the use of contour and domain integrals in fracture mechanics. We must deal with a structural discontinuity (in our case, the interface; in fracture mechanics, the crack), which locally affects our numerically-determined field solutions. Thus we need to develop a formulation in which the desired local quantities are expressed in terms of the field solution in regions as remote as possible from the discontinuity.

In the next section, 3.3.2, we present a derivation of the domain integral expression for the energy variation, δE^T_{EL}, that corresponds to a migration of the interface. We then outline in Section 3.3.3 the finite element formulation of the domain integral method to determine the force on the interface. In Appendix II, we give the listing of the computer program DOMAIN in which this methodology has been implemented.

3.3.2 Derivation of a domain integral expression for the energy perturbation

Let S be the interface separating an inclusion Ω and an otherwise homogeneous medium (Fig. 3.2(a)). If $\vec{n}$ is the unit outward normal to the interface, we can define a perturbation of the interface $\delta\vec{\xi}$ in terms of its normal component $\delta\xi_{\mathrm{n}}$ as

$$\delta\vec{\xi}(\vec{x}) = \delta\xi_{\mathrm{n}}(\vec{x})\,\vec{n}(\vec{x}), \tag{3.5}$$

thus $\delta\vec{\xi}$ is a vector quantity defined at all locations $\vec{x}$ along the interface.

In section 2.3 we have derived an expression for the change in the total energy, δE^T_{EL}, associated with the perturbation $\delta\vec{\xi}$:

$$\delta E^T_{EL} = \int_S \delta\xi_i[P_{ij}]n_j dS. \tag{3.6}$$

Let S^+ now be any surface completely surrounding S, and S^- any surface completely surrounded by S, as illustrated in Fig. 3.2(b). It is also assumed that the volume between S^+ and S^- is simply connected. In order to develop a volume integral expression for δE^T_{EL}, we first introduce the vector field $\vec{q}$ defined over the domain V enclosed by S^- and S^+. The vector field $\vec{q}$ is defined so that it satisfies the following requirements:

$$\begin{array}{lll} \text{(a)} & \vec{q}(\vec{x}) \equiv \delta\vec{\xi}(\vec{x}); & \text{if } \vec{x} \text{ identifies a point on } S; \\ \text{(b)} & \vec{q}(\vec{x}) \equiv 0; & \text{if } \vec{x} \text{ identifies a point on } S^+ \text{or } S^-; \\ \text{(c)} & \vec{q}(\vec{x}) & \text{is a smooth function in } V. \end{array} \tag{3.7}$$

Then (3.5) can be written as

$$\delta E^T_{EL} = -\int_S q_i(\vec{x})[P_{ij}]n_j dS. \tag{3.8}$$

We can recast (3.8) in the form

$$\delta E^T_{EL} = I^+ + I^-, \tag{3.9}$$

where

$$I^+ = -\int_S q_i P^+_{ij} n_j dS \tag{3.10(a)}$$

$$I^- = \int_S q_i P^-_{ij} n_j dS, \tag{3.10(b)}$$

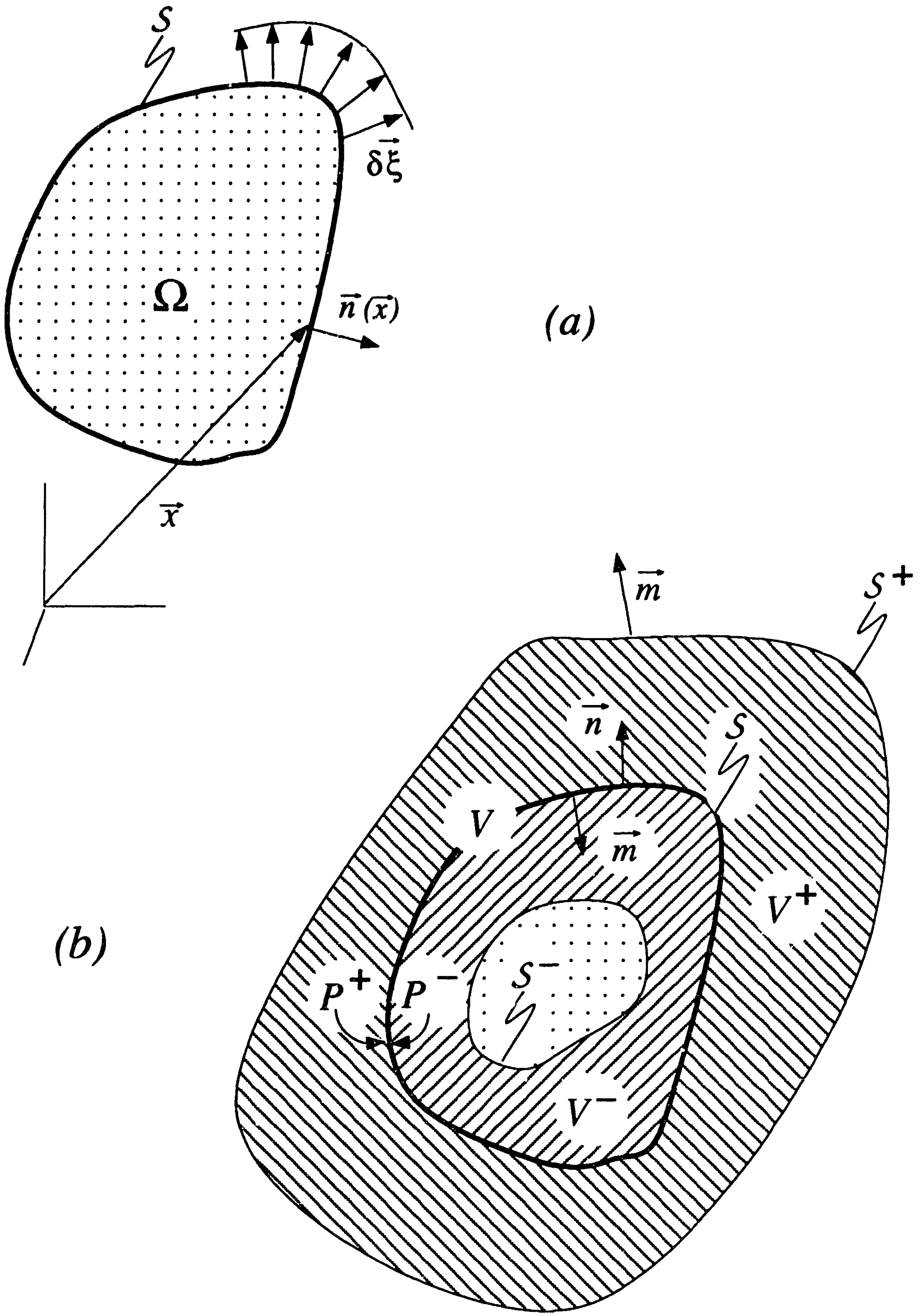


Figure 3.2 Definition of the domain of integration.

with P_{ij}^{+} being the EMT on the "plus side" of the interface (Fig. 3.2(b)) and P_{ij}^{-} the EMT on the "minus side".

Now consider the expression (3.10(b)) for I^{-}; if we indicate with V^{-} the domain delimited by S^{-} and S, and with ∂V^{-} the boundary of V^{-}, since $\vec{q}$ vanishes on S^{-} we can write:

$$I^{-} = \int_{\partial V^{-}} q_i P_{ij}^{-} n_j dS \tag{3.11}$$

and, applying the divergence theorem:

$$I^{-} = \int_{V^{-}} (q_i P_{ij})_{,j} dV \tag{3.12(a)}$$

or

$$I^{-} = \int_{V^{-}} (P_{ij} q_{i,j} + P_{ij,j} q_i) dV. \tag{3.12(b)}$$

In section 2.2 we have derived expression (2.16) for the divergence of P. If we substitute (2.18) in (2.16), we obtain

$$P_{ij,j} = (\frac{\partial W}{\partial X_i})_{inhom} + \frac{\partial W}{\partial \epsilon_{mn}^{T}} \frac{\partial \epsilon_{mn}^{T}}{\partial X_i} + \rho b_k u_{k,i} \tag{3.13}$$

Using definition (3.2) for the strain energy density and substituting for the elastic strain (3.3) we obtain:

$$\frac{\partial W}{\partial \epsilon_{mn}^{T}} = -\frac{\partial W}{\partial \epsilon_{mn}} = -\sigma_{mn} \tag{3.14}$$

so that:

$$P_{ij,j} = (\frac{\partial W}{\partial X_i})_{inhom} - \sigma_{mn} \epsilon_{mn,i}^{T} + \rho b_k u_{k,i}. \tag{3.15}$$

If the material within the domain V^{-} is homogeneous ($(\partial W / \partial X_j)_{inhom} = 0_j$), we have:

$$I^{-} = \int_{V^{-}} \{P_{ij} q_{i,j} + (\rho b_k u_{k,i} - \sigma_{mn} \epsilon_{mn,i}^{T}) q_i\} dV. \tag{3.16}$$

We can perform an analogous sequence of operations on I^{+} :

$$I^{+} = \int_{\partial V^{+}} q_i P_{ij}^{+} m_j dS, \tag{3.17}$$

where V^{+} is the domain delimited by S and $S^{+}, \partial V^{+}$ is its boundary, and $\vec{m}$ is the unit outward normal to ∂V^{+} ($\vec{m} = -\vec{n}$ on S).

By applying the divergence theorem and substituting for $P_{ij,j}$, if the material within V^{+} is homogeneous, we obtain:

$$I^+ = \int_{V^+} \{P_{ij}q_{i,j} + (\rho b_k u_{k,i} - \sigma_{mn}\epsilon^T_{mn,i})q_i\}dV. \tag{3.18}$$

Substituting (3.16) and (3.18) in (3.9) and noting that $V \equiv V^+ + V^-$ we obtain

$$\delta E^T_{EL} = \int_V \{P_{ij}q_{i,j} + (\rho b_k u_{k,i} - \sigma_{mn}\epsilon^T_{mn,i})q_i\}dV, \tag{3.19(a)}$$

or

$$\delta E^T_{EL} = \int_V \{(Wq_{i,i} - \sigma_{kj}u_{k,i}q_{i,j}) + (\rho b_k u_{k,i} - \sigma_{mn}\epsilon^T_{mn,i})q_i\}dV. \tag{3.19(b)}$$

We have thus obtained a volume-integral expression for δE^T_{EL}. Note how, as long as the $\vec{q}$-field and the domain V are chosen so that requirements (3.7) are satisfied, expressions (3.19) will give us the same value for δE^T_{EL} for any choice of the domain V. This *invariance of the domain integral* representation of δE^T_{EL} with respect to variation in domain size and shape will provide a useful independent check on the consistency and quality of our numerical calculations. This is indeed another attractive feature of the domain integral approach.

If we substitute expression (2.31(a)) in (2.30) we have

$$\delta E^T_{EL} = -\int_S \delta\xi_{\mathrm{n}}\ \tau_{\mathrm{n}} dS; \tag{3.20}$$

if we equate the right hand side of Eq. (3.20) and (3.19) we have

$$-\int_S \delta\xi_{\mathrm{n}}\tau_{\mathrm{n}} dS = \int_V \{(Wq_{i,i} - \sigma_{kj}u_{k,i}q_{i,j}) + (\rho b_k u_{k,i} - \sigma_{mn}\epsilon^T_{mn,i})q_i\}dV. \tag{3.21}$$

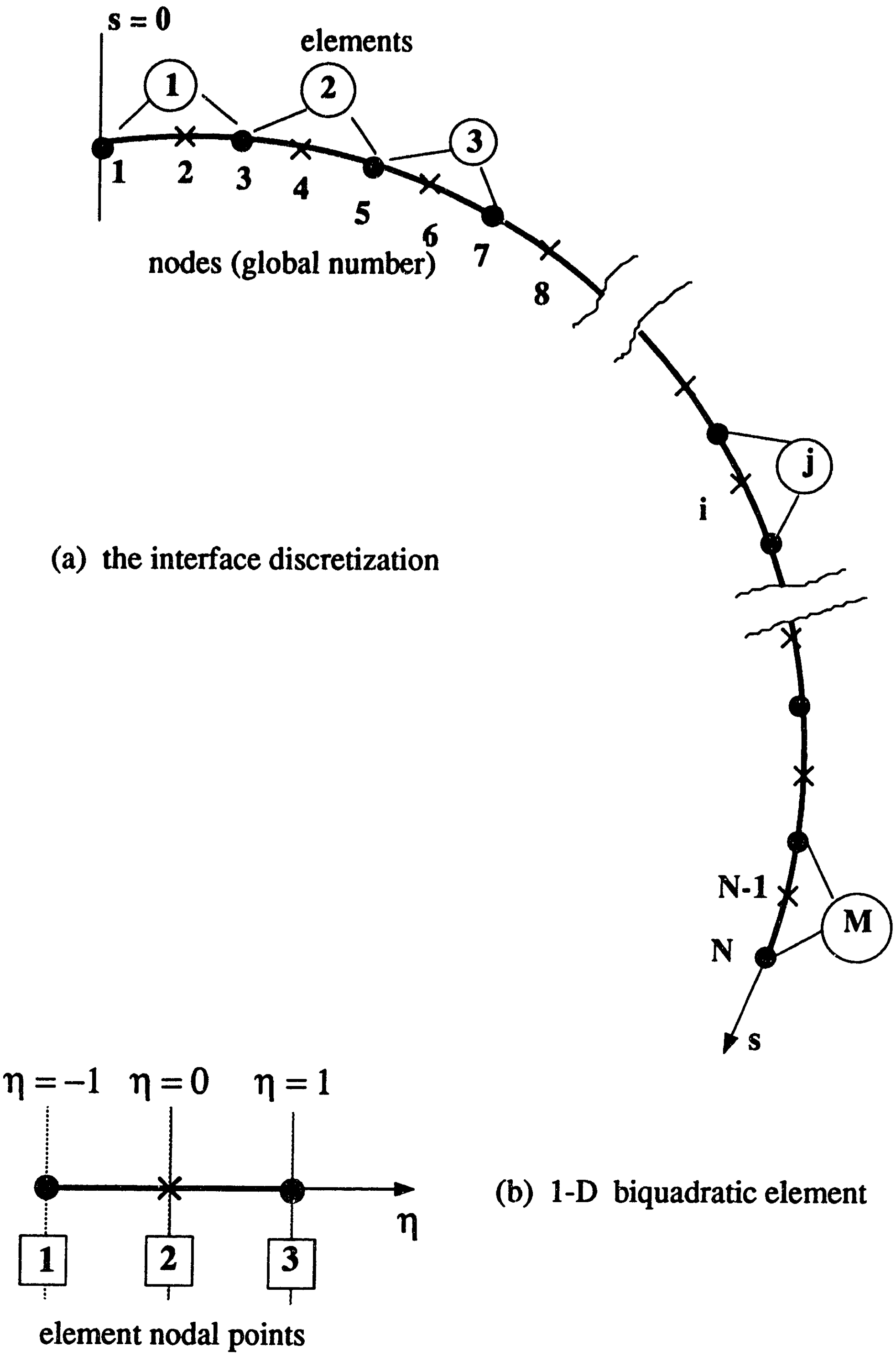
The desired quantity, τ_{n}, appears on the left-hand side, while we can evaluate the expression on the right-hand side in terms of known field quantities: $W, \sigma, u, b, \epsilon^T$.

In the next section, 3.3.3, we will describe how, with an appropriate choice of the perturbation field ($\delta\vec{\xi}$ and $\vec{q}$), we can extract local values of τ_{n} from Eq. (3.21).

3.3.3 Finite element implementation

We will discuss a numerical implementation of the domain integral approach, for 2-D configurations, within the context of the displacement finite-element method using biquadratic Lagrangian shape functions.

We define on the interface S, N nodes and M biquadratic (3-node) isoparametric elements (Fig. 3.3(a)). For each element we can identify an isoparametric coordinate η — as shown in Fig. 3.3(b) — and define the basic shape functions:



(a) the interface discretization

(b) 1-D biquadratic element

Figure 3.3 Finite element discretization of the interface.

$$N^1(\eta) = \frac{1}{2}\eta(\eta - 1)$$
$$N^2(\eta) = (1 - \eta)(1 + \eta)$$
$$N^3(\eta) = \frac{1}{2}\eta(\eta + 1).$$

Throughout our presentation, superscripts refer to the element nodal point number (as in Fig. 3.3(b)), while subscripts refer to the global node/element number along S.

We can express the value of any quantity, g, at location η along the element j in terms of the nodal values of g and of the shape functions:

$$g(\eta) = N^i(\eta)\langle g^i\rangle_j, \tag{3.22}$$

where the summation convention is implied also for superscripts and $\langle g^i\rangle_j$ is the value of g at node i of element j.

In particular, if s is the curvilinear coordinate along S, we have

$$s(\eta) = N^i(\eta)\langle s^i\rangle_j \tag{3.23}$$

and

$$ds = \frac{ds}{d\eta}d\eta = \frac{dN^i}{d\eta}\langle s^i\rangle_j d\eta. \tag{3.24}$$

We can also write:

$$\tau_{\mathrm{n}}(\eta) = N^i(\eta)\langle\tau_{\mathrm{n}}^i\rangle_j, \tag{3.25}$$
$$\delta\xi_{\mathrm{n}}(\eta) = N^i(\eta)\langle\delta\xi_{\mathrm{n}}^i\rangle_j. \tag{3.26}$$

We can then recast Eq. 3.20 in the form

$$\delta E_{EL}^T = \sum_{j=1}^{M}\{-\int_{-1}^{1} N^k(\eta)\langle\delta\xi_{\mathrm{n}}^k\rangle_j N^i(\eta)\langle\tau_{\mathrm{n}}^i\rangle_j \frac{dN^m}{d\eta}\langle s^m\rangle_j d\eta\}, \tag{3.27}$$

where we have extended the summation to all the elements of S and we have substituted the area element dS with the line element ds since we are modeling 2-D problems.

Now consider N *different* patterns for the perturbation $\delta\xi_{\mathrm{n}}$, defined so that, for the p^{th} *perturbation pattern* (Fig. 3.5(a)):

$$(\delta\xi_{\mathrm{n}})_p \equiv 1 \text{ at the } p^{th} \text{ node}$$
$$(\delta\xi_{\mathrm{n}})_p \equiv 0 \text{ at } \textit{all} \text{ other nodes.}$$

If we write Eq. (3.27) for the p^{th} perturbation, only the elements j_p to which node p belongs will give a contribution to the p^{th} perturbation in energy $\delta E^T_{EL_p}$, so that we obtain

$$\delta E^T_{EL_p} = \sum_{j_p}\{-\int_{-1}^{1} N^l(\eta)N^i(\eta)\langle\tau^i_{\mathbf{n}}\rangle_{j_p}\frac{dN^m}{d\eta}\langle s^m\rangle_{j_p}d\eta\}, \tag{3.28}$$

where we are summing *only* over the elements j_p which contain node p and l is the position of node p in element j_p.

We can also write (3.28) as:

$$\delta E^T_{EL_p} = \sum_{j_p}[\tau^i_{\mathbf{n}}]_{j_p}[s^m]_{j_p}\{-\int_{-1}^{1} N^l(\eta)N^i(\eta)\frac{dN^m}{d\eta}d\eta\}, \tag{3.29}$$

and introduce the notation

$$B^{lim} = -\int_{-1}^{1} N^l(\eta)N^i(\eta)\frac{dN^m}{d\eta}d\eta. \tag{3.30}$$

The quadratic form B^{lim} can be explicitly evaluated for all combinations of indices. The results are shown in Table 3.1. Eq. (3.29) can then be written in a more compact form as

$$\delta E^T_{EL_p} = \sum_{j_p}\langle\tau^i_{\mathbf{n}}\rangle_{j_p}\langle s^m\rangle_{j_p}B^{lim}. \tag{3.31}$$

If we write the same expression for all the N possible patterns, we obtain a system of N equations in the N unknown $\tau_{\mathbf{n}}$ nodal values:

$$\mathrm{M}_{pr}\tau_{\mathbf{n}_r} = \delta E^T_{EL_p}, \tag{3.32(a)}$$

or, in matrix notation:

$$[\mathrm{M}]\{\tau_{\mathbf{n}}\} = \{\delta E^T_{EL}\}, \tag{3.32(b)}$$

where

$\{\delta E^T_{EL}\}$ is the vector of the energy perturbations, $\delta E^T_{EL_p}$

$\tau_{\mathbf{n}_r}$ is the nodal value of $\tau_{\mathbf{n}}$ at node r

M_{pr} is the p, r component of the coefficient matrix [M] which is given by:

$$\mathrm{M}_{pr} = \sum_{j_{p,r}}[s^m]_{j_{p,r}}B^{ltm}. \tag{3.33}$$

Here the summation is extended only to the elements $j_{p,r}$ which contain *both* node r and node p (if $r \neq p$ there will be only one element) and t is the position of node r in element $j_{p,r}$.

We can thus easily construct the coefficient matrix [M] from Table 3.1 and the curvilinear coordinates of the nodes.

Table 3.1

B^{ijk} Values

$B^{111} = 1/3$	$B^{112} = -2/5$	$B^{113} = 1/15$
$B^{121} = 1/5$	$B^{122} = -4/5$	$B^{123} = 1/15$
$B^{221} = 8/15$	$B^{222} = 0$	$B^{223} = -8/15$
$B^{131} = -1/30$	$B^{132} = 0$	$B^{133} = 1/30$
$B^{231} = -1/15$	$B^{232} = 4/15$	$B^{233} = -1/5$
$B^{331} = -1/15$	$B^{232} = 2/5$	$B^{333} = -1/3$

To obtain the values of $\tau_{\mathbf{n}}$ from system (3.32) we need now to evaluate the right-hand side vector $\{\delta E^T_{EL}\}$. Consider a finite element discretization of the region surrounding the interface, as shown in Fig. 3.4(a).

We use 2-D isoparametric 8-node elements, for which the nodal point numbers are shown in the isoparametric (η, ζ)-space in Fig. 3.4(b). The 2-D domain, V, around the interface can now be defined by considering a certain number of layers of elements around the interface as shown in Fig. 3.4(c). We can evaluate numerically each $\delta E^T_{EL_p}$, integrating expression 3.19(b) over V. We first define the perturbation field $(\vec{q}\,)_p$ associated with the pth pattern of interface perturbation $(\delta\xi_{\mathbf{n}})_p$. We can express the $(\vec{q})_p$ field in terms of nodal values $\langle\vec{q}\,^k\rangle_p$and 2-D biquadratic shape functions as

$$(\vec{q})_p(\eta,\zeta) = \mathcal{N}^k(\eta,\zeta)\langle\vec{q}\,^k\rangle_p \tag{3.34}$$

where $\mathcal{N}^k(\eta,\zeta)$ are the 2-D shape functions expressed in terms of the isoparametric coordinates (η,ζ) (See Table 3.2). The nodal values of the pth perturbation field, $(\vec{q}\,^k)_p$, can be chosen freely as long as

1. $\langle\vec{q}\,^k\rangle_p \equiv 1 \cdot \vec{n}_p$ at node p on the interface
2. $\langle\vec{q}\,^k\rangle_p \equiv 0$ at all other nodes of the interface
3. $\langle\vec{q}\,^k\rangle_p \equiv 0$ at all nodes along the boundary of V.

A possible choice for the nodal values $\langle\vec{q}\,^k\rangle_p$ is shown in Fig. 3.5(b).

Table 3.2

2-D Shape Functions

$\mathcal{N}^1(\eta,\zeta) = (-1/4)(1-\eta)(1-\zeta)(1+\eta+\zeta)$
$\mathcal{N}^2(\eta,\zeta) = (-1/4)(1+\eta)(1-\zeta)(1+\eta+\zeta)$
$\mathcal{N}^3(\eta,\zeta) = (-1/4)(1+\eta)(1+\zeta)(1+\eta-\zeta)$
$\mathcal{N}^4(\eta,\zeta) = (-1/4)(1-\eta)(1+\zeta)(1+\eta-\zeta)$
$\mathcal{N}^5(\eta,\zeta) = (1/2)(1-\eta)(1+\eta)(1-\zeta)$
$\mathcal{N}^6(\eta,\zeta) = (1/2)(1-\zeta)(1+\eta)(1+\zeta)$
$\mathcal{N}^7(\eta,\zeta) = (1/2)(1-\eta)(1+\eta)(1+\zeta)$
$\mathcal{N}^8(\eta,\zeta) = (1/2)(1-\zeta)(1+\zeta)(1-\eta)$

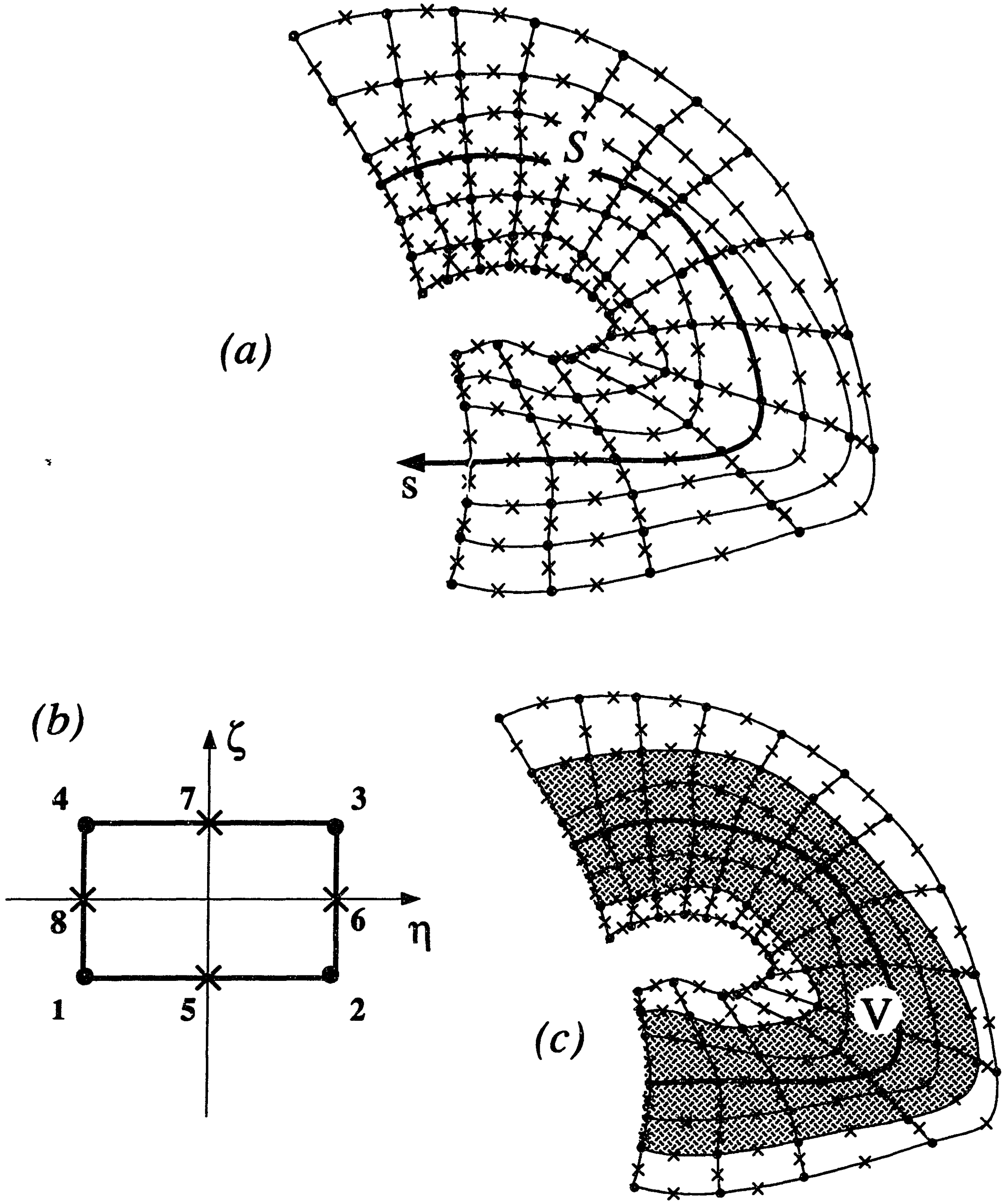
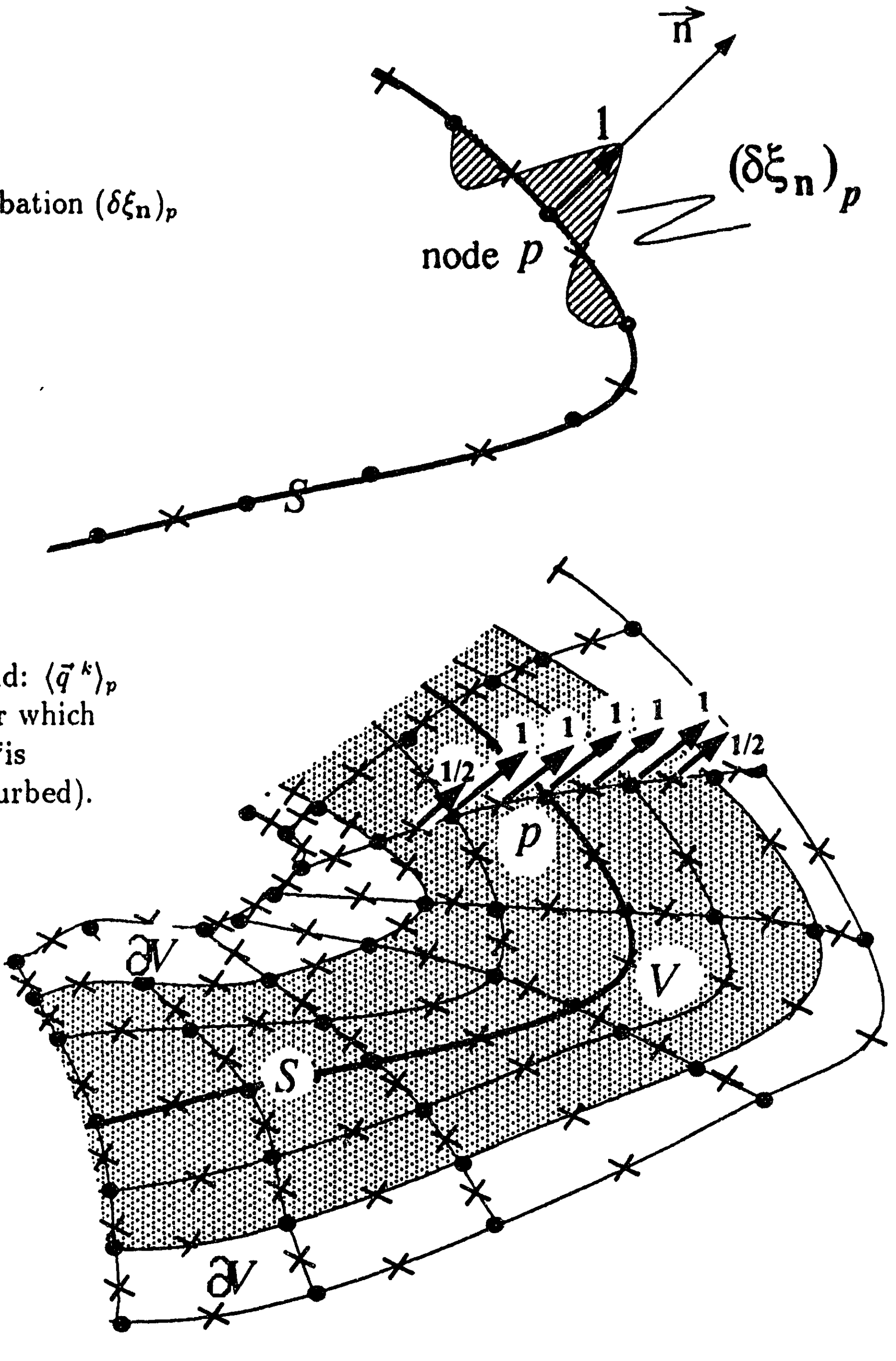


Figure 3.4 Finite element discretization of the domain of integration.



(a) Interface perturbation $(\delta\xi_n)_p$

(b) Perturbation field: $\langle\vec{q}^{\,k}\rangle_p$ (only the nodes for which the vector $\vec{q}^{\,k}$is indicated are perturbed).

Figure 3.5 Perturbation patterns.

The integral form (3.19a) can now be written as

$$\delta E_{EL_p}^T = \sum_V \int_{-1}^{1}\int_{-1}^{1} [P_{ij}(q_{i,j})_p + \{\rho b_k u_{k,i} - \sigma_{mn}\epsilon_{mn,i}^T\}(q_i)_p] \mid J \mid d\eta d\zeta, \tag{3.35}$$

where the sum is extended to all elements in V, and $\mid J \mid$ is the determinant of the Jacobian matrix:

$$\mid J \mid = \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \zeta} - \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \zeta}. \tag{3.36}$$

The integrand in (3.35) is, for each element, a function of the isoparametric coordinates η, ζ :

$$\mathcal{F}(\eta,\zeta) = P_{ij}(q_{i,j})_p + \{\rho b_k u_{k,i} - \sigma_{mn}\epsilon_{mn,i}^T\}(q_i)_p \mid J \mid, \tag{3.37}$$

so that each of the integrals can be evaluated numerically, e.g., by Gauss quadrature:

$$\int_{-1}^{1}\int_{-1}^{1} \mathcal{F}(\eta,\zeta) d\eta d\zeta = \sum_i \sum_j W_i W_j \mathcal{F}(\eta_i, \zeta_j), \tag{3.38}$$

where $\mathcal{F}(\eta_i, \zeta_j)$ is the value of $\mathcal{F}$ at integration point (i, j) and the W_i, W_j are weights for Gauss quadrature.

Thus, if a finite element analysis has been performed on the body, we can calculate the values of $\mathcal{F}$ at the integration points of the elements inside V, substitute these values in (3.38) and apply (3.35) to evaluate the right-hand side of system (3.32).

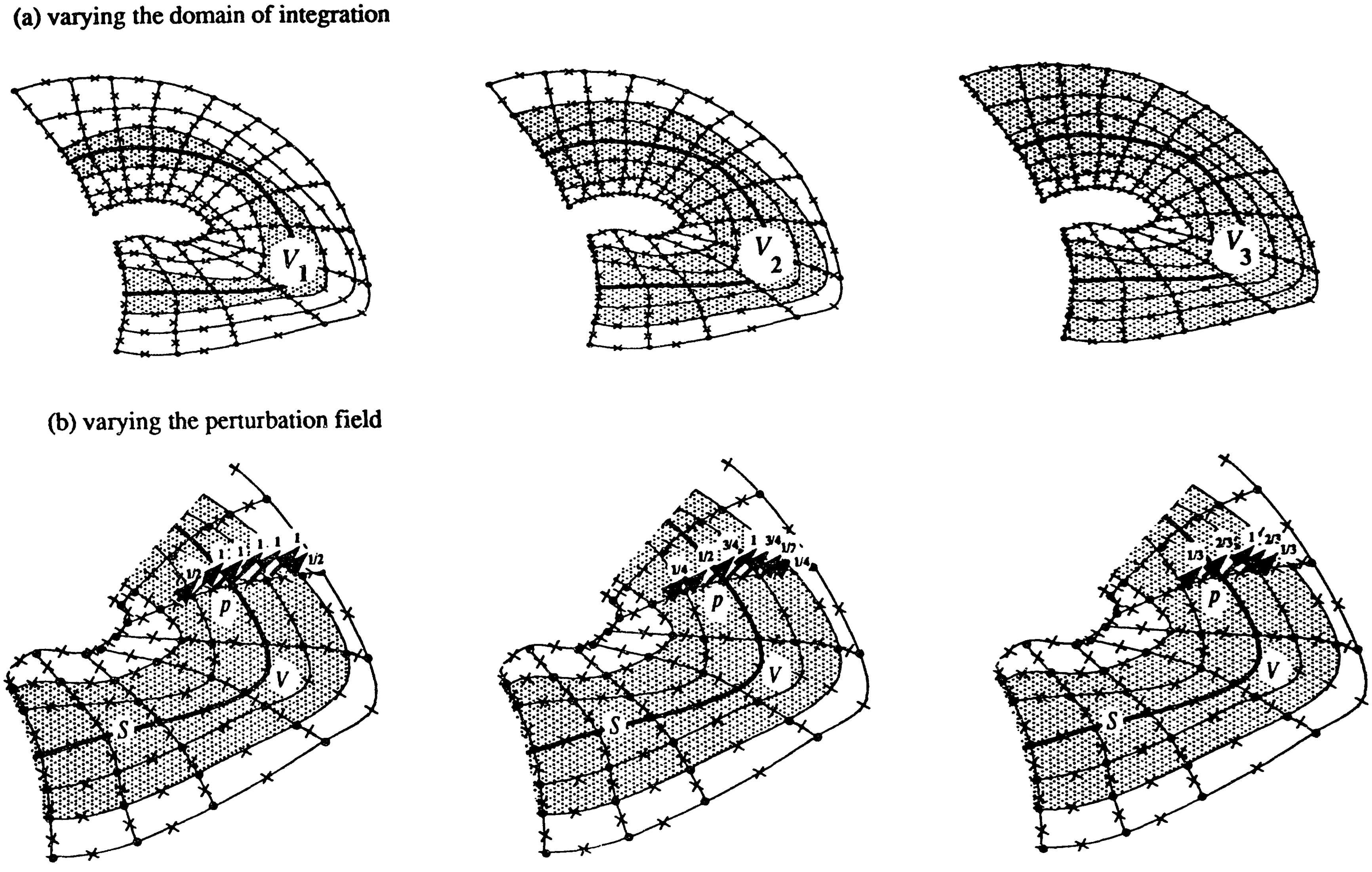
As we have already mentioned in section 3.32, we can use the invariance property of the domain integral expression for δE_{EL}^T as a check for the quality of our numerical calculations.

We can consider several choices for the domain of integration, V, and/or for the form of the perturbation field $\vec{q}$ (See Fig. 3.6).

For each of these choices we can evaluate the RHS vector $\{\delta E_{EL}^T\}$ and solve the system (3.32) for the force on the interface $\{\tau_n\}$.

Differences in the values of the $\{\tau_n\}$ vectors can be regarded as a measure of the numerical error in the solution.

This method has been implemented in the computer program DOMAIN for which a listing of the symbolic code is given in Appendix II.


(a) varying the domain of integration

(b) varying the perturbation field

Figure 3.6 Test for the accuracy of the numerical solution.

CHAPTER 4
ANALYSIS OF RAFTING

4.1 Introduction

The numerical methods described in Chapter 3 allow us to evaluate the distribution of τ_n over a material interface. As discussed in Chapter 2, this is a quantity directly related to the tendency of the interface to migrate and, with the convention used throughout our derivation, a positive value of τ_n, at a certain location, indicates a tendency for the inclusion to grow at that location. If the value of τ_n along a precipitate-matrix interface is not uniform, this indicates a tendency for the inclusion to modify its morphology.

In this chapter we will apply these concepts to the analysis of rafting in $\gamma - \gamma'$ Ni-superalloys.

In paragraph 4.2, we will first assess the accuracy of our models by comparing data obtained by numerical calculations with analytical solutions for the simple case of a cylindrical inclusion in an infinite matrix. This test case will also provide the opportunity to construct an "evolution map" which interestingly compares with Pineau's map (see Section 1.4).

In paragraph 4.3 we will then introduce the finite element model, for the $\gamma - \gamma'$ cuboidal morphology, that we will use to perform the rafting analysis, and we will identify the set of alloys for which we will carry out our calculations.

In paragraph 4.4 we will present and discuss the results of purely elastic analyses, where the force on the interface is evaluated for the initial conditions of applied stress and misfit, and no misfit-relaxation effect due to creep is considered.

In paragraph 4.5 we will follow the evolution of the driving force during the very fiıst stage of primary creep in a simulated stress-annealing experiment.

In paragraph 4.6 we will finally discuss the results, draw our conclusions concerning the rafting process and offer a synoptic interpretation of numerical results and experimental data.

4.2 Test Case: A Cylindrical Inclusion in an Infinite Matrix

As a test case, we will consider the very simple problem of a cylindrical isotropic misfitting inclusion in an infinite isotropic matrix subjected, at infinity, to a state of uniaxial stress σ_∞ normal to the axis of the cylinder (Fig. 4.1(a)). The elastic solution for this problem has been first determined by Goodier [55].

In order to simplify our calculations, we will assume that the matrix and the inclusion have the same Poisson ratio ν. Then the state of stress and strain at any location $(r/r^\Omega; \theta)$, where r^Ω is the radius of the inclusion, can be determined in terms of:

E_p: the stiffness (Young's modulus) of the precipitate
E_m: the stiffness (Young's modulus) of the matrix
σ_∞: the applied stress
δ: the misfit
ν: the Poisson ratio (assumed common to matrix and precipitate)

From the analytical expression for the elastic field, given in [55], the quantity τ_n can be readily evaluated at all locations along the interface using eq. (2.31(b)).

These calculations have been implemented in the simple program GOODIER, listed in Appendix III, which evaluates τ_n at N locations along the interface (N is a user-defined quantity).

Due to the symmetry of the problem, only one fourth of the geometry needs to be considered. (We have chosen the quadrant shown in Fig. 4.1(b)).

The values of τ_n along the interface can be plotted against a normalized arclength s as shown in Fig. 4.2. The origin for s is at the very top of the particle (see Fig. 4.1(b)) so that, in the graphs in Fig. 4.2, the left part of the graph roughly corresponds to the "top" of the particle, while the right part corresponds to the "side".

Then, graph 4.2(a), where the average value of τ_n on the "top" is larger than the average value of τ_n on the "side", indicates a tendency toward a "type P" rafting behavior, with rafts aligned in the direction of the applied stress; graph 4.2(b) indicates a tendency toward an isotropic rafting behavior; and graph 4.2(c) indicates a tendency toward a "type N" rafting behavior with plates forming in the direction normal to the applied stress.

Since, for this case, the analytical distribution of τ_n can be readily found, we have been able to perform a brief parametric study varying the same characteristic quantities chosen by Pineau to construct his map (see paragraph 1.4), namely $\sigma_\infty/E_m\delta$ and E_p/E_m. The results of this study, obtained with a value of the misfit $\delta = 0.1\%$, are summarized in Fig. 4.3 and, with a broader spectrum of E_p/E_m values, in Fig. 4.4.

For each small plot two numerical values are given; the first represents the magnitude (in MPa) of the difference between the average values of τ_n on the top and on the side: this is a measure of the tendency to coarsen with a preferential orientation; the second value represents the average value of $\tau_{\mathbf{n}}$ (in MPa) along the interface: this is a measure of the average tendency for the particle to grow in order to decrease the total elastic energy.

From this set of results, we can sketch the map shown in Fig. 4.5 that can be compared with Pineau's map of Fig. 1.8. Besides the differences in the geometry of the problems (spherical inclusion for Pineau; cylindrical inclusion for our test case), the two maps differ for a fundamental reason: Pineau's map is a *stability map*: it is constructed by determining the configuration that minimizes the energy. Our map is an ***evolution map:*** it is constructed by evaluating the tendency toward shape evolution for the precipitate in the actual initial configuration.

Still, bearing in mind this basic difference, the patterns of the two maps show striking similarities.

We have also explored the effects of varying the sign and magnitude of the misfit. A mere change of the sign of the misfit does not affect the results, as can be noted by comparing Fig. 4.6, obtained with a misfit $\delta = -0.1\%$, and Fig. 4.3 Conversely, if we change the magnitude of the misfit, while the pattern of the map is not modified, the magnitude of the driving force is remarkably altered.

The plots in Fig. 4.7 have been obtained with a misfit $\delta = 0.3\%$: if we compare the numerical values of $\tau_{\mathbf{n}}$ with those in Fig. 4.3 we can notice that the force on the interface is increased by almost one order of magnitude. This result should have been expected since, for a purely elastic problem, $\tau_{\mathbf{n}}$ is a quadratic form in the stress field so that, if we increase the misfit (and thus σ_∞ to keep $\sigma_\infty / E_m \delta$ constant) of a factor 3, we can expect that $\tau_{\mathbf{n}}$ will increase of a factor $3^2 = 9$.

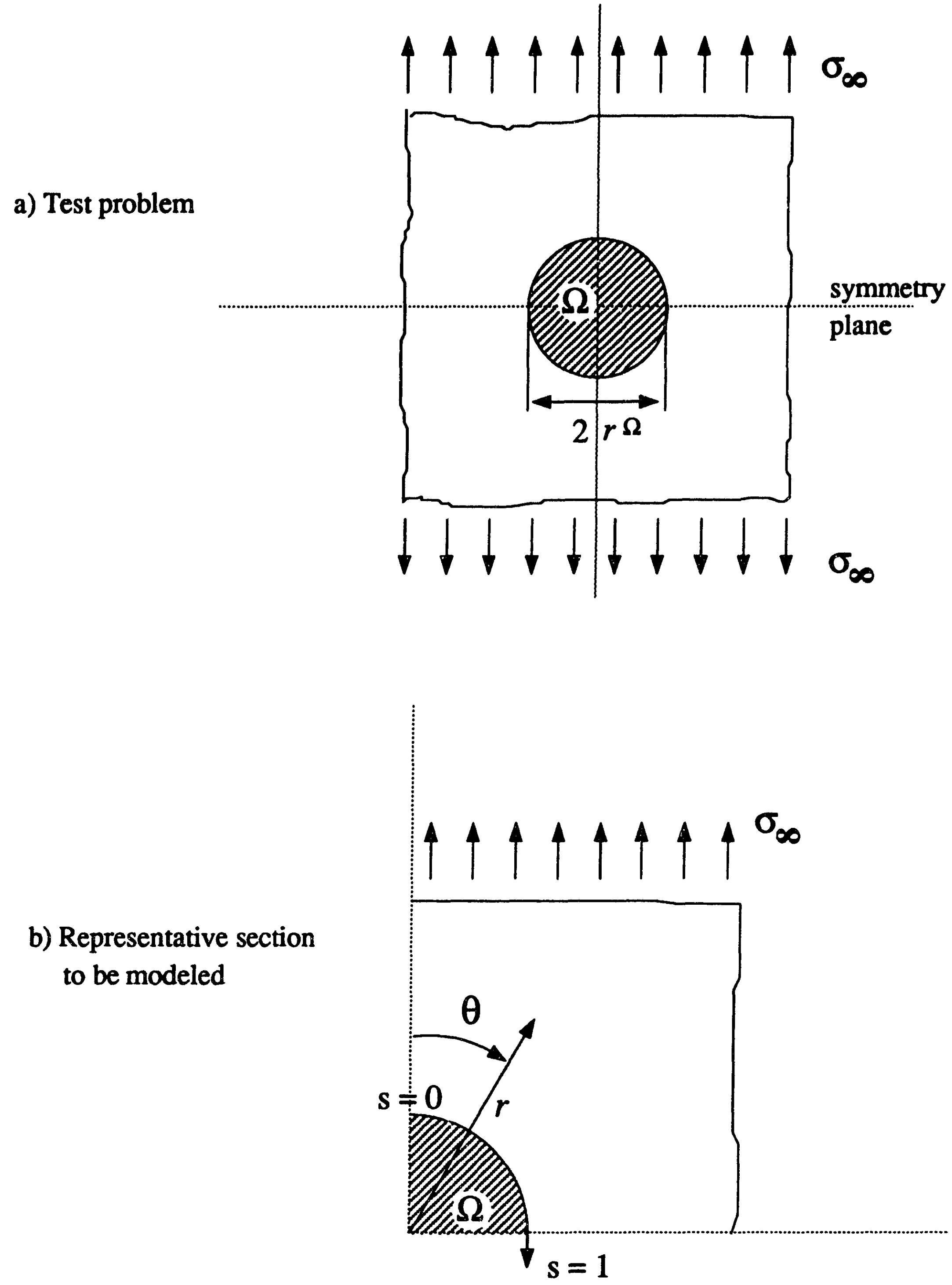
After this digression we will now return to our original task of assessing the accuracy of the numerical methods introduced in Chapter 3 to evaluate $\tau_{\mathbf{n}}$.

A boundary value problem to be solved using the finite element program ABAQUS has been defined as shown in Fig. 4.8. Plane-strain, eight-node isoparametric element have been used. Symmetry conditions are applied at the axes; the nodes along the top of the domain of analysis are constrained to have equal vertical displacement and to these nodes the external load σ_∞ is applied; the nodes along the outer edge are constrained to have equal horizontal displacements.

In this analysis the ratio of the particle radius, r^Ω, to the dimension of the domain of analysis, L_∞, is 1/20. This ratio is low enough to insure "far field" conditions on the outer boundary of the domain. The finite element mesh used for this problem is shown in Fig. 4.9. A listing of the ABAQUS input file is given in Appendix IV.

The results of the finite element analysis have been post processed using both the program POSTABQ and the program DOMAIN to obtain the distribution of $\tau_{\mathbf{n}}$ along the interface.

A comparison of analytical and numerical results, for a representative case, is shown in Fig. 4.10. In general, the domain integral method yields smoother results, but the agreement with the analytic solution is satisfactory also for the direct method. No appreciable difference in the results obtained using the domain integral method is observed when varying the domain of integration, thus confirming the good level of accuracy of the calculations. However, as we will discuss in the next paragraph, there exist particular cases in which the direct method can yield more consistent results than the domain integral method.



a) Test problem

b) Representative section to be modeled

Figure 4.1 Cylindrical inclusion in an infinite matrix.

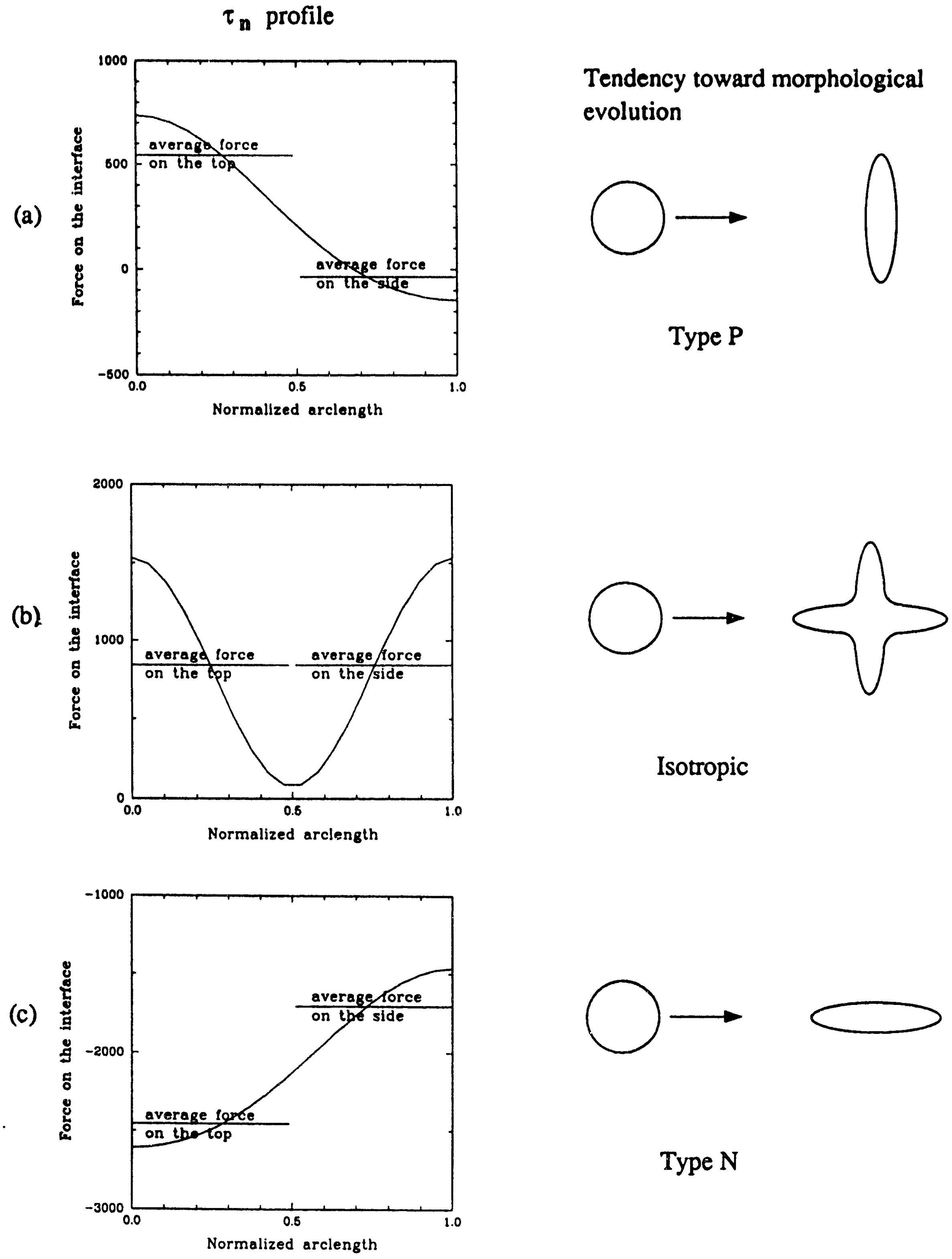


Figure 4.2 Relation between the profile of τ_n along the interface and the tendency toward morphological evolution.

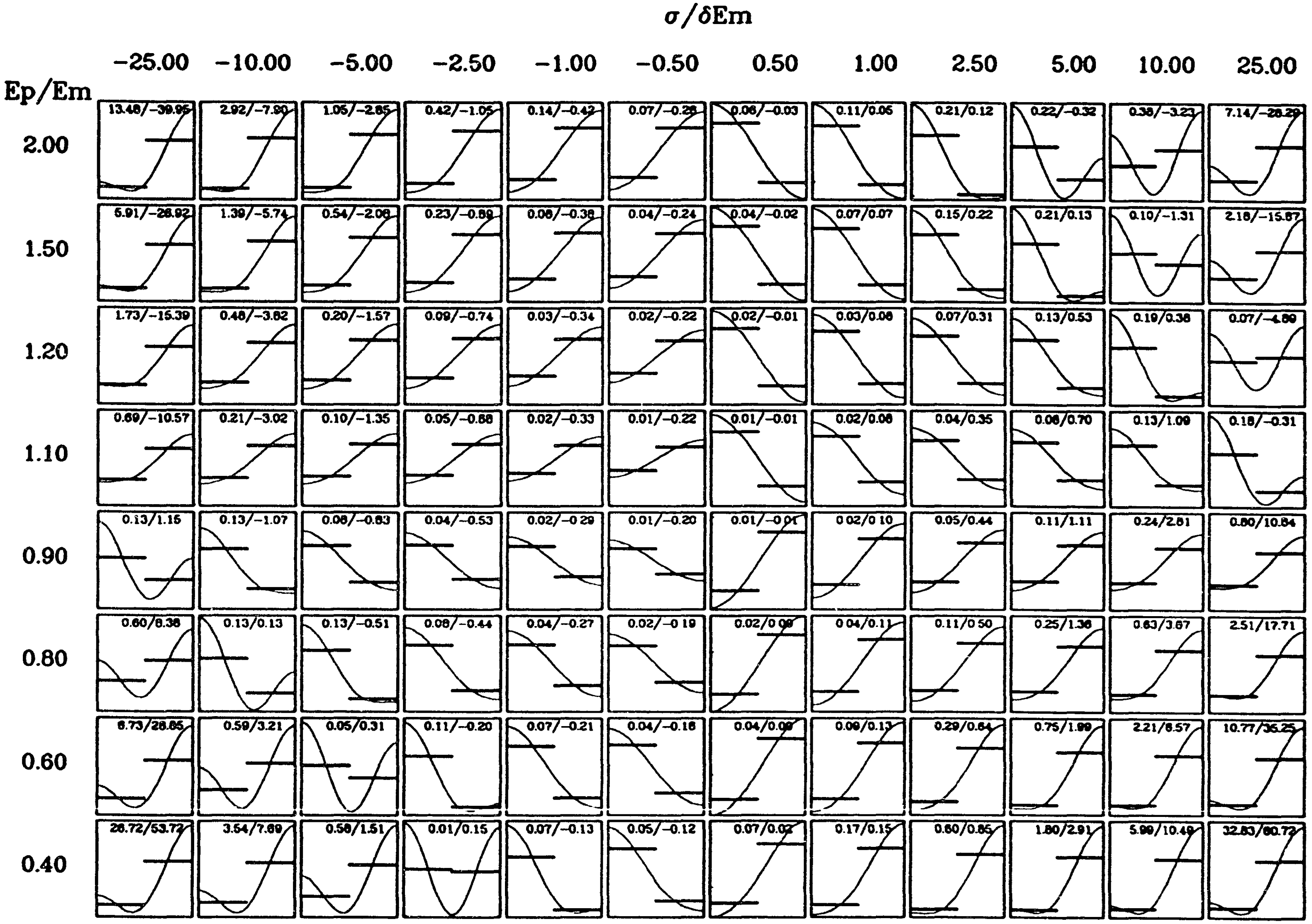


Figure 4.3 Parametric study of τ_n profile for: $\delta = 0.1\%; E_m = 100$ GPa; $0.4 \leq E_p/E_m \leq 2.0$.

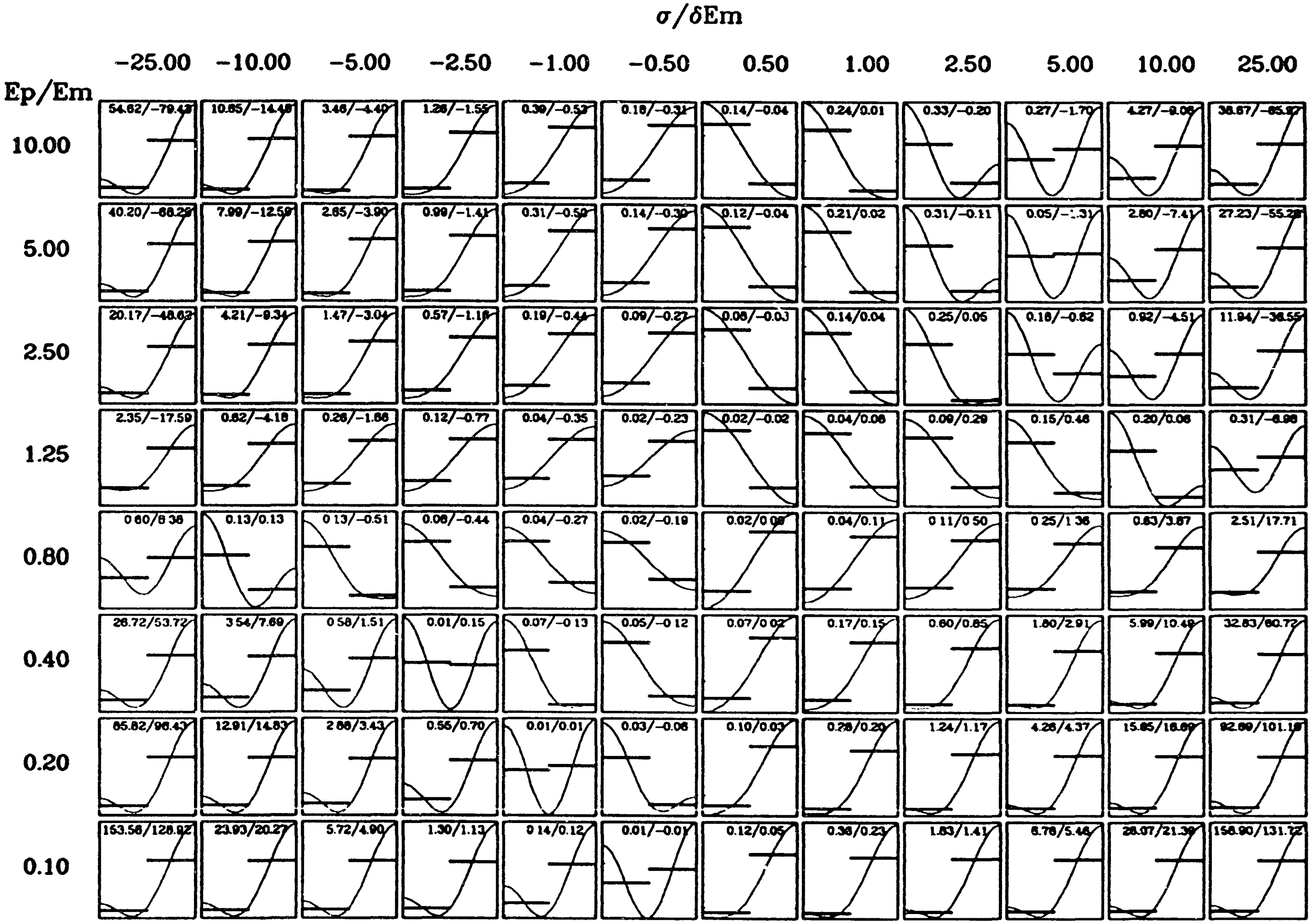


Figure 4.4 Parametric study of τ_n profile for: $\delta = 0.1\%; E_m = 100$ GPa; $0.1 \leq E_p/E_m \leq 10.0$.

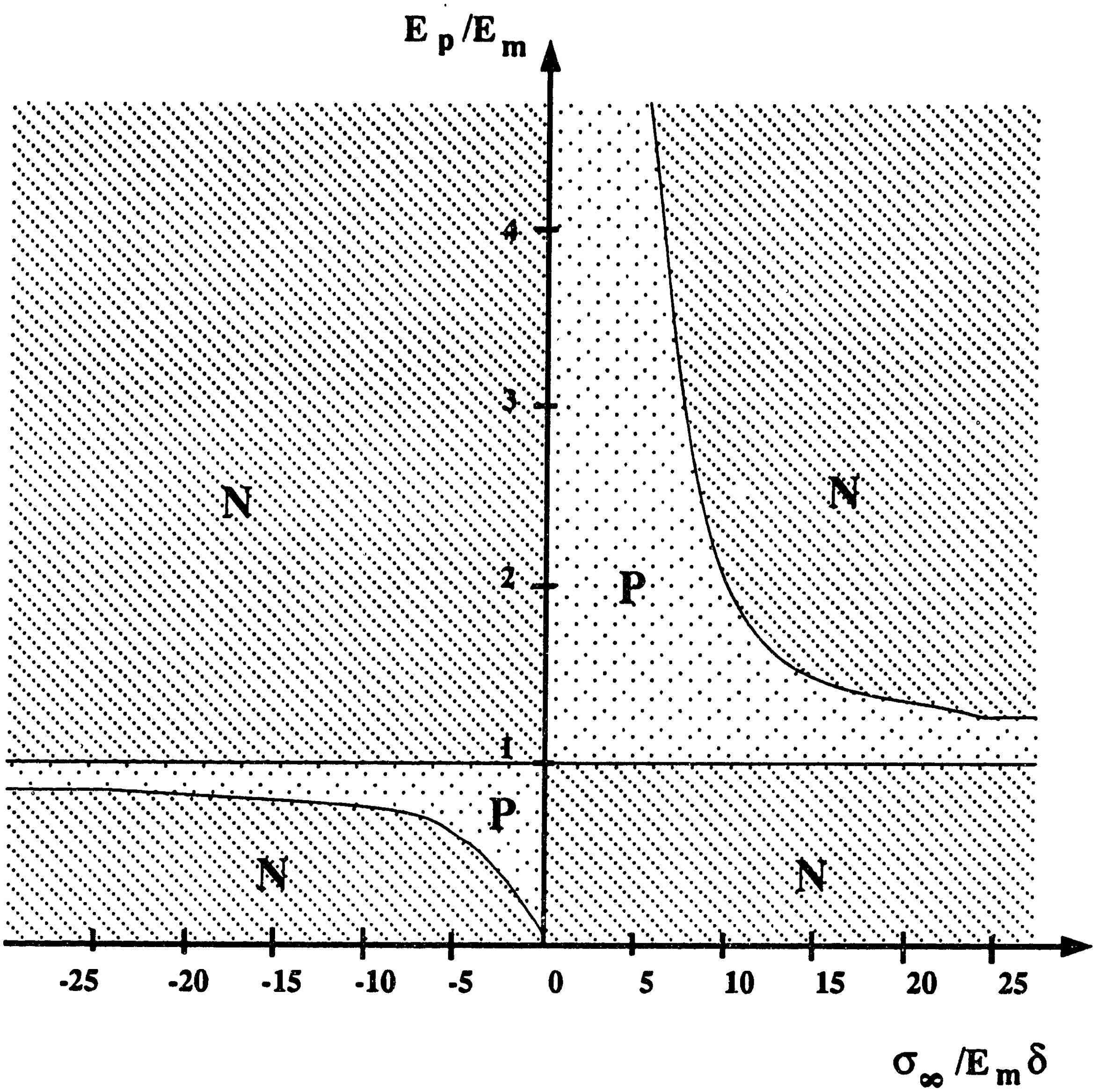


Figure 4.5 Evolution map for an isotropic cylindric inclusion in an infinite matrix.

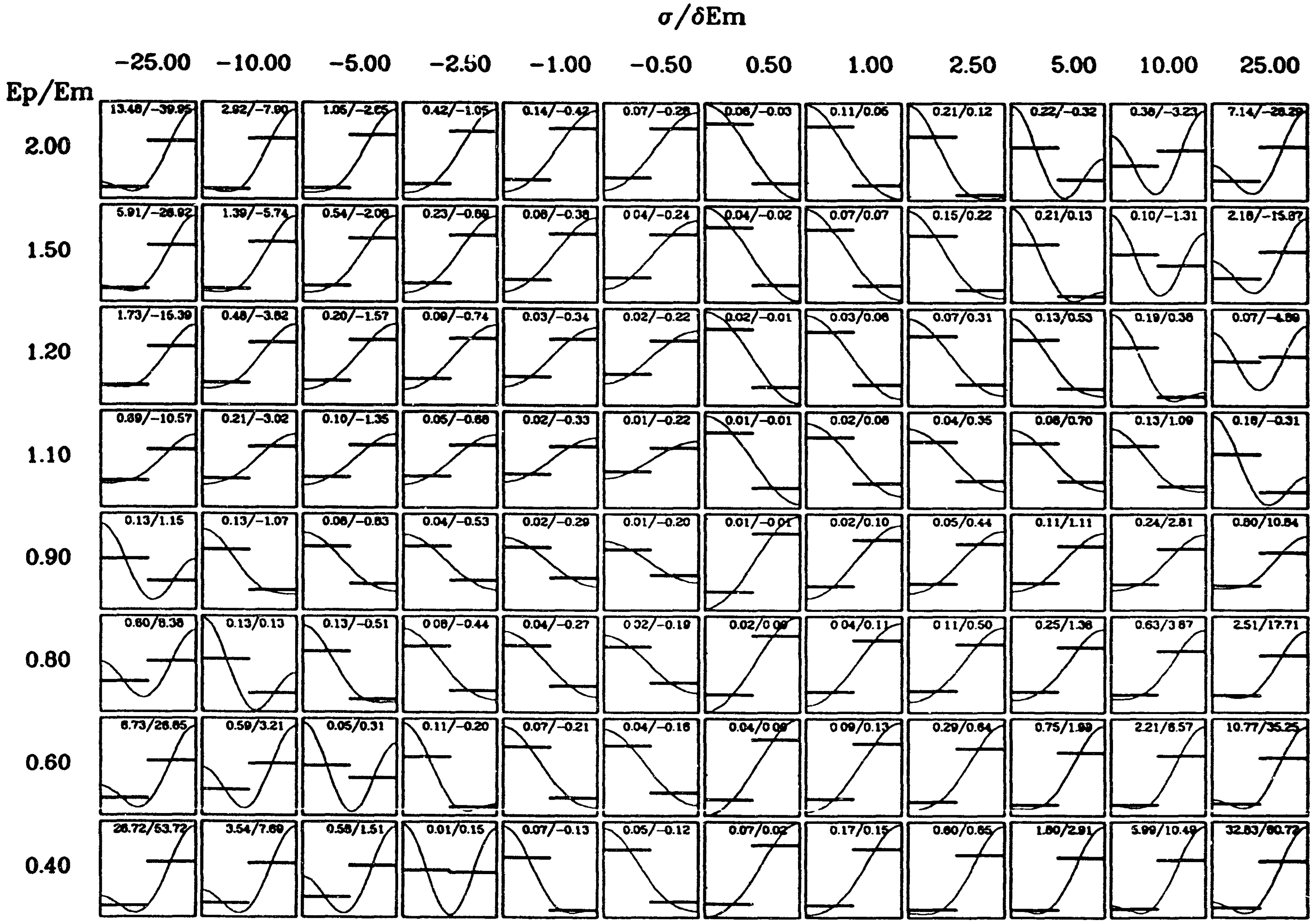


Figure 4.6 Parametric study of τ_n profile for $\delta = -0.1\%; E_m = 100$ GPa; $0.4 \leq E_p/E_m \leq 2.0$.

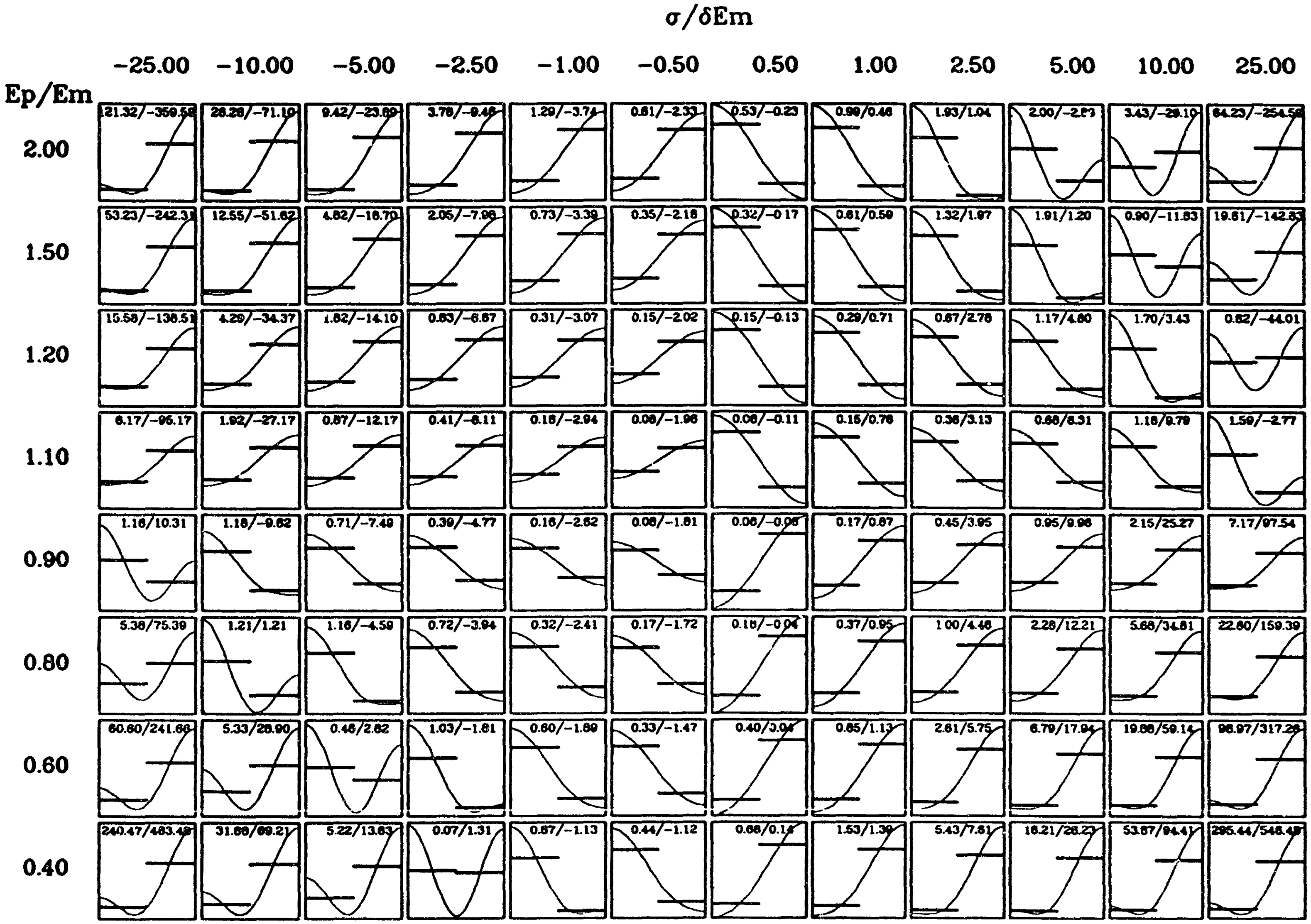


Figure 4.7 Parametric study of τ_n profile for $\delta = 0.3\%$; $E_m = 100$ GPa $0.4 \leq E_p/E_m \leq 2.0$.

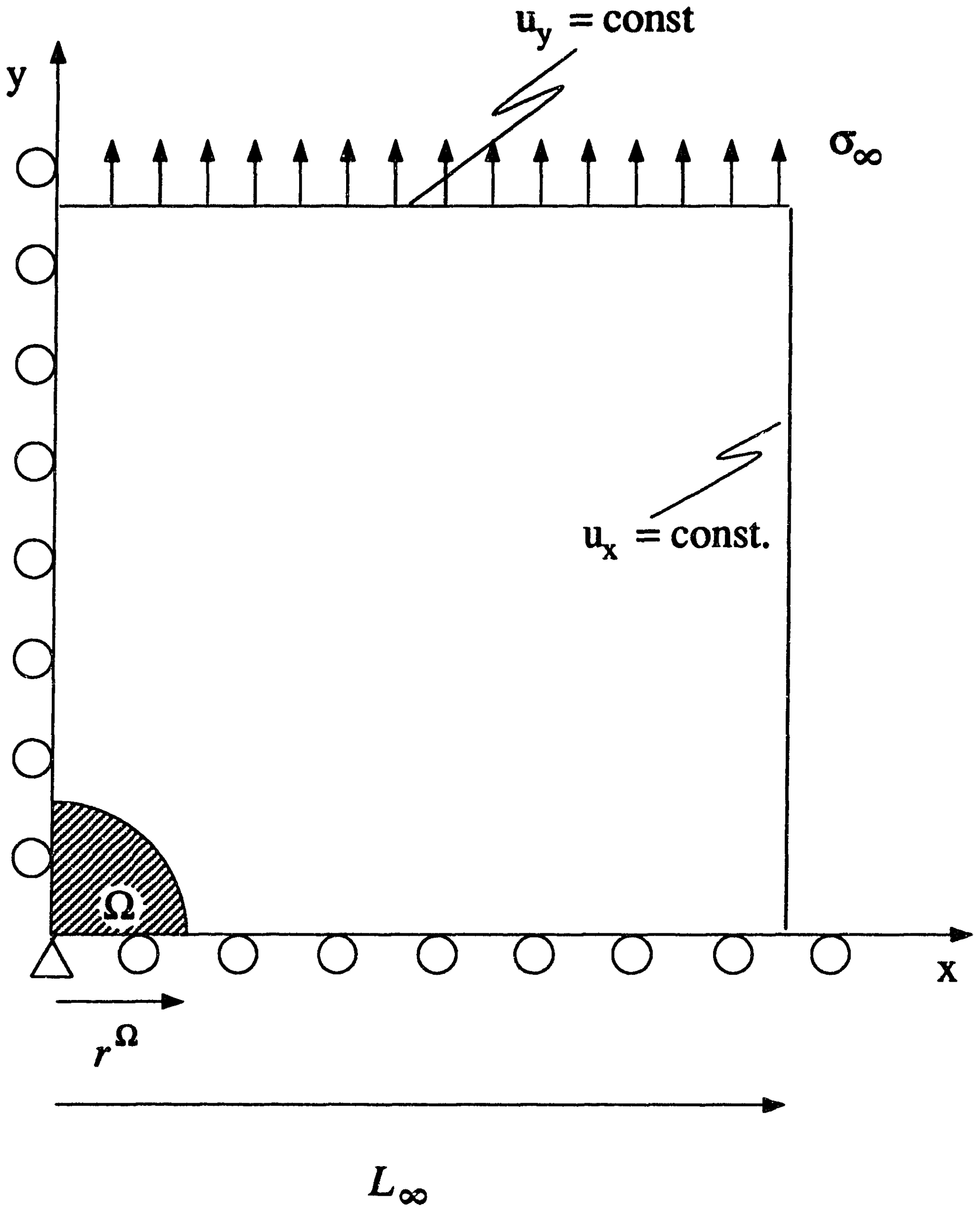


Figure 4.8 Schematic description of the cylindrical inclusion problem modeled, with the boundary conditions indicated $(r^\Omega / L_\infty = 1/20)$.

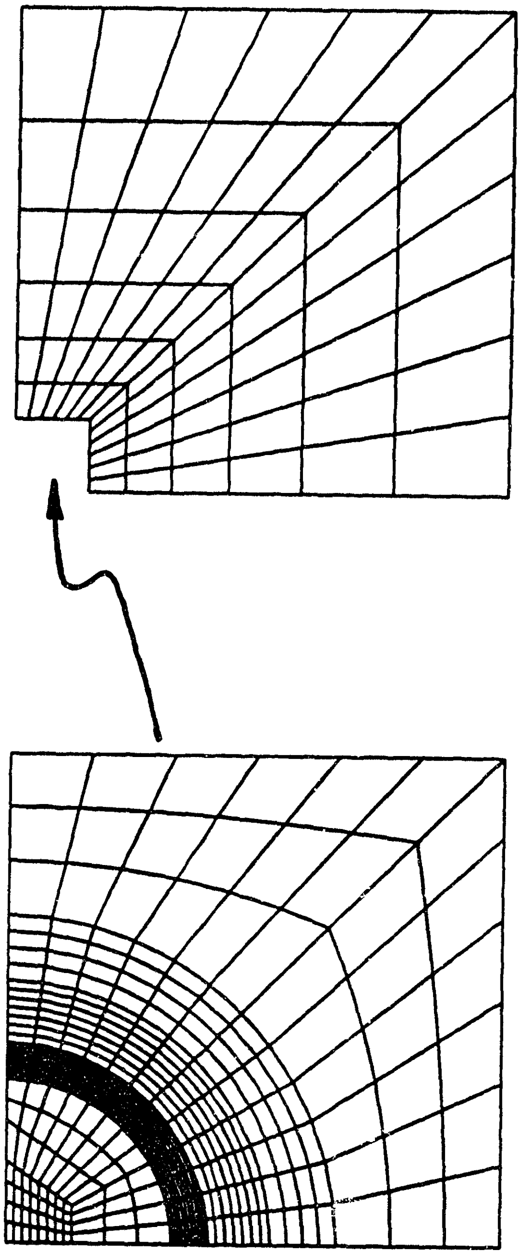


Figure 4.9 Finite element mesh for the cylindrical inclusion problem modeled. In this Figure the bottom mesh ("near-particle" region) fits into the indicated area of the top mesh ("nominal far-field" region).

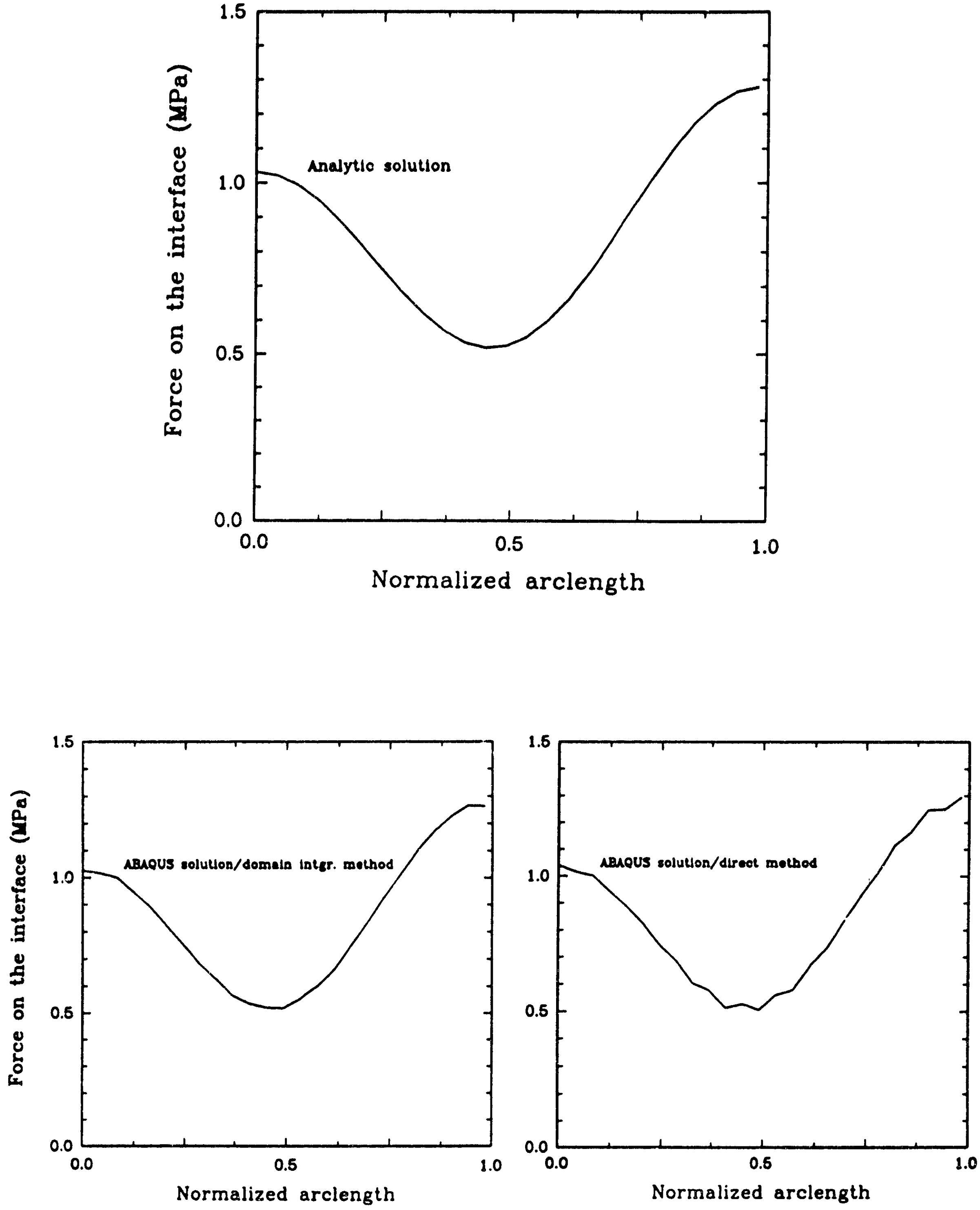


Figure 4.10 Comparison of analytical and numerical results (for $E_m = 200$ GPa; $E_p/E_m = 0.5; \sigma_\infty = 1000$ MPa; $\delta = -0.1\%$).

4.3 Problem Description

In the previous paragraph we have assessed the reliability of the proposed numerical procedures, by direct confrontation of numerical results and analytical solutions in a simple test problem. We can now apply these numerical methods to analyze the morphological evolution of γ' precipitates in Ni-superalloys.

The first step will be to construct a finite element model of the array of cuboidal γ' precipitates. We have simplified the problem by considering a 2-D model using generalized plane strain elements.

These elements permit uniform displacements in the "thickness" direction (normal to the mesh-plane), thus allowing expansion in the third dimension as well as the development of normal stresses and strain in this direction.

Due to the symmetry and periodicity of the microstructure, we can confine our analysis to the "unit cell" depicted in Fig. 4.11(a).

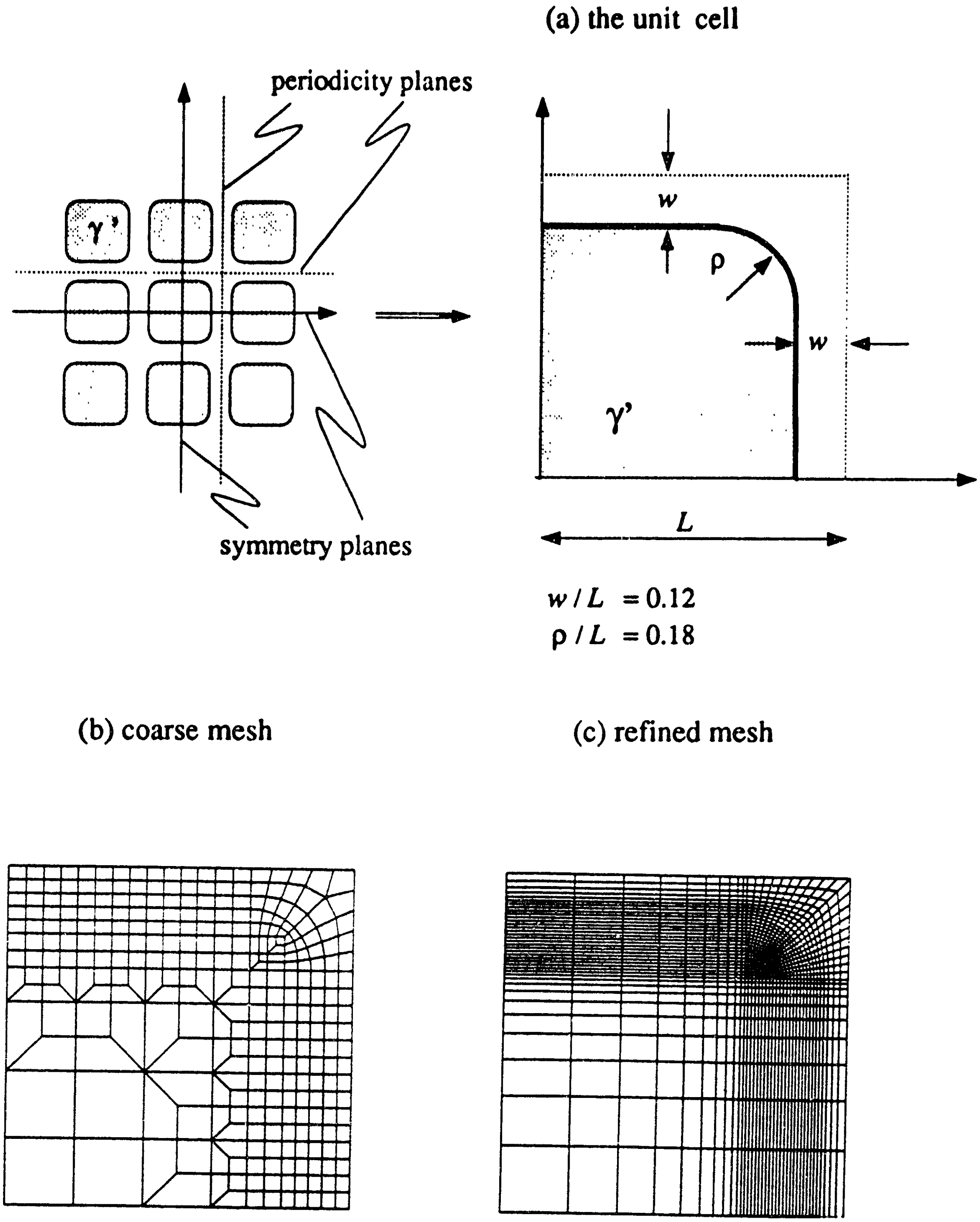
Two different meshes have been generated: a coarse mesh (Fig. 4.11(b)), used in the "tune-up" stage of the research to obtain first order approximations of the τ_n profile with a limited amount of required computational time, and a more refined mesh (Fig. 4.11(c)) for the final calculations.

The level of numerical noise associated with different choices of mesh and/or numerical method can be appreciated by comparing the τ_n profiles in Fig. 4.12.

All of the plots shown in the following paragraphs have been obtained using the refined mesh. Most of them have been obtained by applying the domain integral method. However, as previously mentioned, there exist particular cases in which the level of numerical noise in the domain integral solution is particularly high: if we consider the integral expression for δE^T in Eq. (3.19 (a)), we can see that δE^T depends on the gradient of the transformation strains $\epsilon^T_{mn,i}$. When we will perform the stress-annealing analyses, the transformation strains will be given by the superposition of misfit strains and creep strains ϵ^{creep}. In the very last stage of the analyzed transients, extremely steep ϵ^{creep}-gradients develop around the corners of the γ' precipitates (see Figs 4.34 and 4.38). For these particular conditions the finite element mesh is not refined enough to allow a precise numerical evaluation of the gradient of ϵ^{creep} so that, for these particular cases, the direct method yields more reliable results than the domain integral approach.

In order to perform a finite element analysis of the $\gamma - \gamma'$ crystal, we need the following set of data:

- elastic constants of matrix and precipitate;
- misfit at test temperature;
- loading conditions;



(a) the unit cell

(b) coarse mesh

(c) refined mesh

Figure 4.11 Finite element discretization of the unit cell.

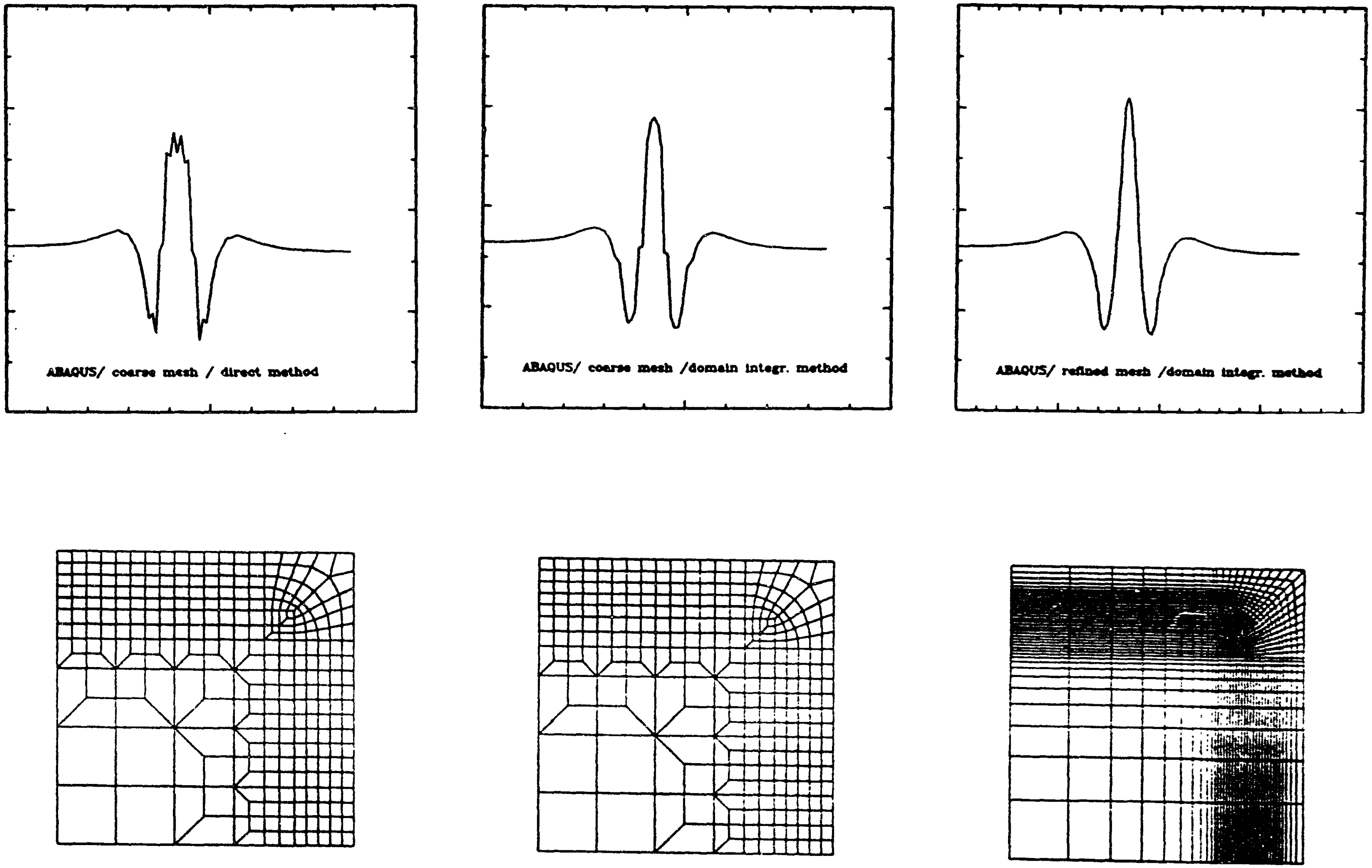


Fig. 4.12 Comparison of τ_n profiles obtained with different choices of mesh and/or .iumerical method.

- creep properties;

- microstructure geometry (dimensions of the γ' cubes and of the γ channels).

While a simulation of actual $\gamma-\gamma'$ crystals is of extreme interest because the results of the analysis can be directly compared with experimental data, we are limited, in this approach, by the extremely exiguous number of well-characterized alloys available in the literature. Values for the elastic constants of the γ and of the γ' phases are available only for three alloys on which rafting experiments have been carried out. These are:

a. The negative-misfit alloy studied by Pollock [25]

b. The alloy studied by Tien and Copley [13]

c. The alloys studied by Miyazaki, Nakamura and Mori [15]

For these alloys the loading conditions and the test temperature are available as well; also the value of the misfit at test temperature is given (alloys (a) and (c)) or can be evaluated (alloy (b): see the end of paragraph 1.2). In Table 4.1 we have summarized the characteristics of these alloys, together with the rafting behavior observed in the experiments. Thus we have only six cases, to which we have assigned an acronym that we will use to identify each case, that we can use to compare our results with experimental data.

In order to explore a wider range of possibilities, we have "created" a few *hypothetical* alloys by altering some of the characteristics of the alloys of Table 4.1.

Altering the characteristics of the alloy studied by Miyazaki Nakamura and Mori, we have "created" an alloy with positive misfit and soft precipitates by switching the elastic constants of the two phases, and an alloy with negative misfit and hard precipitates by changing the sign of the misfit.

For the alloy studied by Tien and Copley, we have explored the effect of assuming the value of the misfit reported in the paper: in this way we have studied the behavior of an alloy with soft particle and small positive misfit. Finally, we have explored the effect of varying the crystal symmetry by considering for both phases isotropic crystal lattices with elastic constants *equivalent* to those of the cubic lattices of the alloy studied by Pollock; we have evaluated an equivalent Young's modulus as

$$E_{eq} = \frac{(C_{11} - C_{12})(3 + (C_{11} - C_{12})/C_{12})}{2 + (C_{11} - C_{12})/C_{12}} \tag{4.1}$$

and an equivalent Poisson's ratio as

$$\nu_{eq} = \frac{1}{2 + (C_{11} - C_{12})/C_{12}} \tag{4.2}$$

where C_{11} and C_{12} are the elastic constants of the cubic lattice [58].

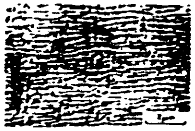
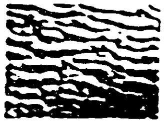
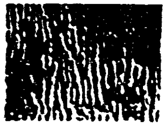
researchers	crystal symmetry	elastic constants of the γ'-phase	elastic constants of the γ-phase	misfit	applied load (along $< 001 >$)	type of observed rafting behavior	case no.	case acronym
Miyazaki, Nakamura, Mori [16]	cubic	$C_{11} = 167$ GPa $C_{12} = 106$ GPa $C_{44} = 99$ GPa	$C_{11} = 112$ GPa $C_{12} = 63$ GPa $C_{44} = 57$ GPa	+0.56%	+ 147 MPa	P	1	MNMT
					- 147 MPa	N 	2	MNMC
Tien, Copley [13]	isotropic	$E = 144$ GPa $\nu = 0.3$	$E = 158$ GPa $\nu = 0.3$	−0.30%	+ 155 MPa	N 	3	TCT
					- 155 MPa	P 	4	TCC
Pollock [25]	cubic	$C_{11} = 179$ GPa $C_{12} = 120$ GPa $C_{44} = 88$ GPa	$C_{11} = 202$ GPa $C_{12} = 139$ GPa $C_{44} = 95$ GPa	−0.38%	+ 50 MPa	N	5	PLT
				−0.39%	+ 138 MPa	N	6	PHT

Table 4.1 Characteristics of experimental alloys and observed rafting behavior.

notes	crystal symmetry	elastic constants of the γ'-phase	elastic constants of the γ-phase	misfit	applied load (along $< 001 >$)	case no.	case acronym
Same as MNMT case, but with the elastic constants of the γ' and γ phases inverted (soft precipitate)	cubic	$C_{11} = 112$ GPa $C_{12} = 63$ GPa $C_{44} = 57$ GPa	$C_{11} = 167$ GPa $C_{12} = 106$ GPa $C_{44} = 99$ GPa	+0.56%	+ 147 MPa	7	MNMT_{INV}
Same as MNMT case, but with negative misfit	cubic	$C_{11} = 167$ GPa $C_{12} = 106$ GPa $C_{44} = 99$ GPa	$C_{11} = 112$ GPa $C_{12} = 63$ GPa $C_{44} = 57$ GPa	−0.56%	+ 147 MPa	8	MNMT_{NMS}
Same as TCT,TCC cases, but with the positive value of misfit given in [13]	isotropic	$E = 144$ GPa $\nu = 0.3$	$E = 158$ GPa $\nu = 0.3$	+0.02%	+ 155 MPa	9	TCT_{PMS}
					- 155 MPa	10	TCC_{PMS}
Same as the PLT case, but with equivalent isotropic crystal symmetry	isotropic	$E = 82$ GPa $\nu = 0.3$	$E = 89$ GPa $\nu = 0.3$	−0.38%	+ 50 MPa	11	PLT_{ISO}

Table 4.2 Characteristics of hypothetical alloys examined in the rafting analysis.

The characteristics of these hypothetical alloys and the loading conditions for which the simulations have been carried out are summarized in Table 4.2

Regarding the creep properties, in the range of loading conditions that will be explored, the predominant creep mechanism involves dislocation motion primarily in the γ-matrix. TEM studies [25] show that the γ' cuboids are not sheared by dislocations until the latest stage of steady state creep. Thus in our model of the very first stage of the stress-annealing transient, the γ' precipitate is allowed to deform only elastically while the matrix material can deform by creep.

Since the < 001 > orientation of the applied load is a multiple slip orientation, a simple isotropic power-law relation has been used to characterize the creep properties of the matrix:

$$\dot{\epsilon}^{cr} = A\sigma^{n}. \tag{4.3}$$

In all our stress annealing transients the creep properties given by Johnson et al., [56] for Ni-6W tested at 854° have been used, with:

$$n = 4.8$$

$$A = 5.4\mathrm{E} - 15(s^{-1}\mathrm{MPa}^{-4.8})$$

Actually, since we are not particularly interested in "real time" simulation of the stress-annealing transient, the accuracy of the pre-exponential coefficient A, is of secondary relevance; the stress exponent, n, is a more important quantity but it should not exhibit significant modifications when the alloying of the γ phase is slightly modified.

Regarding the microstructure geometry, we have relied on the data obtained by Pollock [25] by TEM study of CMSX3. This choice will not yield an accurate simulation of the Tien and Copley and of the Miyazaki-Nakamura-Mori alloys which are characterized by a lower volume fraction of the γ' phase. However, we believe that this inaccuracy will not affect the basic pattern of the results. This point should be, however, proved by further simulations in which the γ' volume fraction is modified.

According to Pollock's data, the γ' precipitates have an average cube length of 0.45 μm and the matrix passages have an average thickness of 60 nm, giving a γ' volume fraction of 0.68. The finite element models shown in Fig. 4.11 have been formulated so as to preserve these dimensions, so that since a generalized plane strain analysis is done, the volume fraction of the γ' in the model is somewhat higher than in the real material because the matrix channels parallel to the plane of the mesh are neglected. While we are aware of the fact that our models still present a certain degree of inaccuracy, and that the eleven cases that we will analyze do not constitute a complete parametric study, we will see in the following paragraphs how the results of our simulations fit in a logical pattern that can yield significant insight to the rafting phenomenon.

4.4 Elastic Analysis

An elastic analysis for the eleven cases characterized in Tables 4.1 and 4.2, has been carried out. Both the matrix and the precipitate are only allowed to deform elastically and coherency at the matrix-precipitate interface is enforced. A typical ABAQUS input file for this type of analysis is listed in Appendix V. The particular file listed is relative to case 1 (MNMT), but all the other cases can be easily obtained by correcting the values of misfit, elastic constants and applied load, according to the data of Tables 4.1 and 4.2.

The results of the elastic analyses have been postprocessed to obtain the distribution of the force on the interface. Profiles of τ_n for each of the eleven cases as shown in Figs. 4.13 through 4.23. A synopsis of the results is given in Table 4.3.

We can notice observing the τ_n profiles, that the value of the difference of driving force on the top and on the side of the particle is always very limited: at most it is of the order of 0.3 MPa for the "MNM" - cases and it is as low as 0.006 MPa for the TCT_{PMS} and TCC_{PMS} cases.

If we evaluate the "Pineau's parameters" E_p/E_m and $\sigma/E_m\delta$ for all the eleven cases (for the cubic crystals an equivalent Young's modulus can be calculated using Eq. (4.1)), we can see how the type of rafting behavior indicated by the τ_n profiles is perfectly consistent with the behavior predicted by our simple evolution map of Fig. 4.5.

This suggests that a very similar map could be constructed by performing a parametric study on the actual $\gamma - \gamma'$ morphology.

Table 4.3 Synopsis of the Elastic Analysis

Case #	Case Acronym	E_p/E_m	$\sigma/E_m\delta$	Rafting behavior predicted by the model	Consistency with the evolution map of Fig. 4.5	Consistency with experimental results
1	MNMT	1.23	0.388	P	YES	YES
2	MNMC	1.23	-0.388	N	YES	YES
3	TCT	0.91	-0.327	P	YES	NO
4	TCC	0.91	+0.327	N	YES	NO
5	PLT	0.92	-0.148	P	YES	NO
6	PHT	0.92	-0.399	P	YES	NO
7	MNMT_{INV}	0.81	+0.314	N	YES	
8	MNMT_{NMS}	1.23	-0.388	N	YES	
9	TCT_{PMS}	0.91	+4.9	N	YES	
10	TCC_{PMS}	0.91	-4.9	P	YES	
11	PLT_{ISO}	0.92	-0.148	P	YES	

Unfortunately, such a map would be totally inadequate to predict the rafting behavior of the crystals since, as indicated in Table 4.3, our predictions based on elastic calculation of the τ_n profile are not always in agreement with the available experimental data.

This leads us toward a more accurate simulation of the actual conditions in which the rafting behavior is observed, as will be discussed in the next paragraph.

It is interesting to note that if we assume, for the alloy studied by Tien and Copley, the erroneous value of the misfit given in the paper, the rafting behavior predicted by our evolution map, and by the τ_n profiles, is consistent with the experimental observations.

This unfortunate contingency has represented, in the past, a confusing source of misplaced confidence in the methods based on purely elastic calculations.

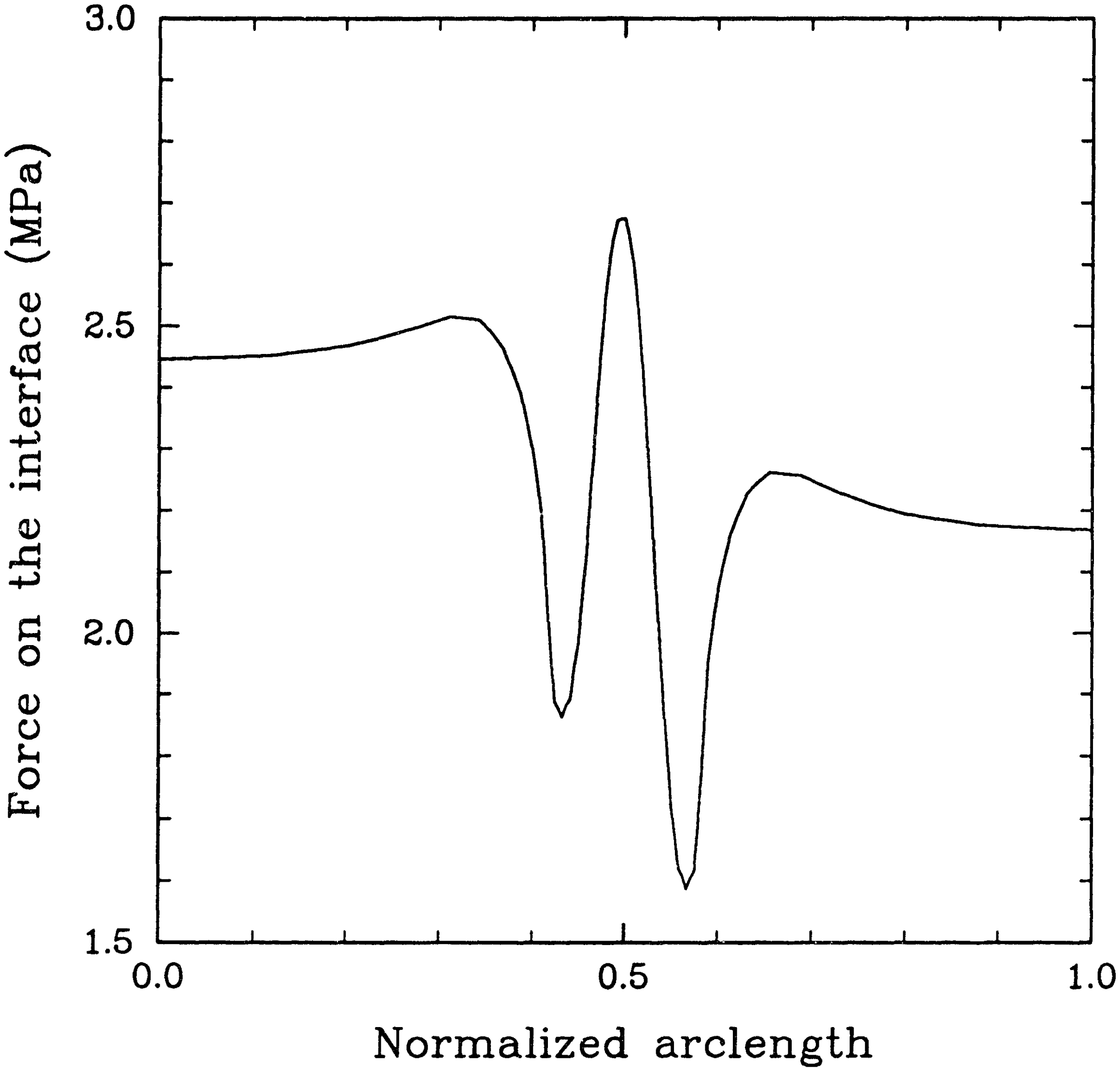


Figure 4.13 τ_n profile for the elastic analysis of case 1 (MNMT).

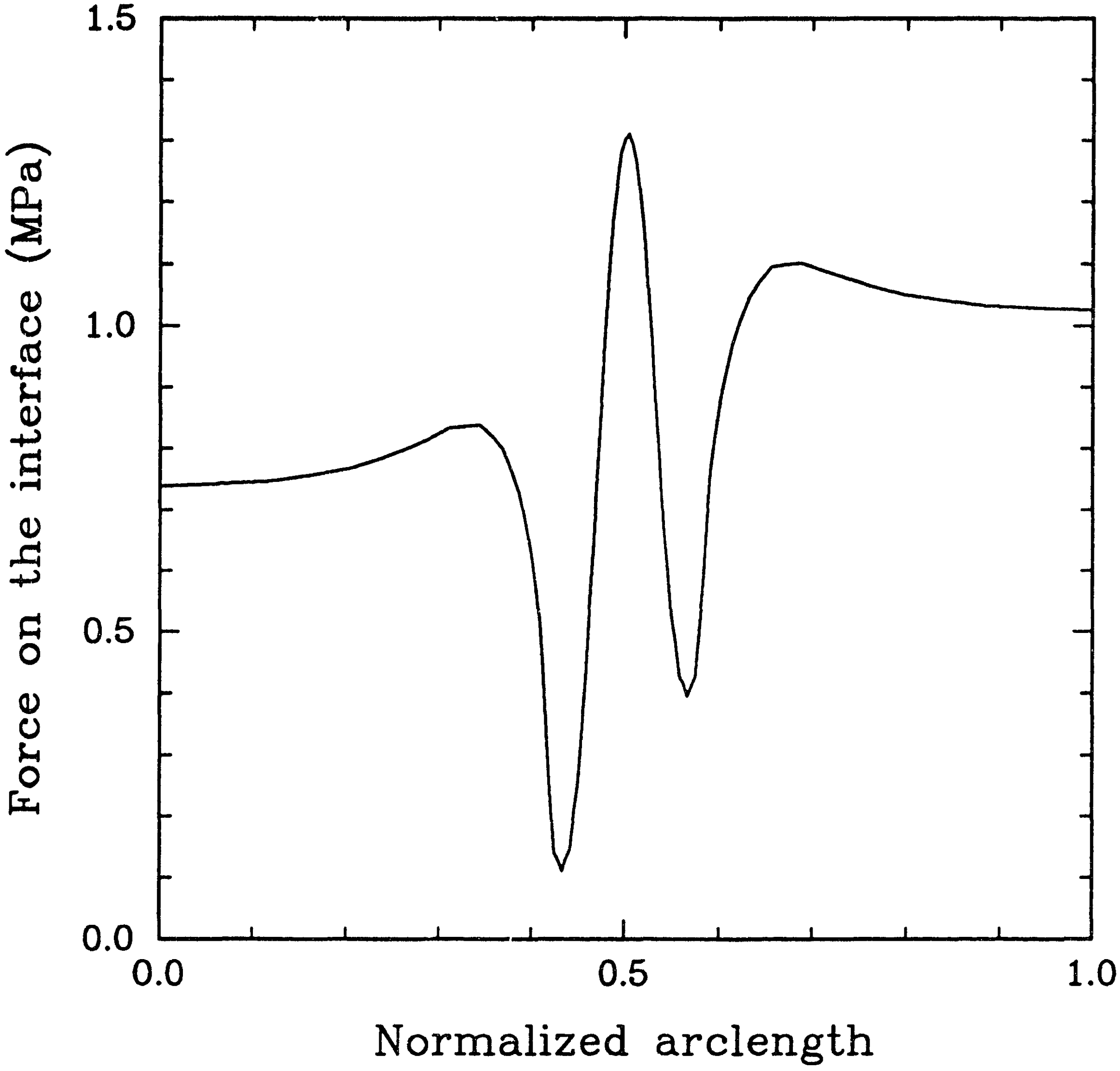


Figure 4.14 τ_n profile for the elastic analysis of case 2 (MNMC).

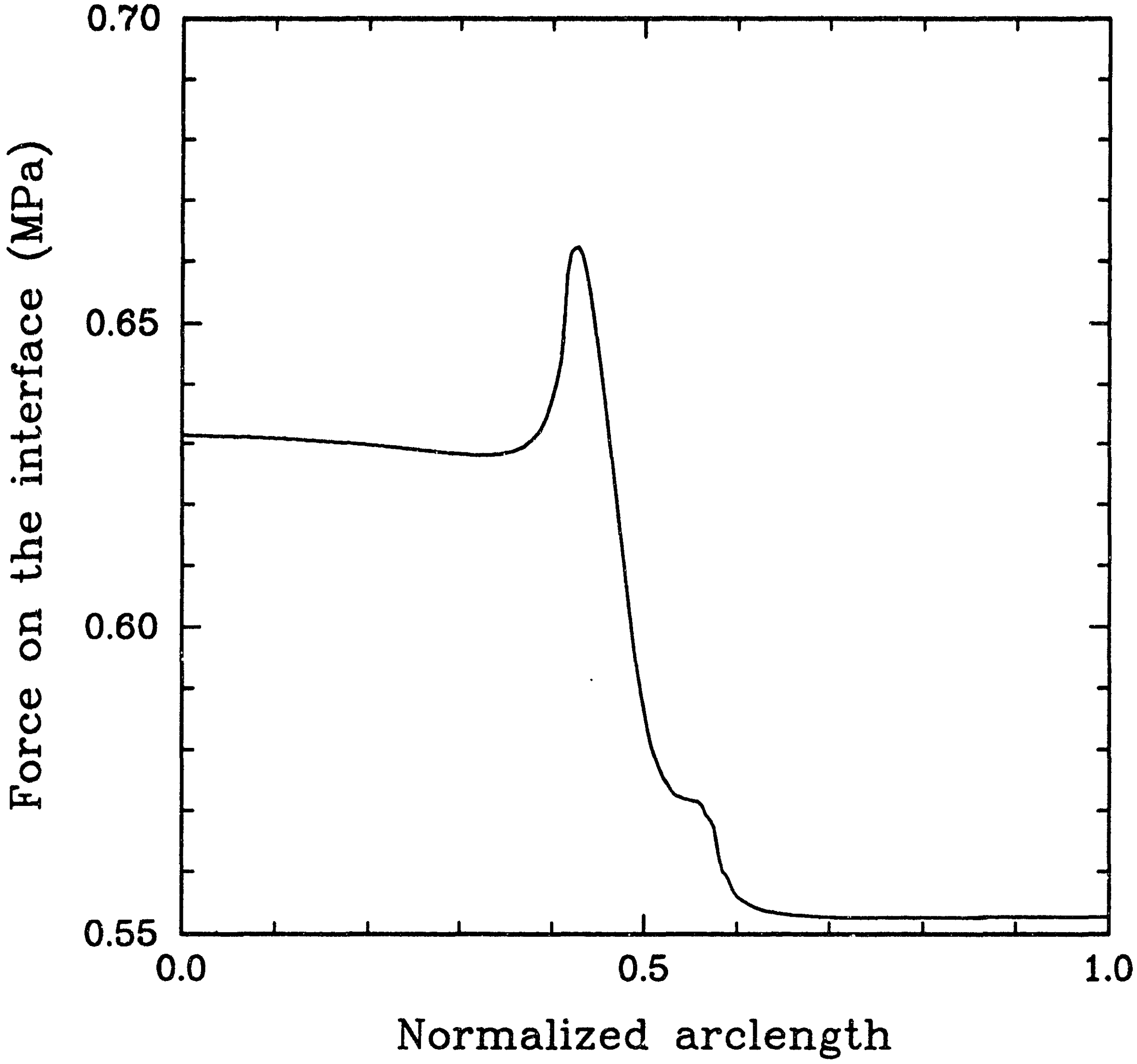


Figure 4.15 τ_n profile for the elastic analysis of case 3 (TCT).

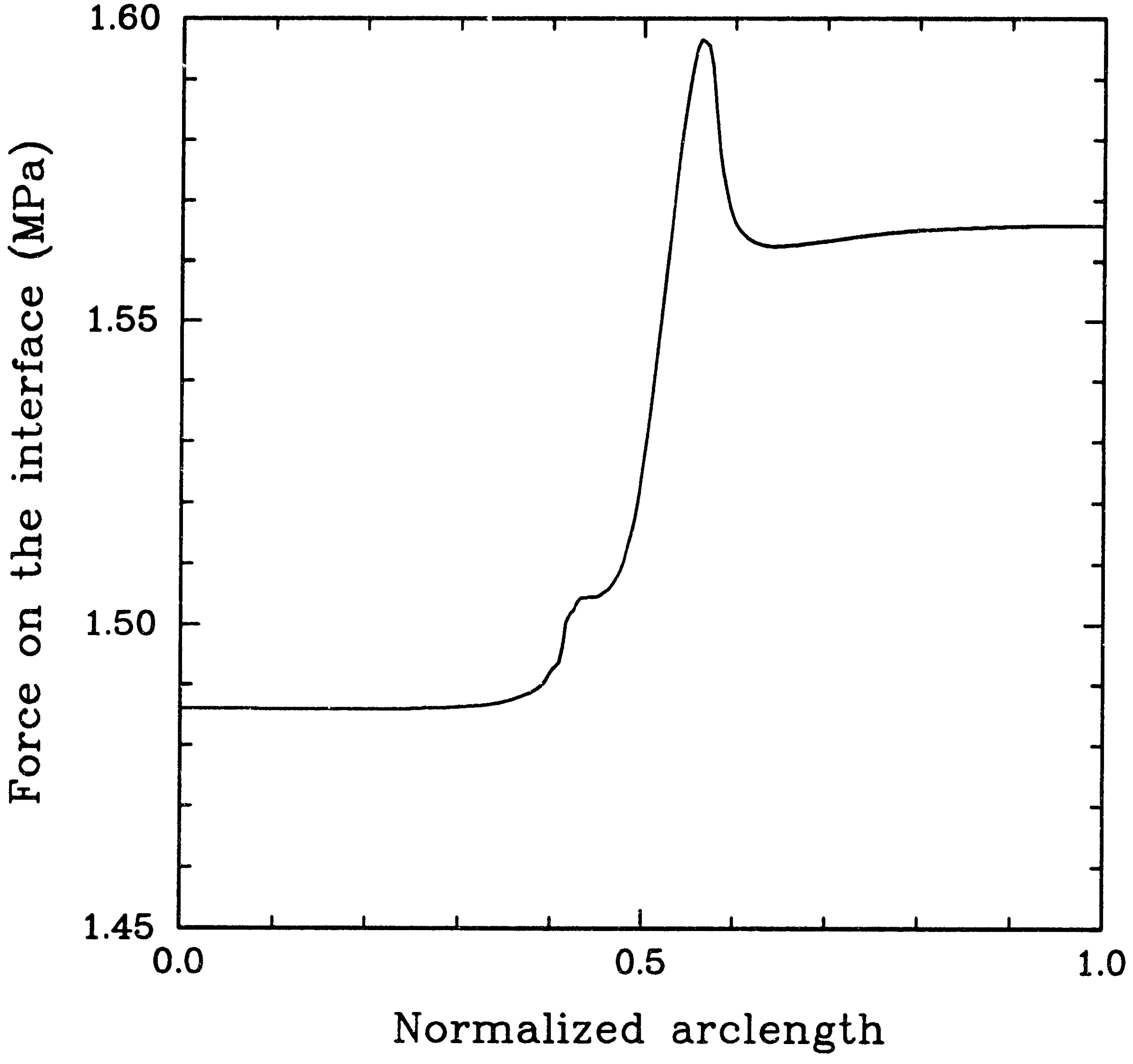


Figure 4.16 τ_n profile for the elastic analysis of case 4 (TCC).

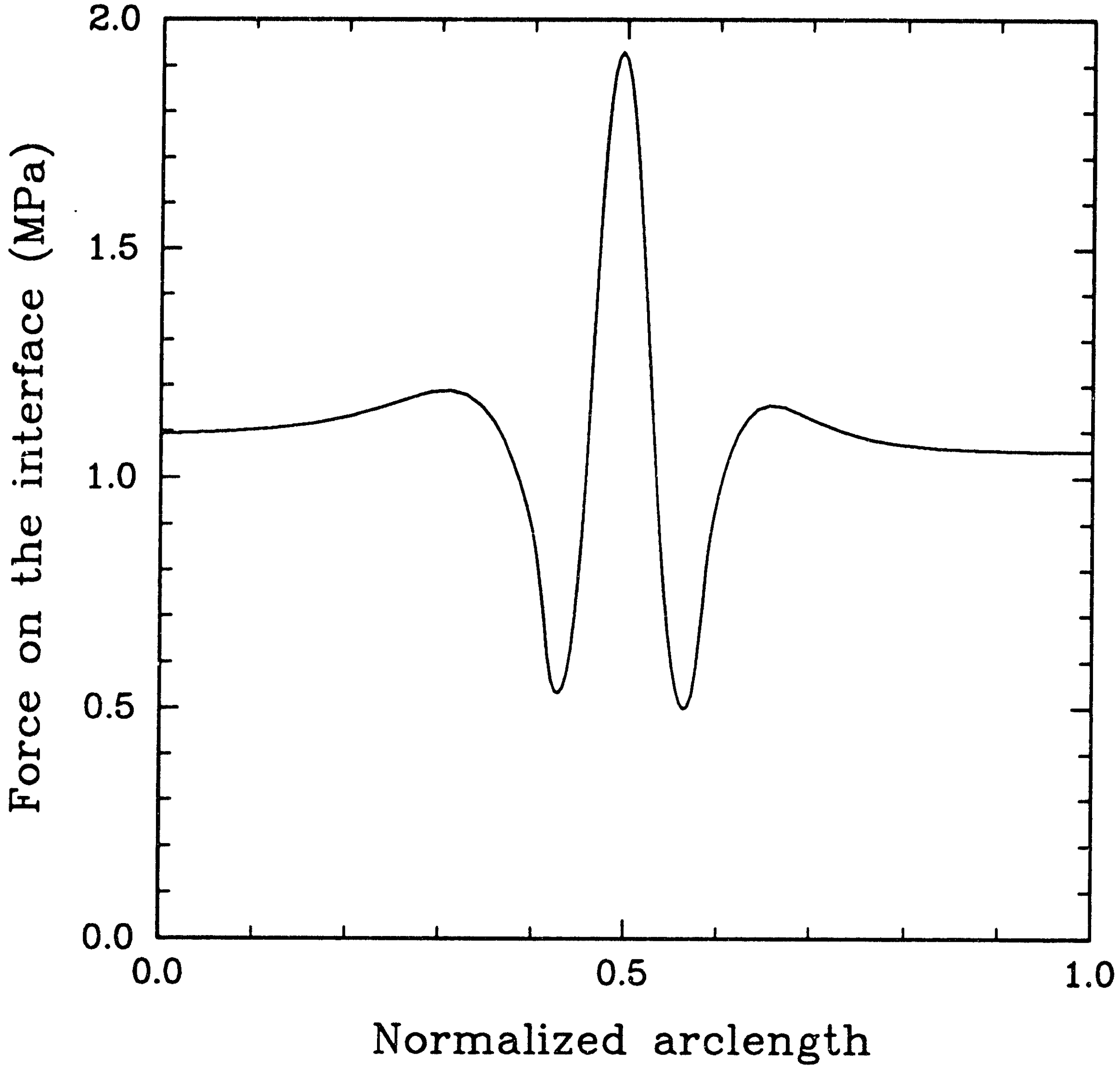


Figure 4.17 τ_n profile for the elastic analysis of case 5 (PLT).

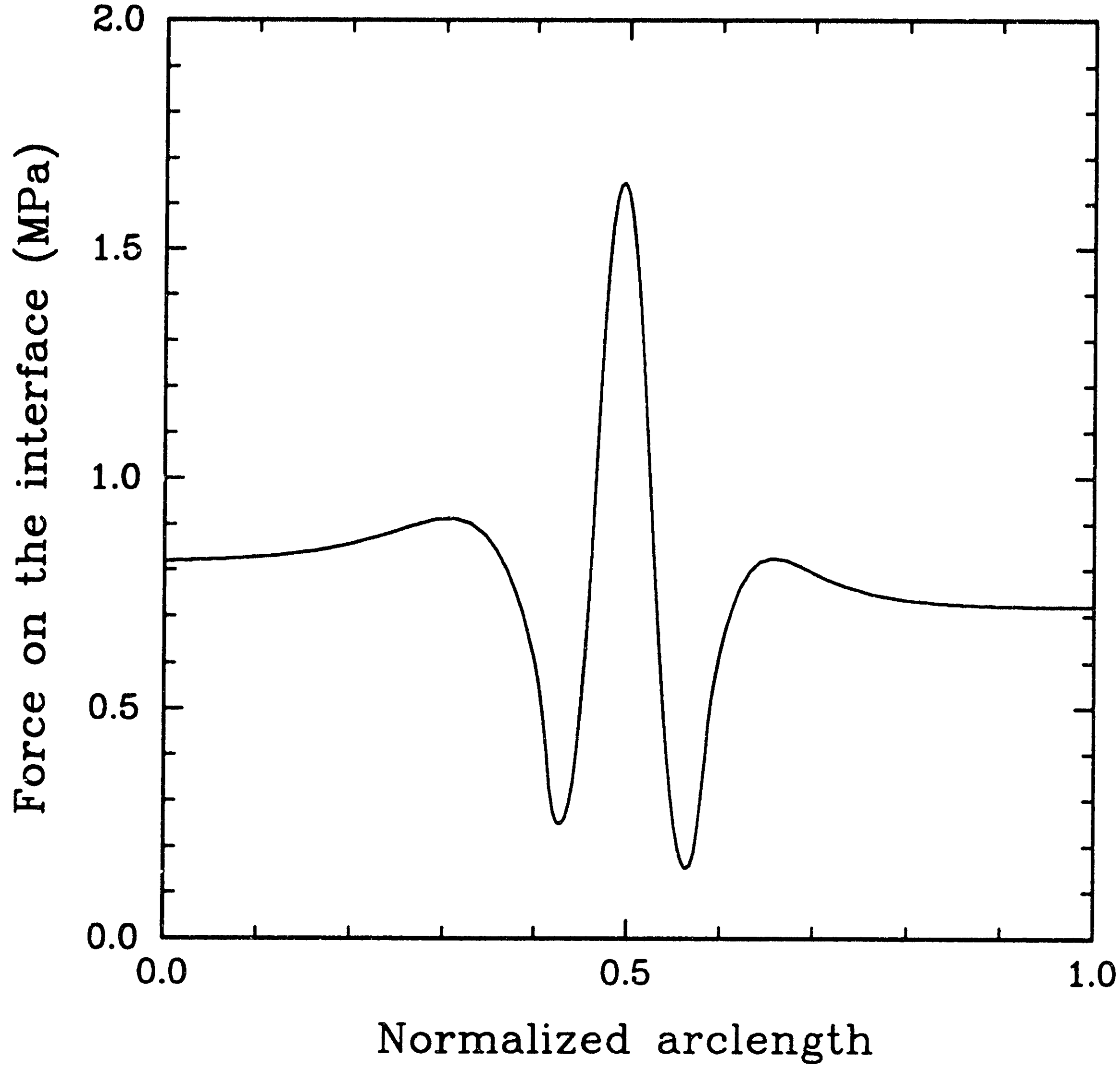


Figure 4.18 τ_n profile for the elastic analysis of case 6 (PHT).

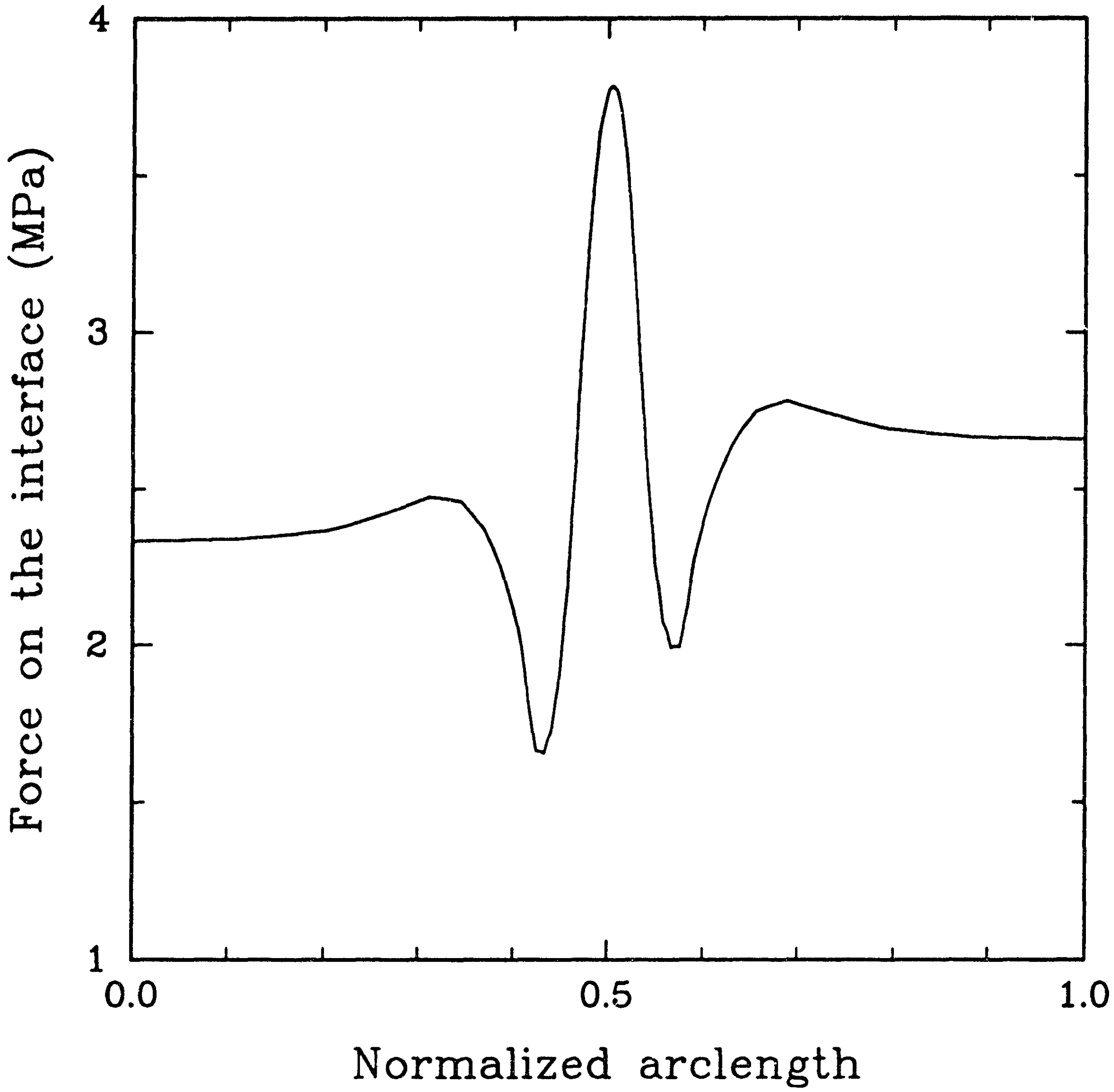


Figure 4.19 τ_n profile for the elastic analysis of case 7 (MNM_{INV}).

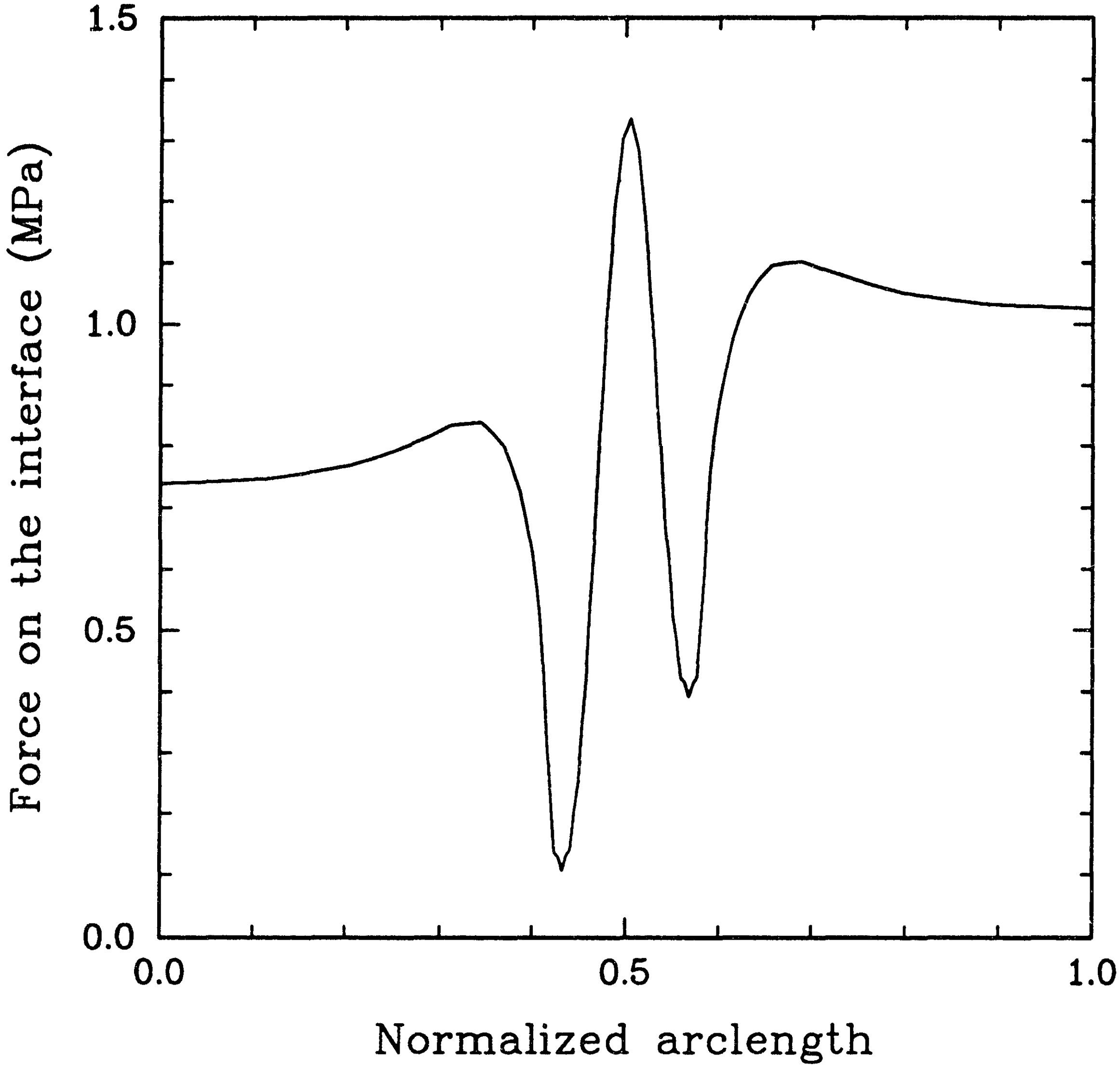


Figure 4.20 τ_n profile for the elastic analysis of case 8 (MNMT$_{NMS}$).

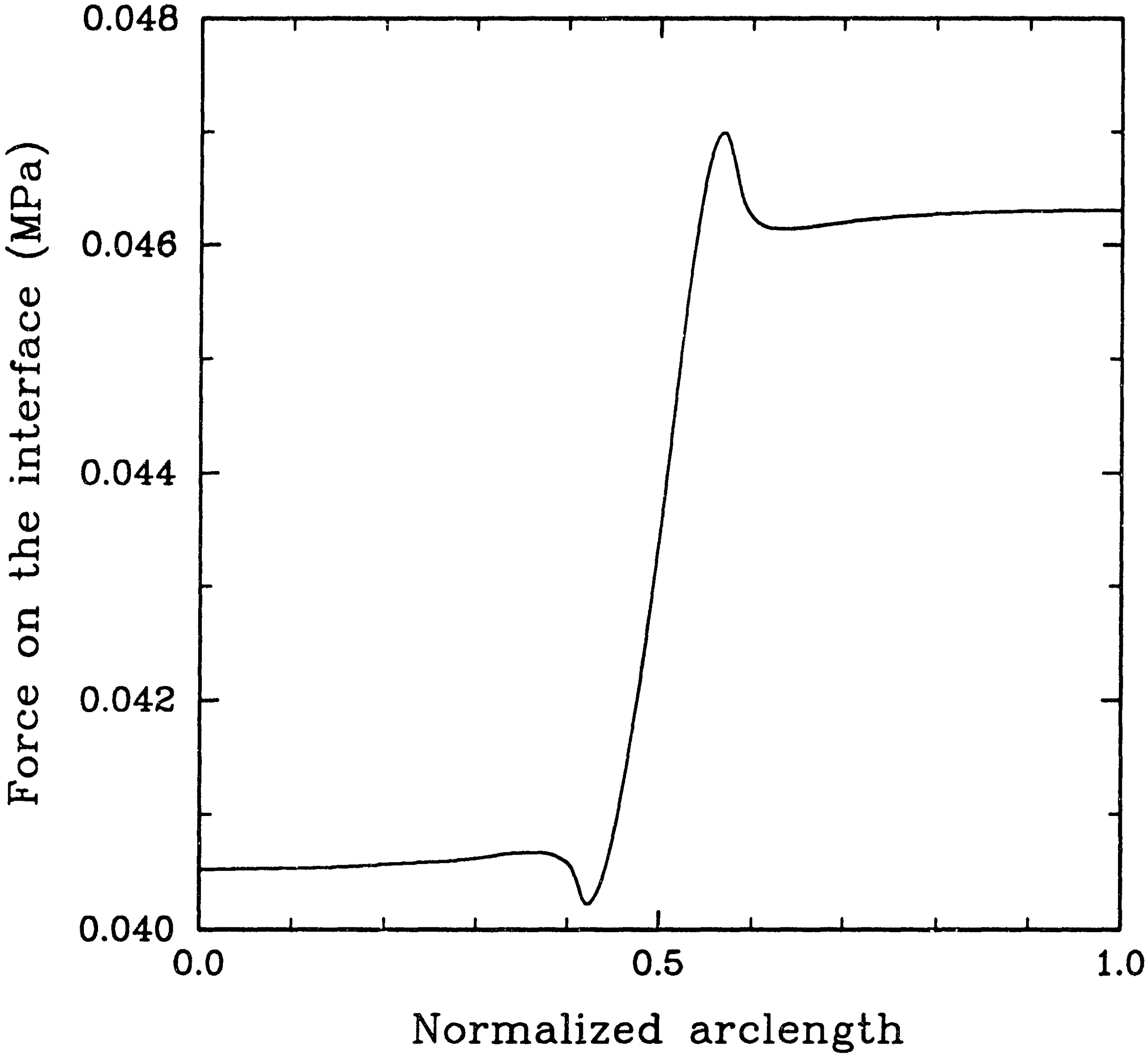


Figure 4.21 τ_n profile for the elastic analysis of case 9 (TCT_{PMS}).

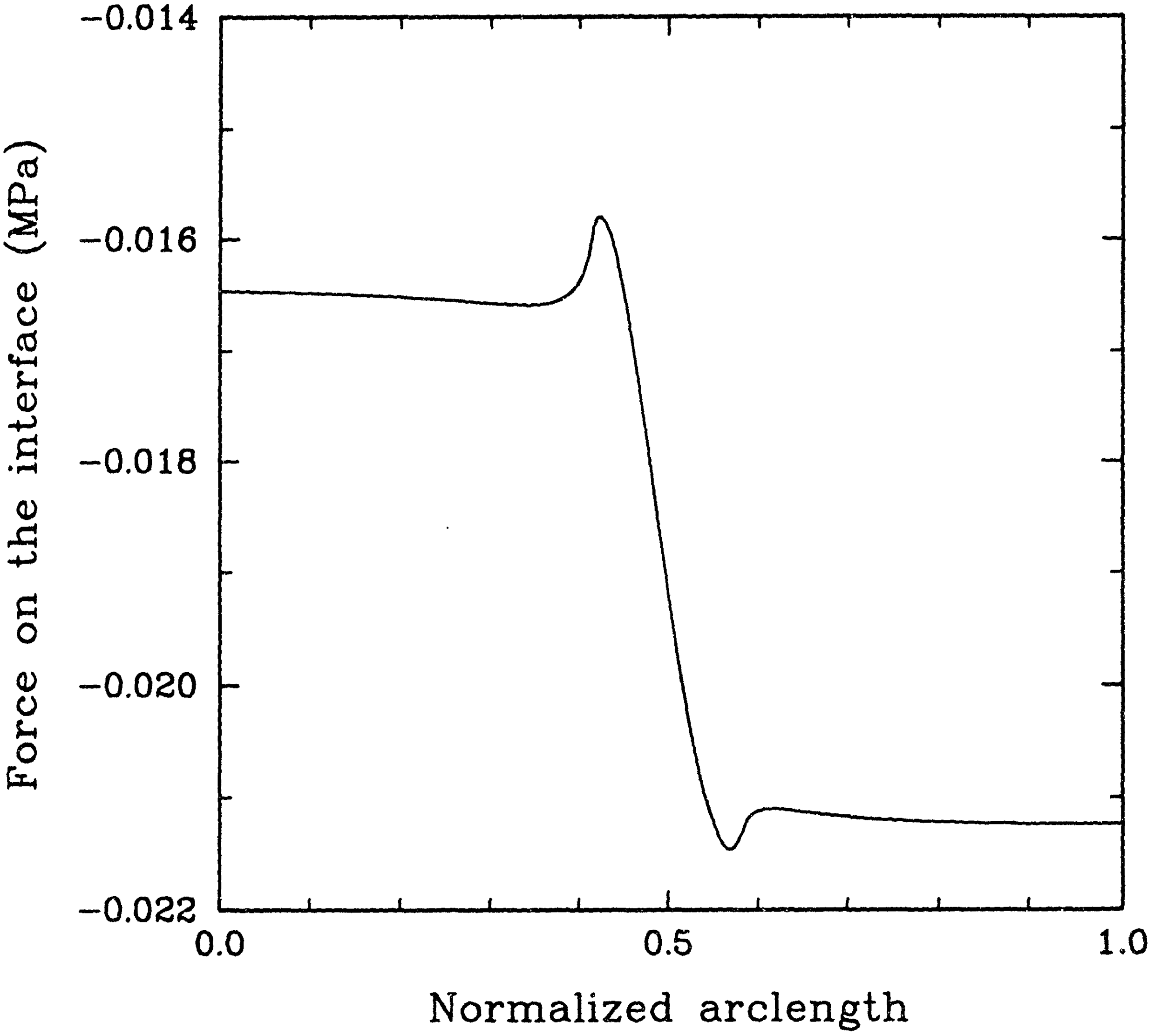


Figure 4.22 τ_n profile for the elastic analysis of case 10 (TCC$_{PMS}$).

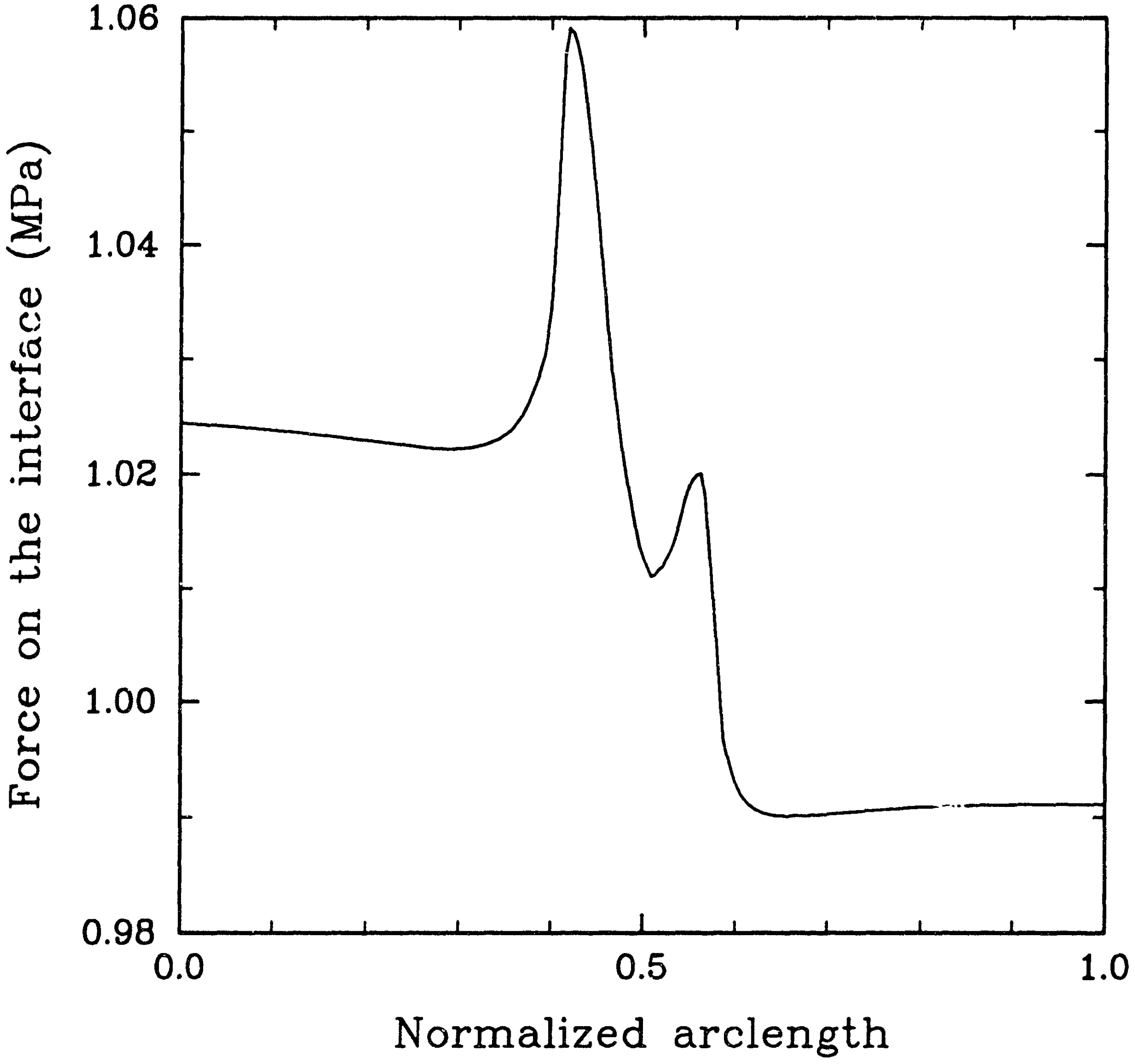


Figure 4.23 τ_n profile for the elastic analysis of case 11 (PLT_{ISO}).

4.5 Analysis of the Stress-Annealing Transients

As previously discussed in paragraph 1.2, TEM observations of $\gamma-\gamma'$ crystals show how, in the very first stage of stress-annealing tests, mobile dislocations start to form and multiply in the γ matrix. As soon as the crystal is brought to test temperature, networks of dislocations form at the $\gamma - \gamma'$ interfaces, wrapping the γ' cuboids and relieving most of the shear component of the misfit strain. When an external load is applied, dislocations glide and climb in the narrow γ channels so that a macroscopic creep flow in the γ matrix is observed.

As a result of this process, the elastic fields in the crystal are dramatically modified. Therefore a meaningful evaluation of the force on the interface should be based on these altered fields.

Even if, considering the scale of the microstructure, an accurate simulation of the process should be based on a model capable to treat discrete dislocations, some interesting results can be obtained by considering a simplified macroscopic creep model for the γ phase. As we have already mentioned in paragraph 4.3, we have modeled the creep behavior of the matrix with the simple isotropic power-law relation (4.3).

We have carried out a simulation of the very first stage of the stress annealing transients, using the ABAQUS code, and analyzed the corresponding evolution of the τ_n profile.

A typical ABAQUS input file for this kind of analysis is listed in Appendix VI. The particular file listed corresponds to case 1 (MNMT).

As *initial* conditions for the simulated transients we have considered the elastic field in the crystal associated with the misfit strain at test temperature, enforcing perfect coherency at the interfaces. We have then applied the test load, over a short linear ramp of one second, and let the matrix creep according to equation 4.3.

Typical contour plots of the *initial* stress state, due to the misfit only, are shown in Figs. 4.24 through 4.29 for a positive-misfit alloy (Miyazaki-Nakamura-Mori alloy: Cases 1,2,7) and for a negative-misfit alloy (Pollock alloy: Case 5).

As schematically indicated in Fig. 4.30(a), the main components of stress in the matrix, σ_0 in the Figure, are in the planes of the channels (tensile for positive-misfit alloys, compressive for negative-misfit alloys). The magnitude of the biaxial stress σ_0 in the γ channels varies with the misfit and the elastic constants of the two phases but typically is of the order of 400 - 500 MPa in most commercial alloys. The component of stress normal to the planes of the channels is one order of magnitude smaller and of opposite sign. The γ' precipitates are almost in a state of hydrostatic stress of the same magnitude as the normal stress in the γ channels.

Thus in the very first stage of a stress-annealing transient with an applied stress, σ, in the range 50-200 MPa, the effect of the misfit stresses dominates over the effect of the applied load, and the γ matrix creeps so as to accommodate the misfit (Fig.

4.30(b)). This stage corresponds to the formation of dislocation networks at the $\gamma - \gamma'$ interfaces.

In Figs. 4.31 through 4.38, a sequence of contour plots of the equivalent creep strain in the matrix, ϵ_{eq}^{creep}, at subsequent stages of the primary creep transient for case 1 (MNMT) and case 2 (MNMC) is shown.

The equivalent creep strain in the matrix is defined as:

$$\epsilon_{eq}^{creep} = \sqrt{\frac{2}{3}\epsilon_{ij}^{creep}\epsilon_{ij}^{creep}}, \tag{4.4}$$

where ϵ_{ij}^{creep} are the components of the creep strain tensor.

The steep gradient of ϵ_{eq}^{creep} in the direction normal to the interface, particularly noticeable around the corner of the precipitate, is the "continuum" analogy to the network of dislocations at the interface. The gradient would be even steeper with a higher creep exponent , n, in expression 4.3 .

As the creep transient progresses, and the deviatoric components of the stress tensor are accommodated by the creep strains, a state of hydrostatic stress builds up in the matrix channels. Regardless of the sign of the misfit and of the relative stiffnesses of matrix and precipitates, all the transients evolve toward a stress state, at the end of the primary creep stage, in which (Fig. 4.30 (c)):

- if a *tensile* stress is applied, there is a build up of *negative* pressure in the *horizontal* channels (\e channels normal to the applied stress} and of *positive* pressure in the *vertica* channels (the channels parallel to the applied stress);
- if a *compressive* stress is applied, there is a build up of *positive* pressure in the *horizontal* channels and of *negative* pressure in the *vertical* channels.

The magnitude of the stress in the horizontal channels is higher than the applied stress, while the magnitude of the stress in the vertical channels is much lower than the applied stress.

In Figs. 4.39 through 4.54, typical contour plots of the stress state at the end of the primary creep stage are shown for applied tensile loads (Cases 1,3,5,6,9) and compressive loads (Cases 2,4,6).

As the stress state evolves, the force on the interface is modified as well. Plots of the evolution of the τ_n profile during the stress annealing transient for the eleven c‹ ›es considered are shown in Figs. 4.55 through 4.65. For each case, nine plots are given of the τ_n profile at successive times of the transient. As a normalized measure of the stage of evolution of the creep transient we have chosen the ratio between the average equivalent creep strain in the matrix and the initial m'sfit ($\overline{\epsilon}^{creep}/\delta$). The average creep strain in the matrix is evaluated as a volume averag·, over the γ phase, of the equivalent creep strain:

$$\overline{\epsilon}^{\text{creep}} = \frac{1}{V_\gamma} \int_{V_\gamma} \epsilon_{\text{eq}}^{\text{creep}} dV_\gamma, \tag{4.5}$$

where V_γ is the volume occupied by the γ phase.

We have chosen the parameter $\overline{\epsilon}^{\text{creep}}/\delta$ rather than a more immediate quantity, such as the creep time, because it is more directly related to the evolution of the τ_n profile, as we will discuss in the next paragraph.

Let's now consider the evolution of the τ_n profiles for the first six cases, relative to the simulation of stress annealing tests performed on experimental alloys.

We can notice how the profiles quickly evolve toward configurations that show a marked driving force for directional coarsening. If we compare these graphs with those obtained in our purely elastic analyses, we can see that the *absolute values* of the differences between the force on the top and on the side of the precipitates, $\Delta\tau_n$, are generally increased by one order of magnitude; however, the most important result is that the *signs* of $\Delta\tau_n$ are inverted for several cases so that now for *all* the experimental alloys we have perfect agreement between the observed rafting behavior and the tendency toward directional coarsening that can be inferred from the τ_n profiles.

If we now consider the evolution of the τ_n profiles for the hypothetical alloys of cases 7 through 11, we can list a number of considerations:

- comparing case 7 (MNMT_{INV}) and case 1 (MNMT) we can see that, if we invert the relative stiffnesses of matrix and precipitate, the driving force for rafting remains virtually identical. This result is in antithesis to that of a purely elastic calculation for which an inversion of the elastic constants brings about a parallel inversion of the rafting tendency (see map in Fig. 4.5);
- conversely, comparing case 8 (MNMT_{NMS}) and case 1 (MNMT) we see that if we change the sign of the misfit the τ_n profiles result totally reversed;
- we compare now case 9 (TCT_{PMS}) and case 7 (MNMT_{INV}). Both cases are relative to positive-misfit alloys with soft precipitates but the misfit for case 9 is about 1/20th of the misfit for case 7. We can see that the fourth plot of Fig. 4.63 (case 9), relative to a stage of the transient for which $\overline{\epsilon}^{\text{creep}}/\delta = 1.88$ shows a profile similar to that of the eighth plot of Fig. 4.61 (case 7), for which $\overline{\epsilon}^{\text{creep}}/\delta = 1.83$. However, the value of $\Delta\tau_n$ for this stage of case 9 is only a few percent of the correspondent value for case 7;
- comparing case 10 (TCC_{PMS}) and case 9 (TCT_{PMS}) we see that if we invert the direction of the applied load, the τ_n profiles are reversed. (This can be also noted comparing cases 1 and 2 and cases 3 and 4).
- comparing case 11 (PLT_{ISO}) and case 5 (PLT) we see that a change in the symmetry of the crystal brings about only minor changes in the τ_n profiles which are mainly localized around the corners of the precipitate.

In the next paragraph, we will reinterpret these observations, as well as other features of the τ_n profiles, within the context of a more general framework and we will offer a general discussion of the rafting phenomenon.

σ_{11}	Value (MPa)
1	-2.00E+02
2	-1.30E+02
3	-5.99E+01
4	+1.00E+01
5	+8.00E+01
6	+1.50E+02
7	+2.20E+02
8	+2.90E+02
9	+3.60E+02
10	+4.30E+02
11	+5.00E+02

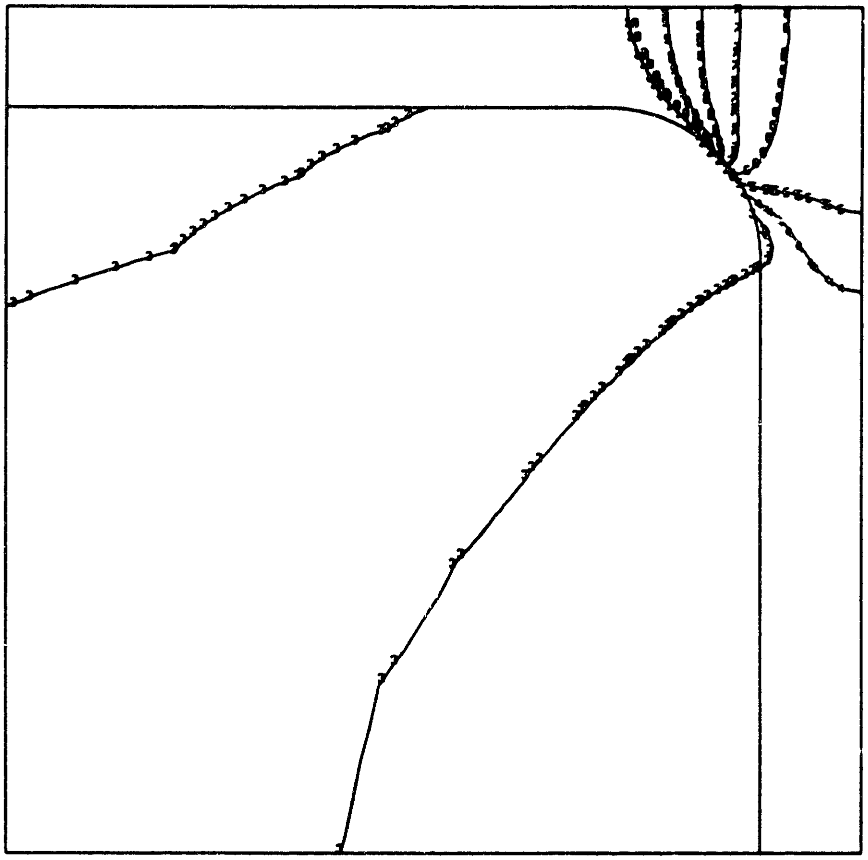


Figure 4.24 Contours of σ_{11} due to misfit only ($\delta = +.56\%$) for the alloy tested by Miyazaki, Nakamura and Mori [16].

σ_{22}	Value (MPa)
1	-2.00E+02
2	-1.30E+02
3	-5.99E+01
4	+1.00E+01
5	+8.00E+01
6	+1.50E+02
7	+2.20E+02
8	+2.90E+02
9	+3.60E+02
10	+4.30E+02
11	+5.00E+02

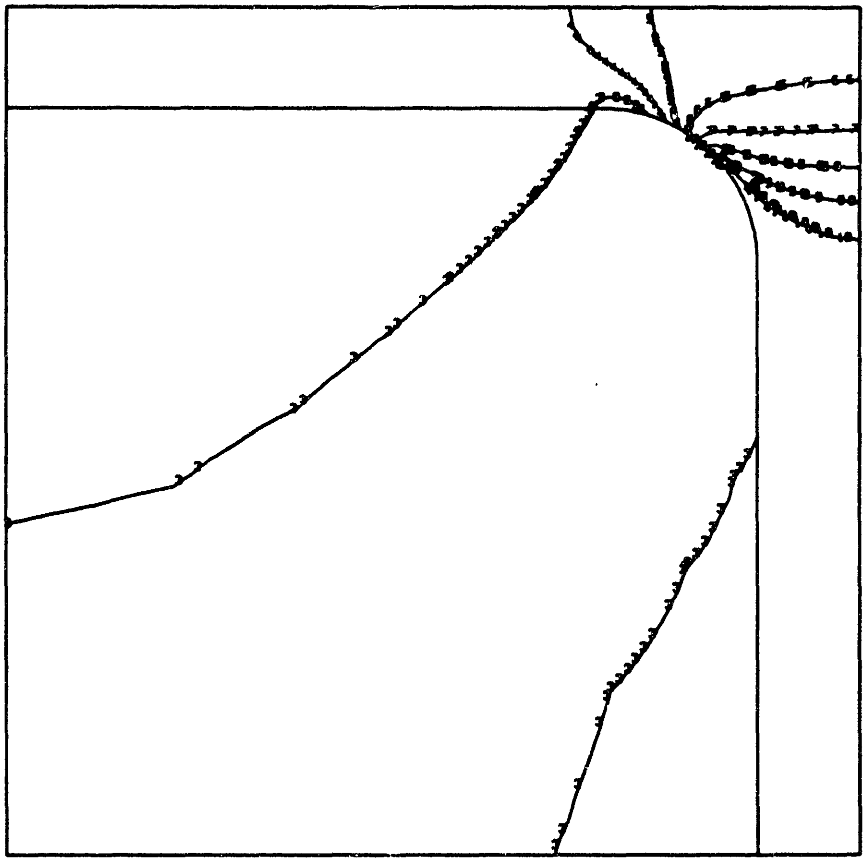


Figure 4.25 Contours of σ_{22} due to misfit only ($\delta = +.56\%$) for the alloy tested by Miyazaki, Nakamura and Mori [16].

σ_{12}	Value (MPa)
1	-5.00E+02
2	-4.40E+02
3	-3.80E+01
4	-3.20E+02
5	-2.60E+02
6	-2.00E+02
7	-1.40E+02
8	-8.00E+01
9	-1.99E+01
10	+4.00E+01
11	+1.00E+02

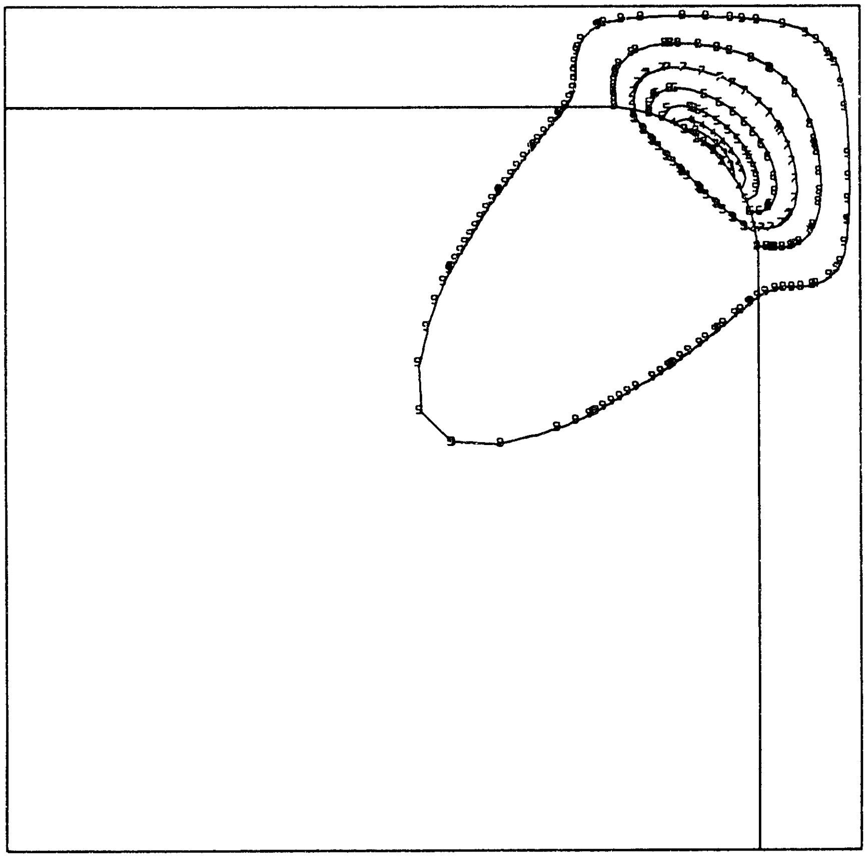


Figure 4.26 Contours of σ_{12} due to misfit only ($\delta = +.56\%$) for the alloy tested by Miyazaki, Nakamura and Mori [16].

σ_{11}	Value (MPa)
1	-6.00E+02
2	-5.20E+02
3	-4.40E+02
4	-3.60E+02
5	-2.80E+02
6	-2.00E+02
7	-1.20E+02
8	-3.99E+01
9	-4.00E+01
10	+1.20E+02
11	+2.00E+02

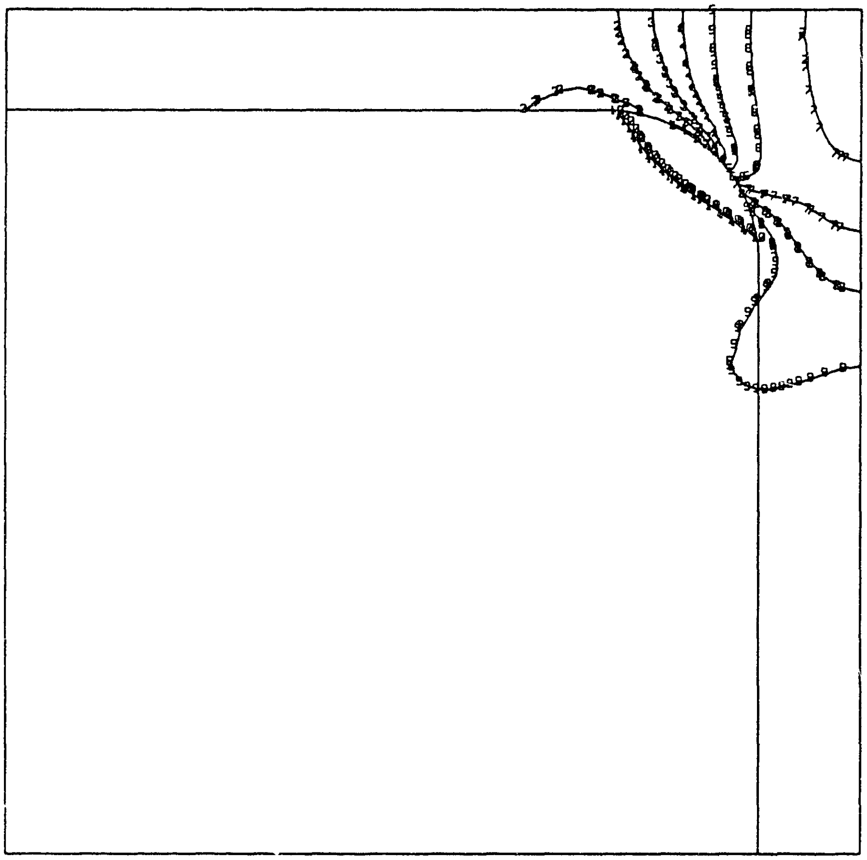


Figure 4.27 Contours of σ_{11} due to misfit only ($\delta = -0.38\%$) for the alloy tested by Pollock [25].

σ_{22}	Value (MPa)
1	-6.00E+02
2	-5.20E+02
3	-4.40E+02
4	-3.60E+02
5	-2.80E+02
6	-2.00E+02
7	-1.20E+02
8	-3.99E+01
9	+4.00E+01
10	+1.20E+02
11	+2.00E+02

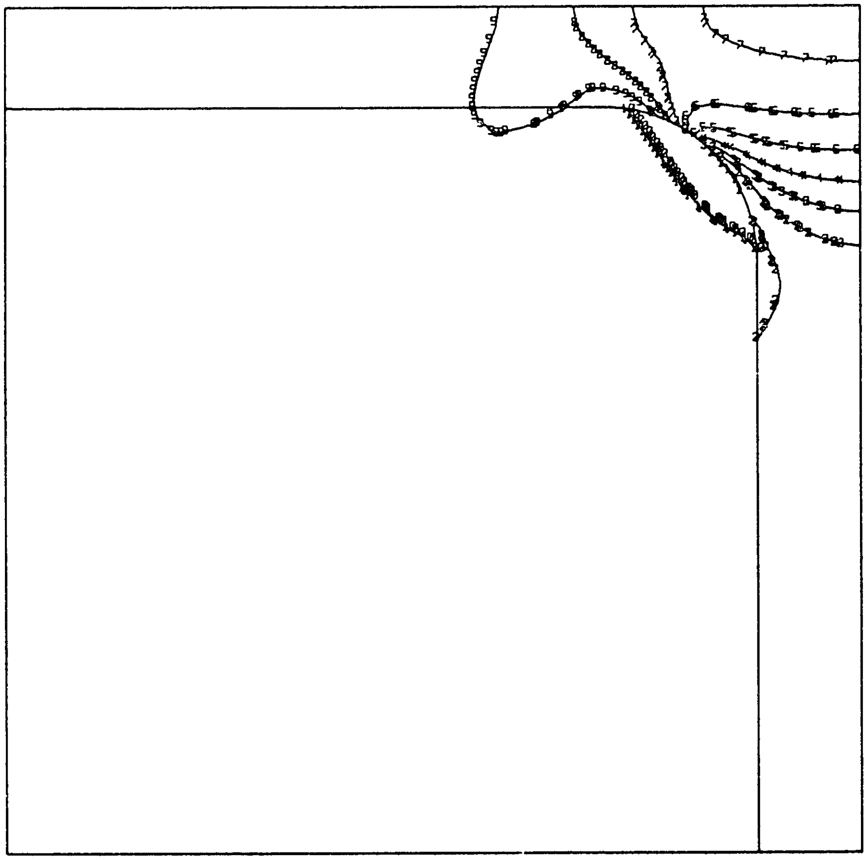


Figure 4.28 Contours of σ_{22} due to misfit only ($\delta = -0.38\%$) for the alloy tested by Pollock [25].

σ_{12}	Value (MPa)
1	+7.00E+05
2	+7.00E+01
3	+1.40E+02
4	+2.10E+02
5	+2.80E+02
6	+3.50E+02
7	+4.20E+02
8	+4.90E+02
9	+5.50E+02
10	+6.30E+02
11	+7.00E+02

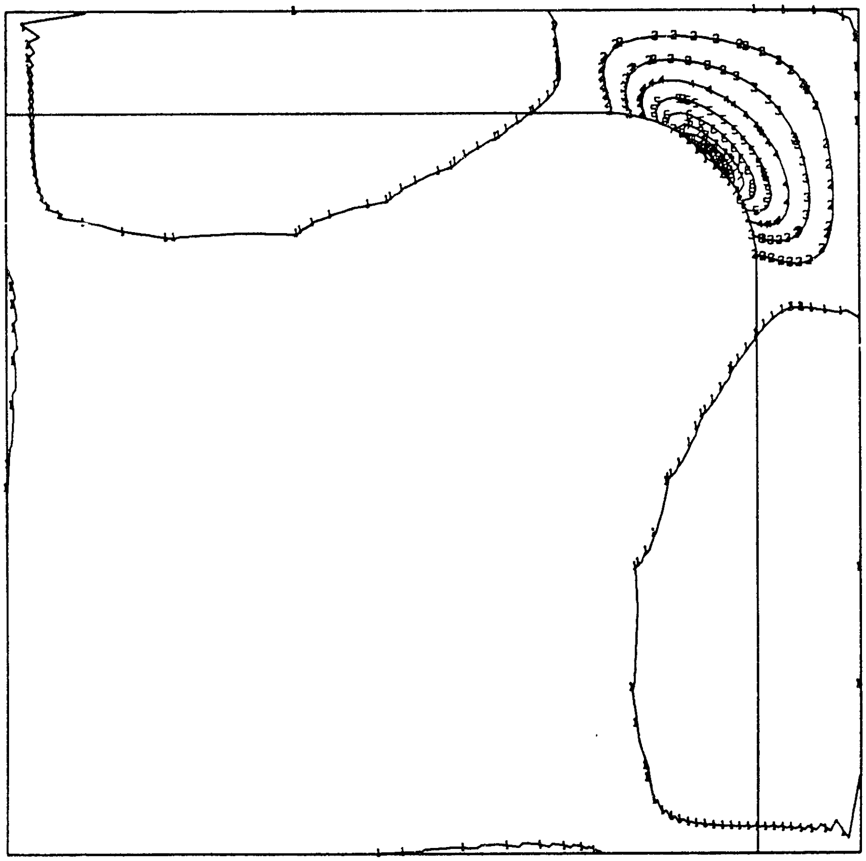


Figure 4.29 Contours of σ_{12} due to misfit only ($\delta = -0.38\%$) for the alloy tested by Pollock [25].

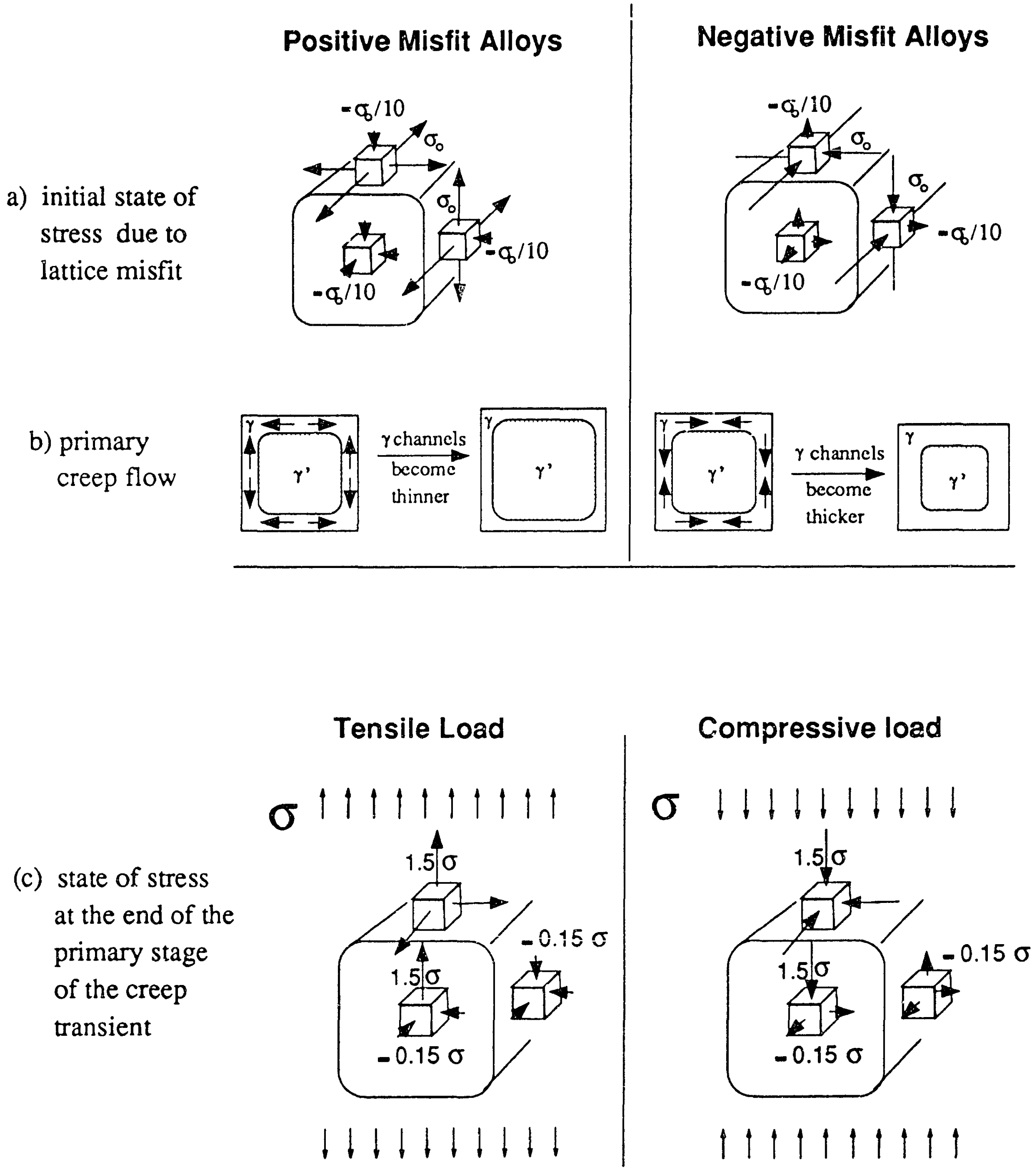


Figure 4.30 Schematic representation of the evolution of the stress and strain fields in the primary stage of the creep transient.

ϵ_{eq}^{creep}	Value
1	+3.00E-10
2	+3.00E-04
3	+6.00E-04
4	+9.00E-04
5	+1.20E-03
6	+1.50E-03
7	+1.80E-03
8	+2.10E-03
9	+2.40E-03
10	+2.70E-03
11	+3.00E-03

$\overline{\epsilon}^{creep}/\delta = 0.0853$

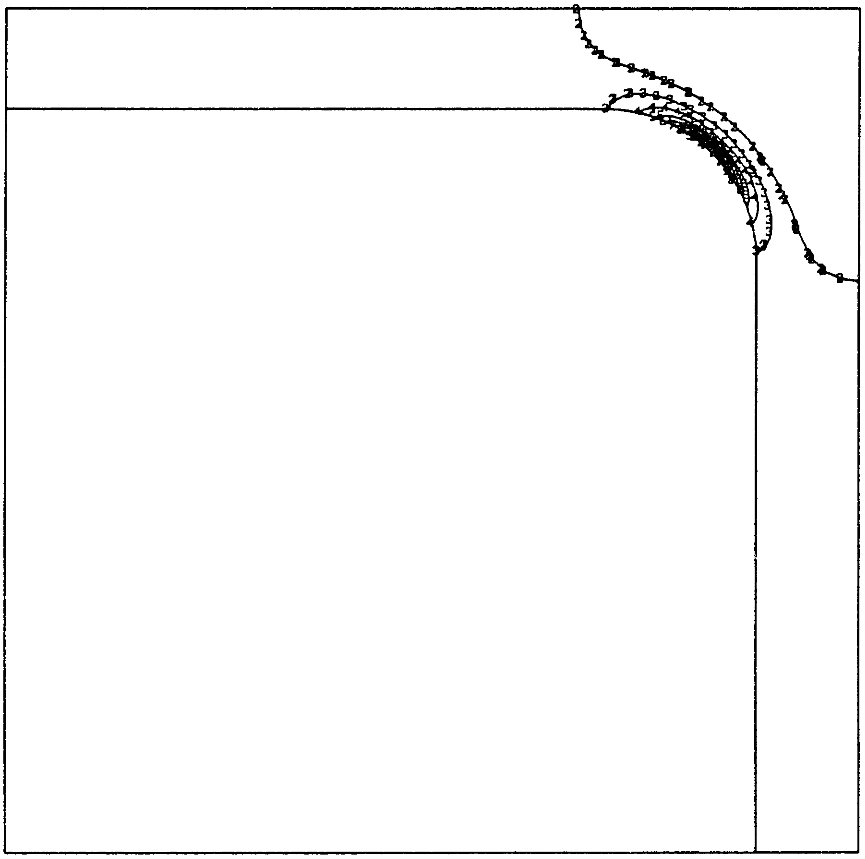


Figure 4.31 Contours of ϵ_{eq}^{creep} for the transient analyzed in case 1 (MNMT) at creep time t = 0.011 sec.

ϵ_{eq}^{creep}	Value
1	+9.00E–10
2	+9.00E–04
3	+1.80E–03
4	+2.70E–03
5	+3.60E–03
6	+4.50E–03
7	+5.40E–03
8	+6.30E–03
9	+7.20E–03
10	+8.10E–03
11	+9.00E–03

$\overline{\epsilon}^{creep}/\delta = 0.98$

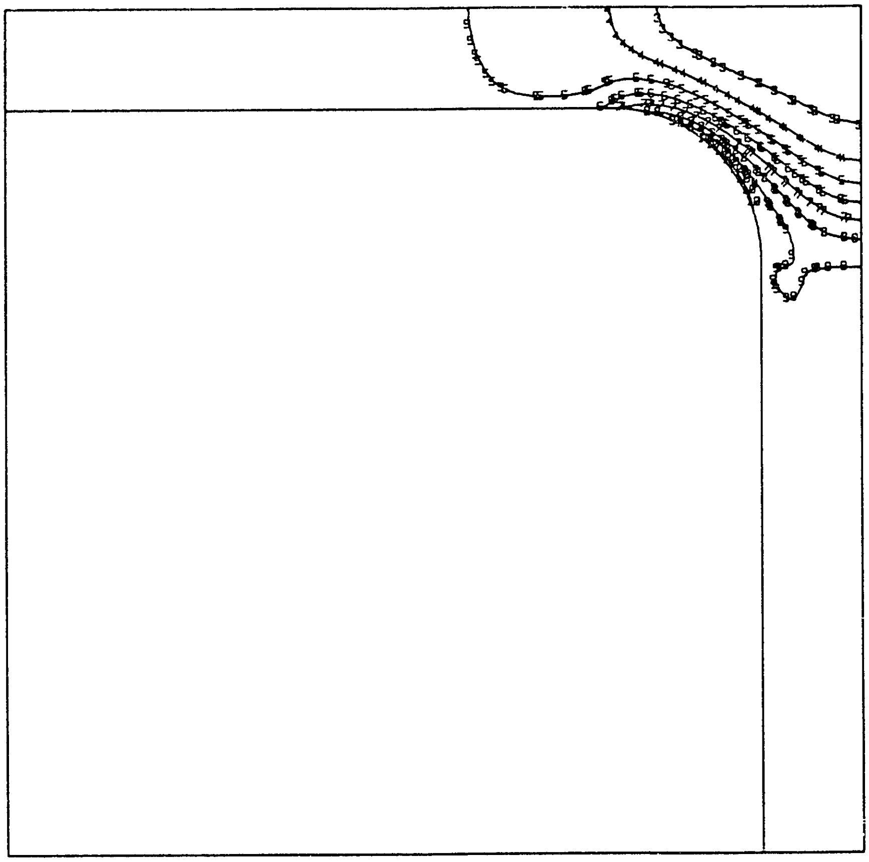


Figure 4.32 Contours of ϵ_{eq}^{creep} for the transient analyzed in case 1 (MNMT) at creep time t = 4.46 sec.

ϵ_{eq}^{creep}	Value
1	+3.00E-09
2	+3.00E-03
3	+6.00E-03
4	+9.00E-03
5	+1.20E-02
6	+1.50E-02
7	+1.80E-02
8	+2.10E-02
9	+2.40E-02
10	+2.70E-02
11	+3.00E-02

$\overline{\epsilon}^{creep}/\delta = 1.55$

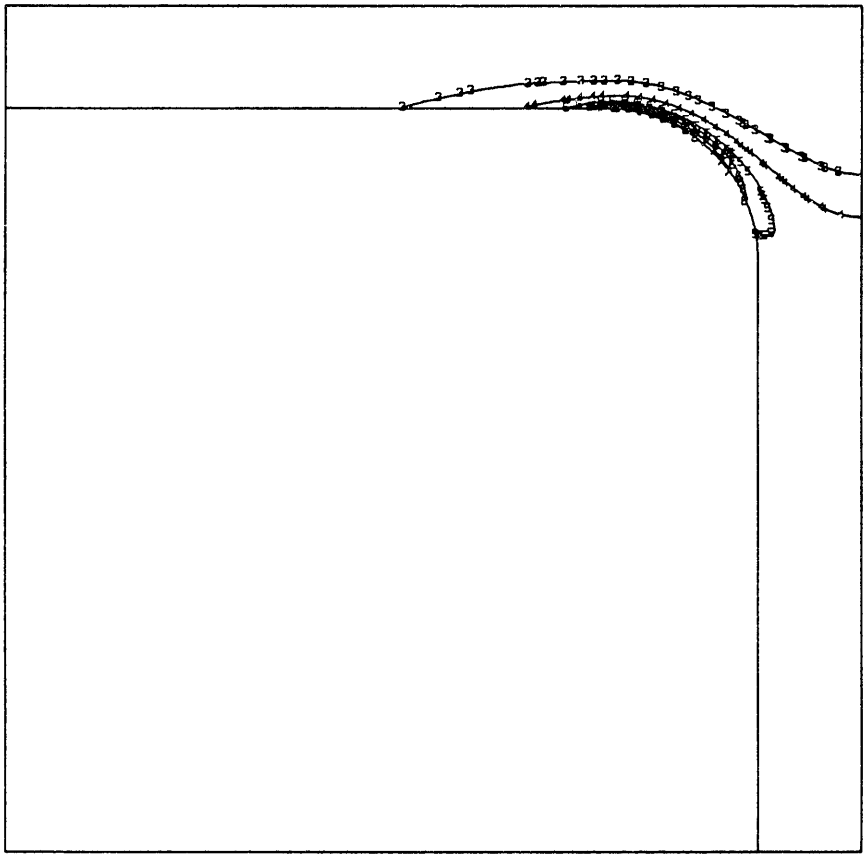


Figure 4.33 Contours of ϵ_{eq}^{creep} for the transient analyzed in case 1 (MNMT) at creep time t = 63.47 sec.

ϵ_{eq}^{creep}	Value
1	+9.00E-09
2	+9.00E-03
3	+1.80E-02
4	+2.70E-02
5	+3.60E-02
6	+4.50E-02
7	+5.40E-02
8	+6.30E-02
9	+7.20E-02
10	+8.10E-02
11	+9.00E-02

$\overline{\epsilon}^{creep}/\delta = 2.39$

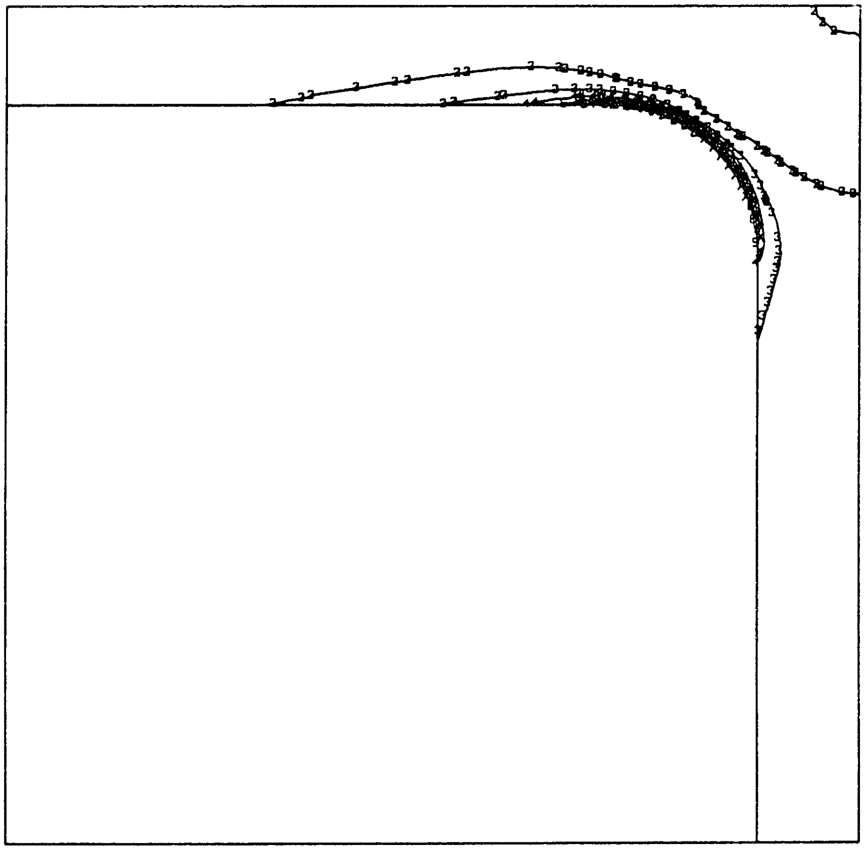


Figure 4.34 Contours of ϵ_{eq}^{creep} for the transient analyzed in case 1 (MNMT) at creep time t = 50,000 sec.

$\epsilon_{\mathrm{eq}}^{\mathrm{creep}}$	Value
1	+3.00E-10
2	+3.00E-04
3	+6.00E-04
4	+9.00E-04
5	+1.20E-03
6	+1.50E-03
7	+1.80E-03
8	+2.10E-03
9	+2.40E-03
10	+2.70E-03
11	+3.00E-03

$\overline{\epsilon}^{\mathrm{creep}}/\delta = 0.085$

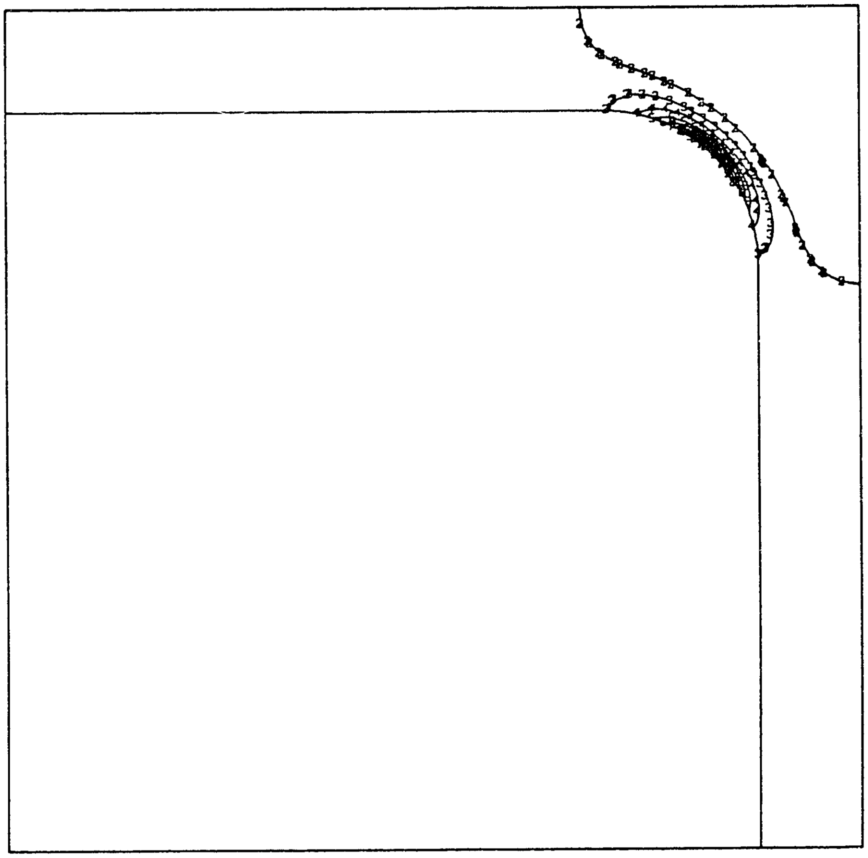


Figure 4.35 Contours of $\epsilon_{\mathrm{eq}}^{\mathrm{creep}}$ for the transient analyzed in case 2 (MNMC) at creep time t =0.011 sec.

$\epsilon_{\text{eq}}^{\text{creep}}$	Value
1	+1.00E-09
2	+1.00E-03
3	+2.00E-03
4	+3.00E-03
5	+4.00E-03
6	+5.00E-03
7	+6.00E-03
8	+7.00E-03
9	+8.00E-03
10	+9.00E-03
11	+1.00E-02

$\overline{\epsilon}^{\text{creep}}/\delta = 1.25$

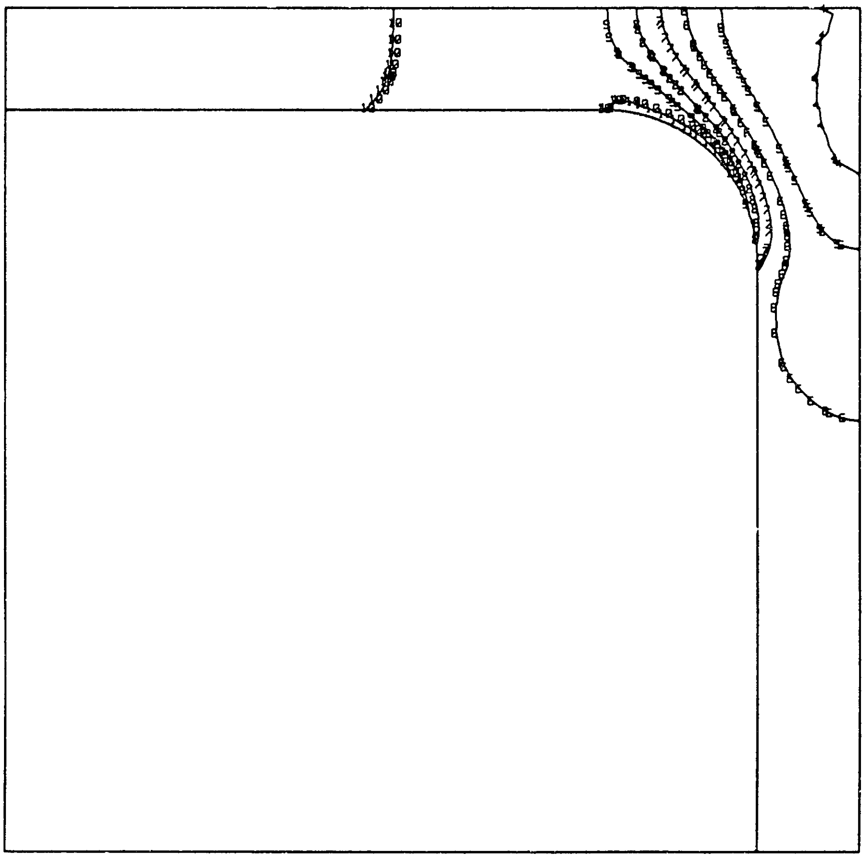


Figure 4.36 Contours of $\epsilon_{\text{eq}}^{\text{creep}}$ for the transient analyzed in case 2 (MNMC) at creep time t = 3.86 sec.

ϵ_{eq}^{creep}	Value
1	+4.00E-09
2	+4.00E-03
3	+8.00E-03
4	+1.20E-02
5	+1.60E-02
6	+2.00E-02
7	+2.40E-02
8	+2.80E-02
9	+3.20E-02
10	+3.60E-02
11	+4.00E-02

$\bar{\epsilon}^{creep}/\delta = 2.23$

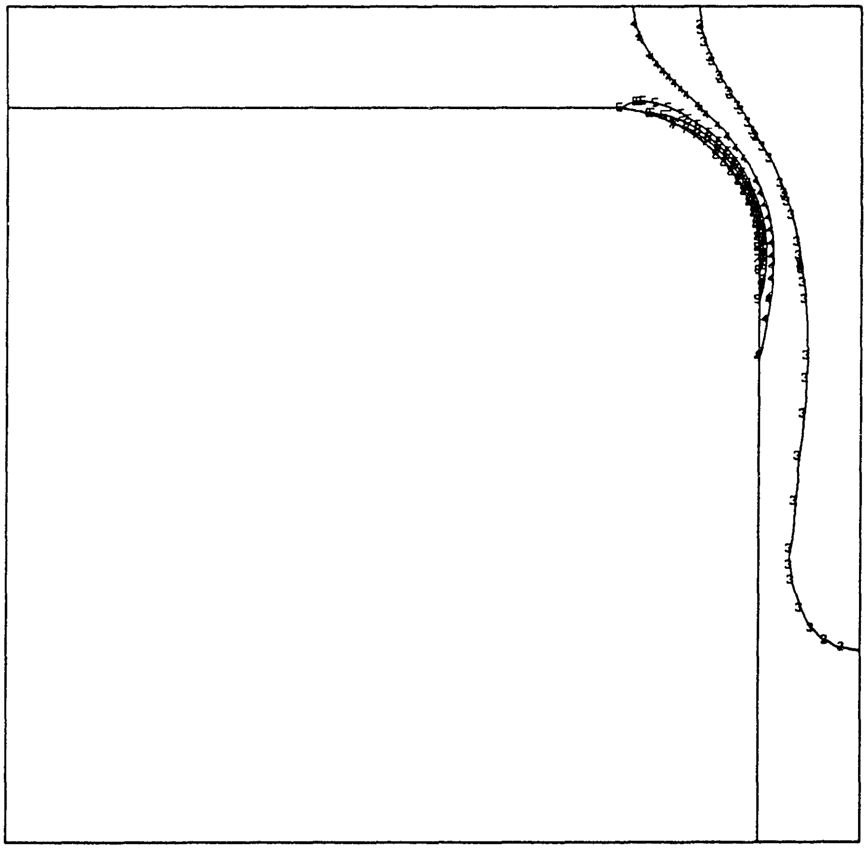


Figure 4.37 Contours of ϵ_{eq}^{creep} for the transient analyzed in case 2 (MNMC) at creep time t = 252 sec.

$\epsilon_{\mathrm{eq}}^{\mathrm{creep}}$	Value
1	+8.00E-09
2	+8.00E-03
3	+1.60E-02
4	+2.40E-02
5	+3.20E-02
6	+4.00E-02
7	+4.80E-02
8	+5.60E-02
9	+6.40E-02
10	+7.20E-02
11	+8.00E-02

$\overline{\epsilon}^{\mathrm{creep}}/\delta = 2.71$

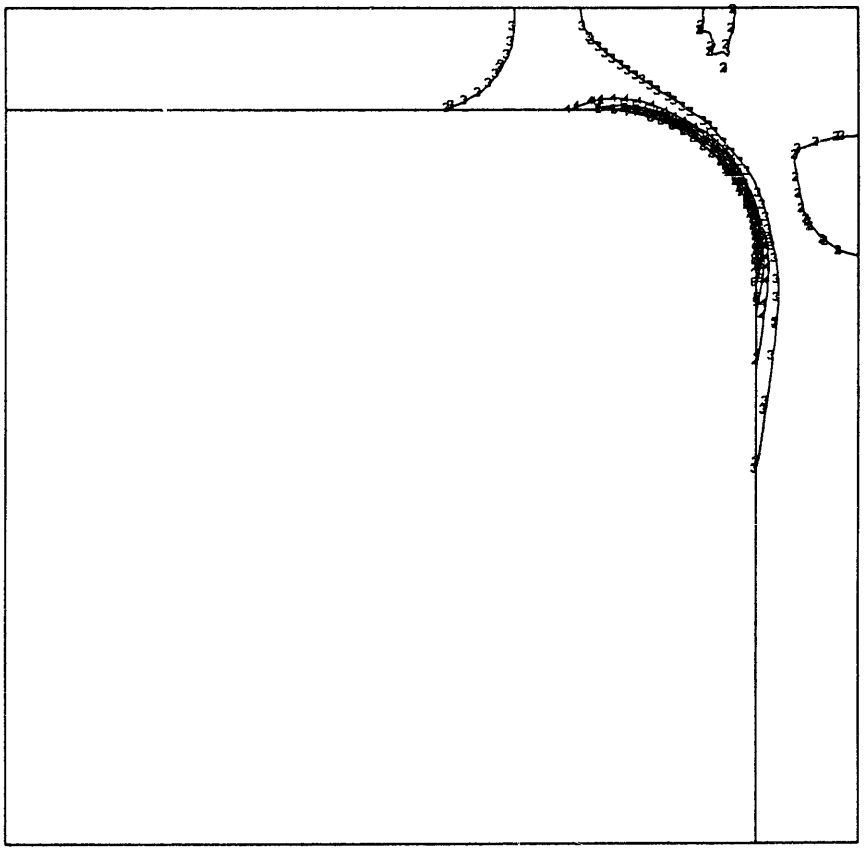


Figure 4.38 Contours of $\epsilon_{\mathrm{eq}}^{\mathrm{creep}}$ for the transient analyzed in case 2 (MNMC) at creep time t = 50,000 sec.

σ_{11}	Value (MPa)
1	-1.00E+02
2	-7.00E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

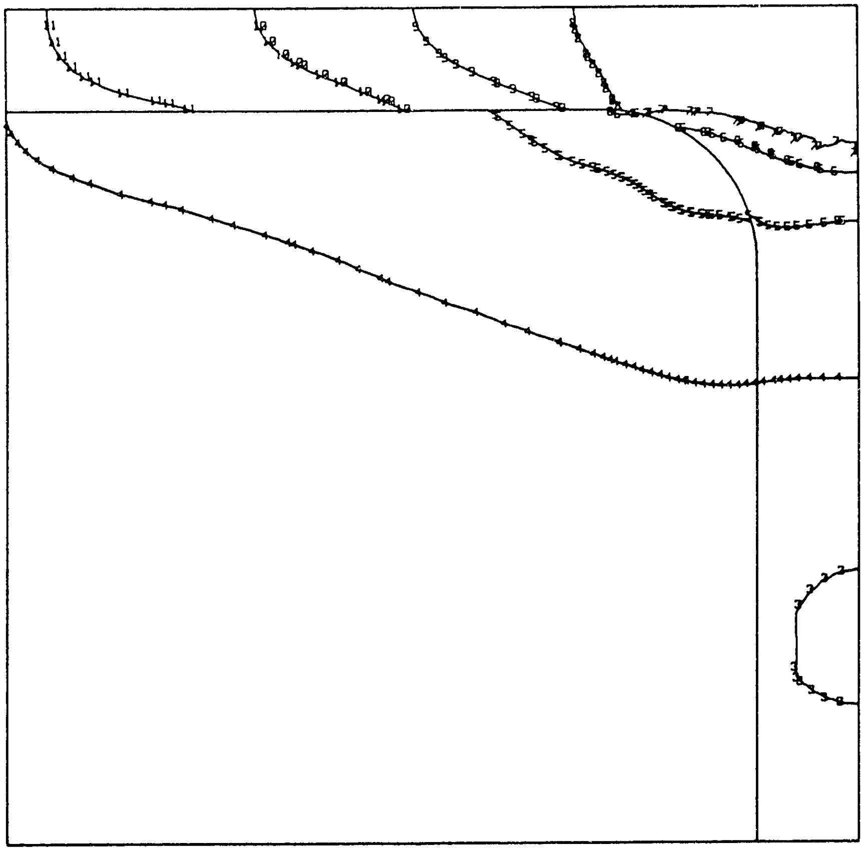


Figure 4.39 Contours of σ_{11} at the end of the analyzed transient for case 1 (MNMT).

σ_{22}	Value (MPa)
1	-1.00E+02
2	-7.00E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

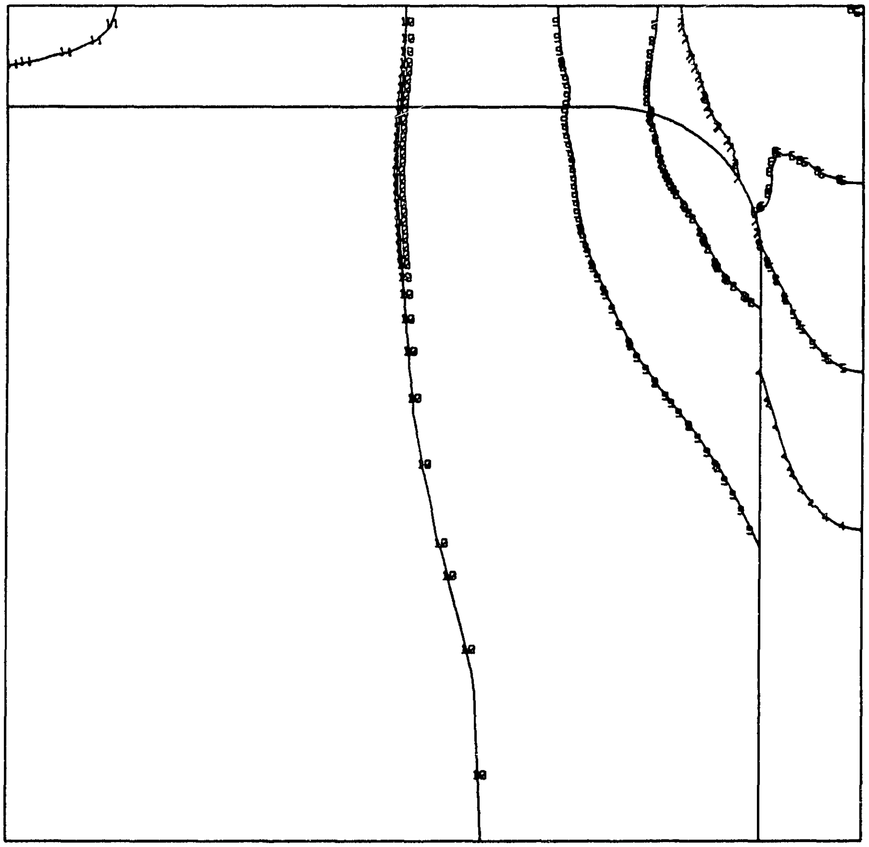


Figure 4.40 Contours of σ_{22} at the end of the analyzed transient for case 1 (MNMT).

σ_{11}	Value (MPa)
1	-1.00E+02
2	-7.00E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

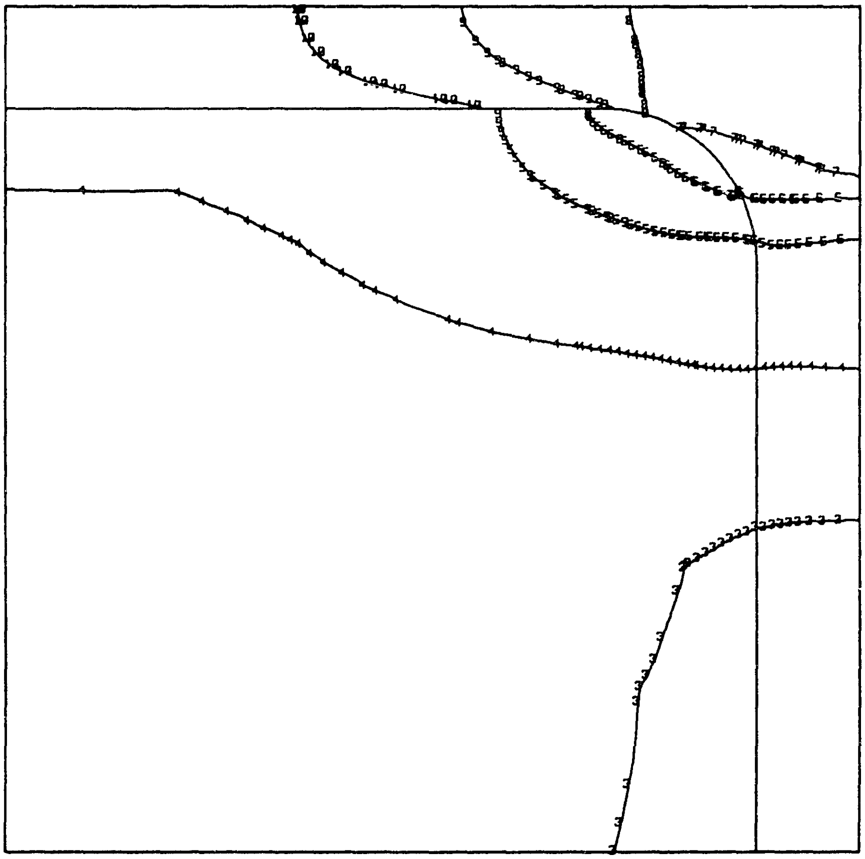


Figure 4.41 Contours of σ_{11} at the end of the analyzed transient for case 3 (TCT).

σ_{22}	Value (MPa)
1	-1.00E+02
2	-7.00E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

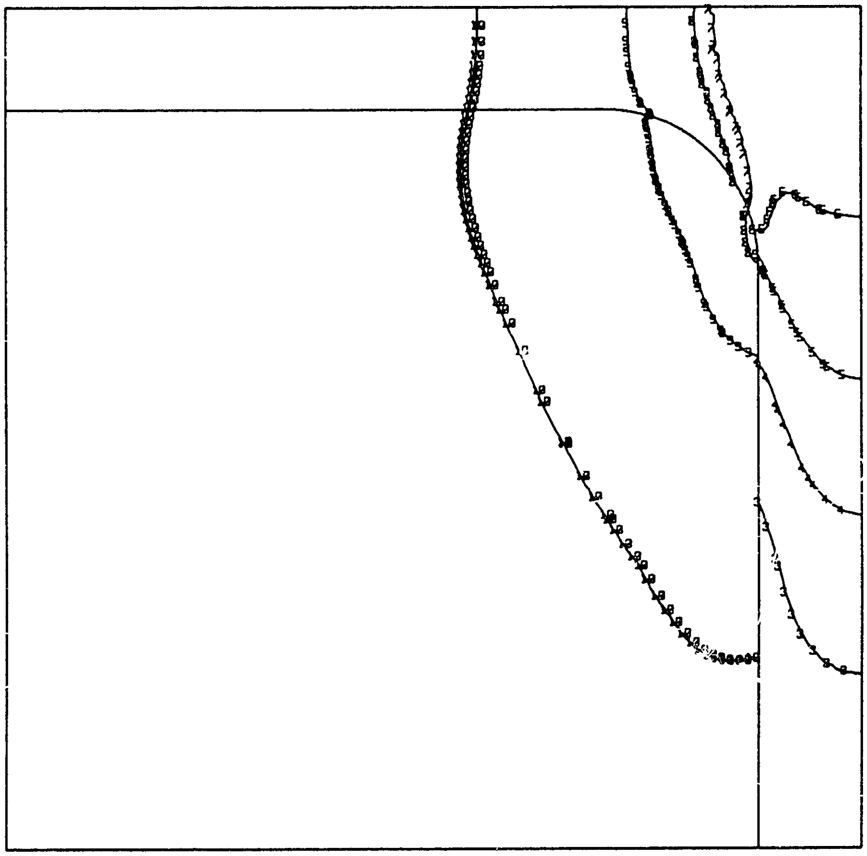


Figure 4.42 Contours of σ_{22} at the end of the analyzed transient for case 3 (TCT).

σ_{11}	Value (MPa)
1	-2.00E+01
2	-1.20E+01
3	-3.99E+00
4	-4.00E+00
5	+1.20E+01
6	+2.00E+01
7	+2.80E+01
8	+3.60E+01
9	+4.40E+01
10	+5.20E+01
11	+6.00E+01

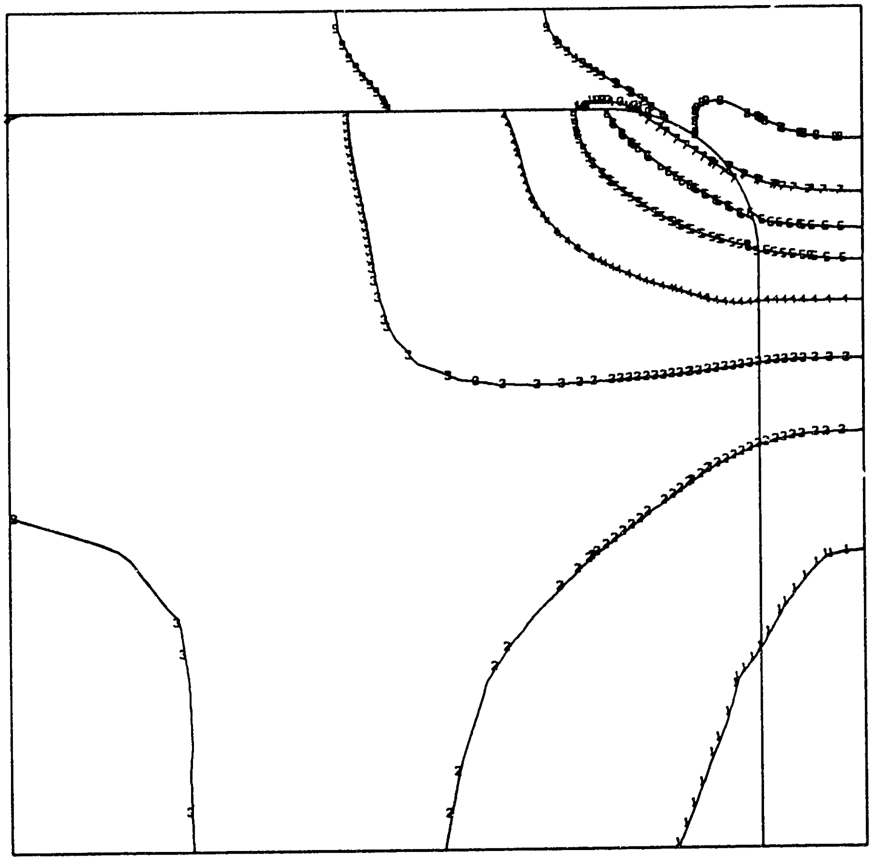


Figure 4.43 Contours of σ_{11} at the end of the analyzed transient for case 5 (PLT).

σ_{22}	Value (MPa)
1	-3.00E+01
2	-2.10E+01
3	-1.20E+01
4	-2.99E+00
5	+6.00E+00
6	+1.50E+01
7	+2.40E+01
8	+3.30E+01
9	+4.20E+01
10	+5.10E+01
11	+6.00E+01

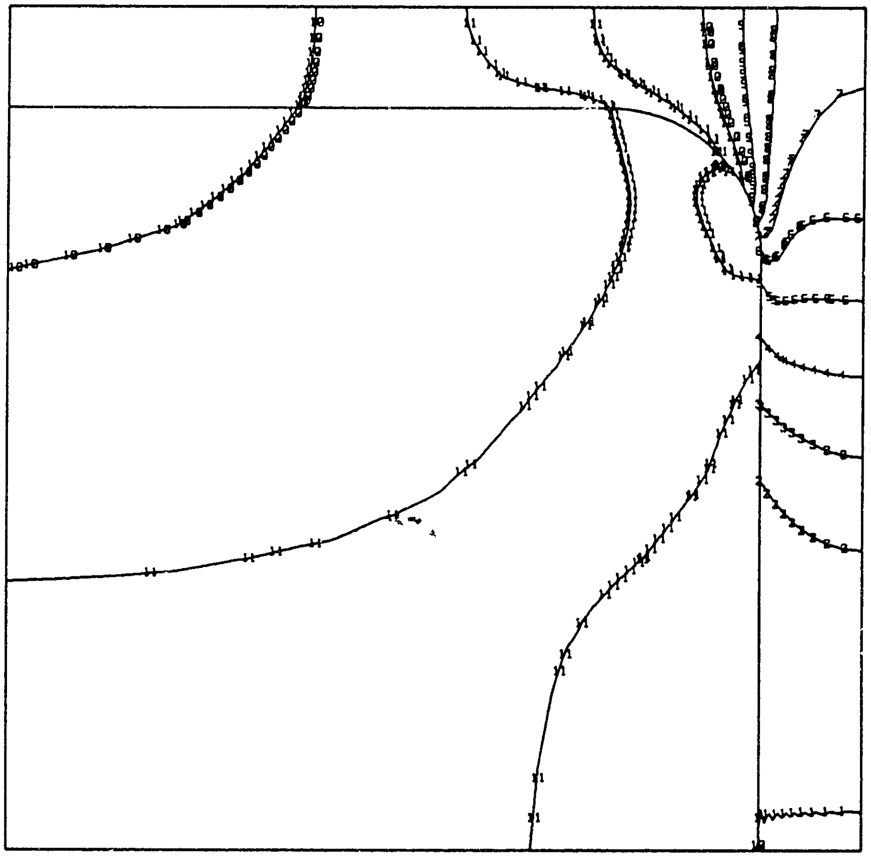


Figure 4.44 Contours of σ_{22} at the end of the analyzed transient for case 5 (PLT).

σ_{11}	Value (MPa)
1	-1.00E+02
2	-7.10E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

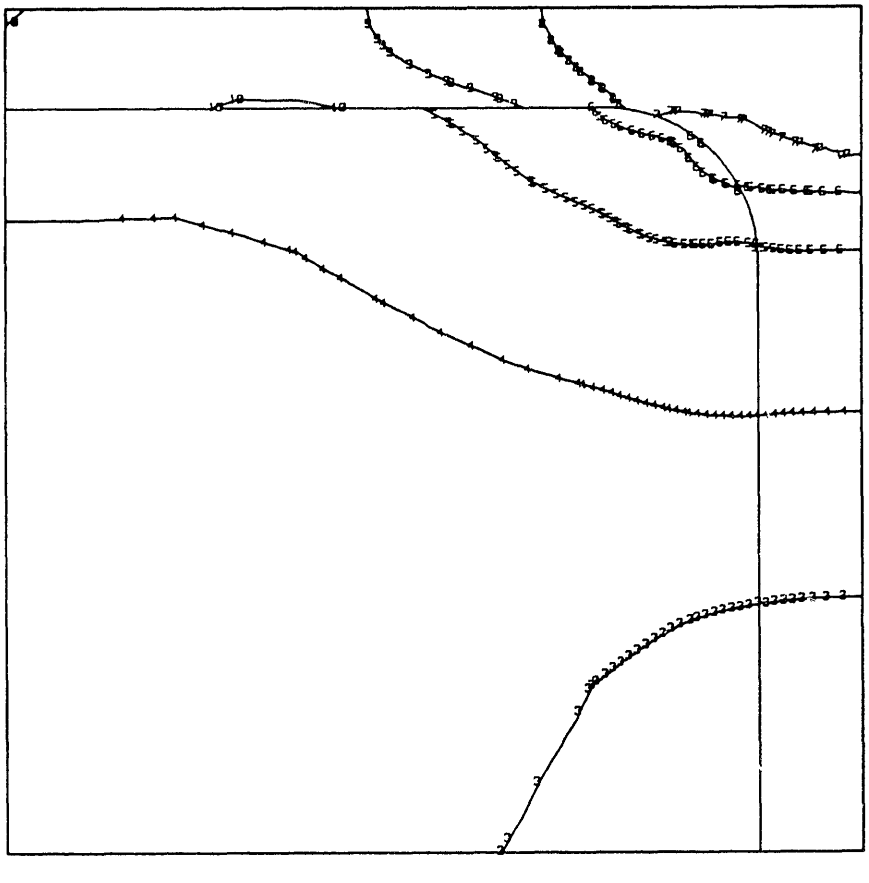


Figure 4.45 Contours of σ_{11} at the end of the analyzed transient for case 6 (PHT).

σ_{11}	Value (MPa)
1	-1.00E+02
2	-7.10E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

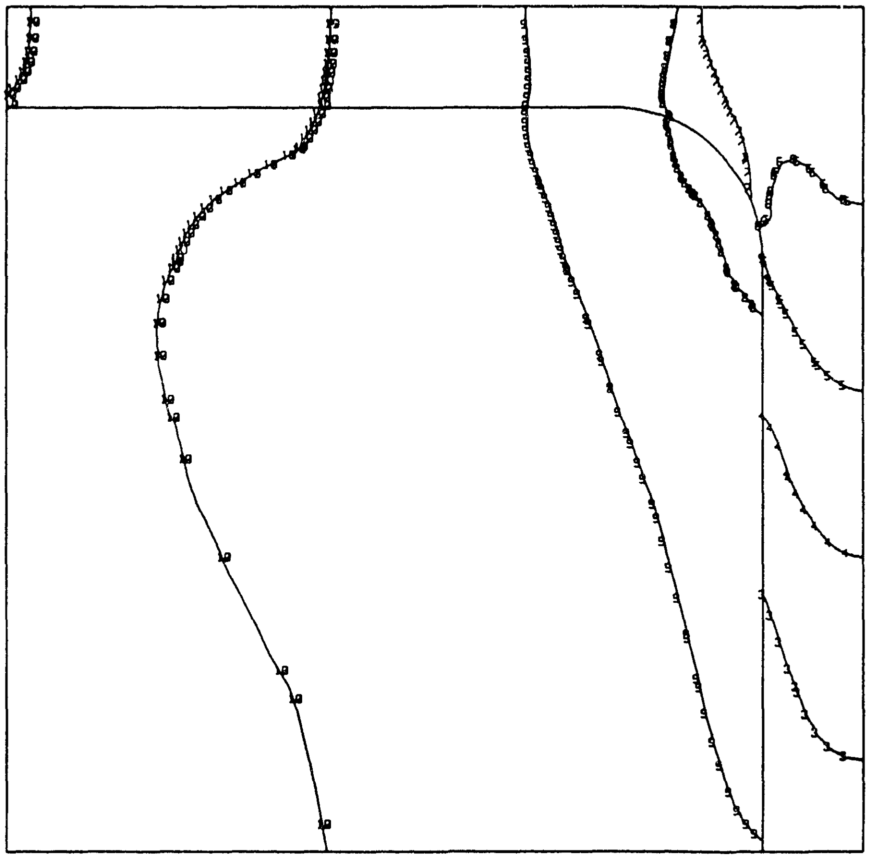


Figure 4.46 Contours of σ_{22} at the end of the analyzed transient for case 6 (PHT).

σ_{11}	Value (MPa)
1	-1.00E+02
2	-7.10E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

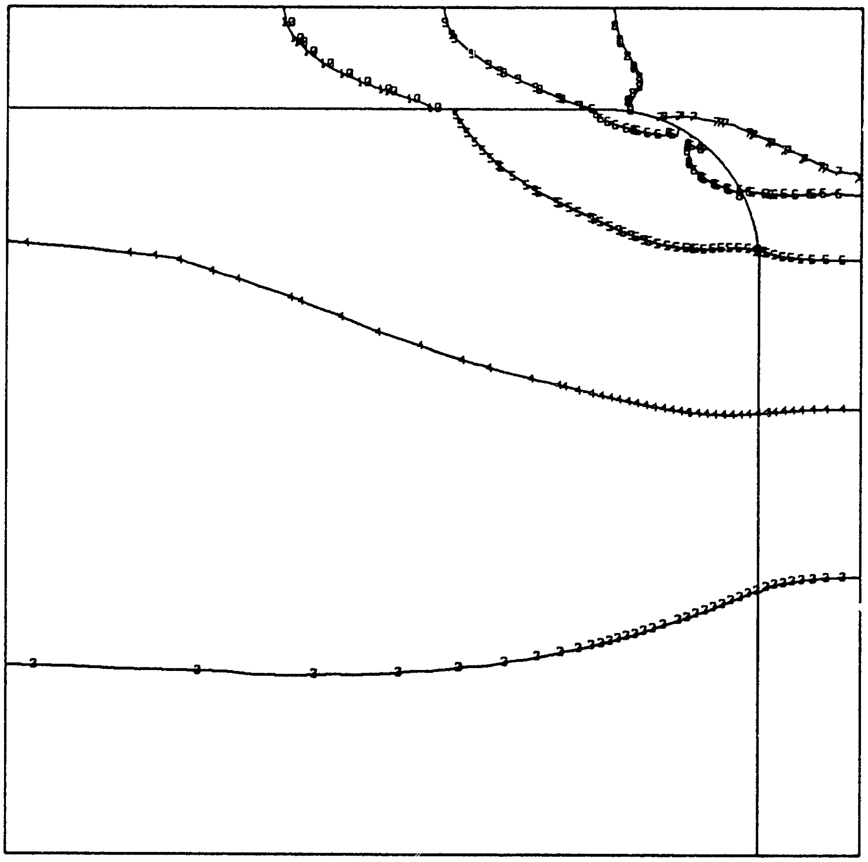


Figure 4.47 Contours of σ_{11} at the end of the analyzed transient for case 9 (TCT_{PMS}).

σ_{22}	Value (MPa)
1	-1.00E+02
2	-7.10E+01
3	-4.00E+01
4	-9.99E+00
5	+2.00E+01
6	+5.00E+01
7	+8.00E+01
8	+1.10E+02
9	+1.40E+02
10	+1.70E+02
11	+2.00E+02

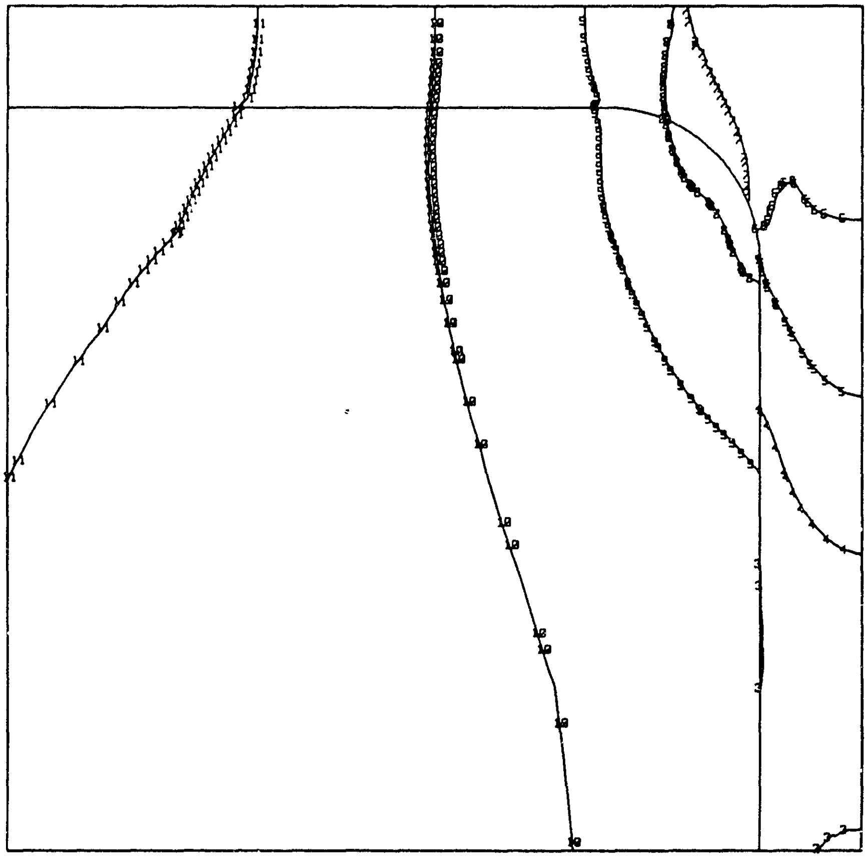


Figure 4.48 Contours of σ_{22} at the end of the analyzed transient for case 9 (TCT_{PMS}).

σ_{11}	Value (MPa)
1	-2.00E+02
2	-1.70E+02
3	-1.40E+02
4	-1.10E+02
5	-8.00E+01
6	-5.00E+01
7	-1.99E+01
8	+1.00E+01
9	+4.00E+01
10	+7.00E+01
11	+1.00E+02

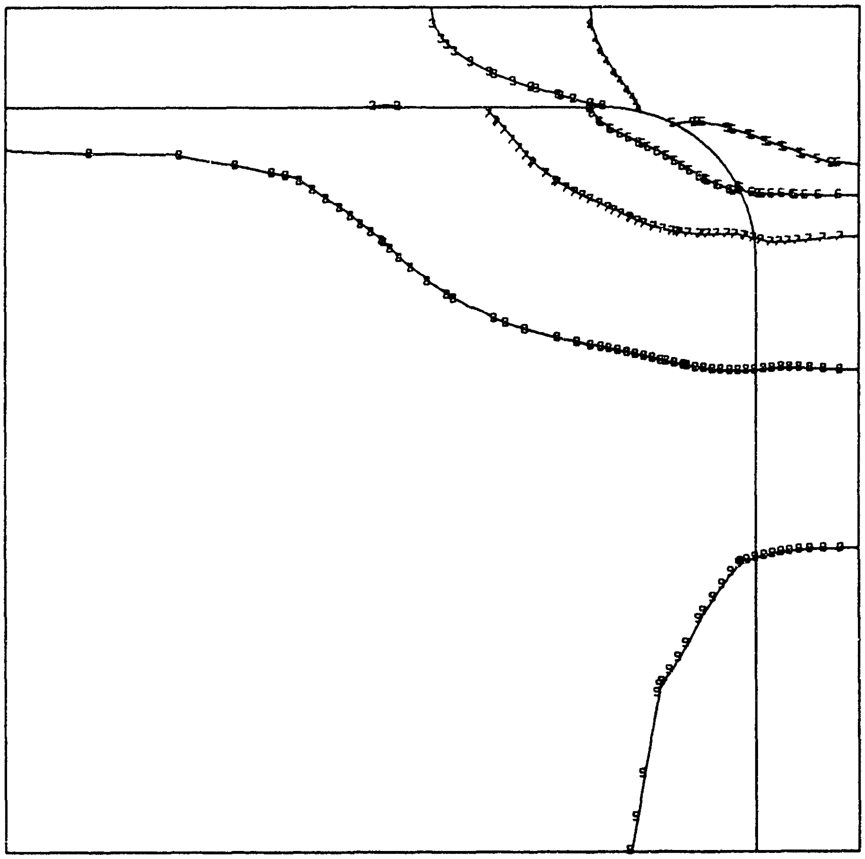


Figure 4.49 Contours of σ_{11} at the end of the analyzed transient for case 2 (MNMC).

σ_{22}	Value (MPa)
1	-2.00E+02
2	-1.70E+02
3	-1.40E+02
4	-1.10E+02
5	-8.00E+01
6	-5.00E+01
7	-1.99E+01
8	+1.00E+01
9	+4.00E+01
10	+7.00E+01
11	+1.00E+02

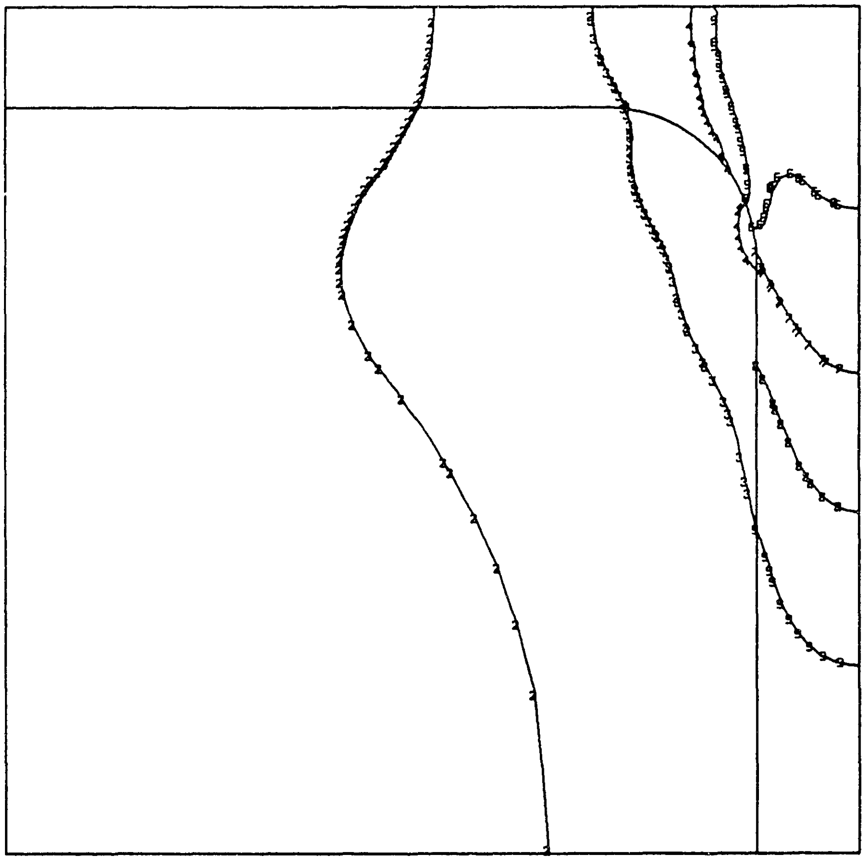


Figure 4.50 Contours of σ_{22} at the end of the analyzed transient for case 2 (MNMC).

σ_{11}	Value (MPa)
1	-3.00E+02
2	-2.60E+02
3	-2.20E+02
4	-1.80E+02
5	-1.40E+02
6	-1.00E+02
7	-6.00E+01
8	+1.99E+01
9	+2.00E+01
10	+6.00E+01
11	+1.00E+02

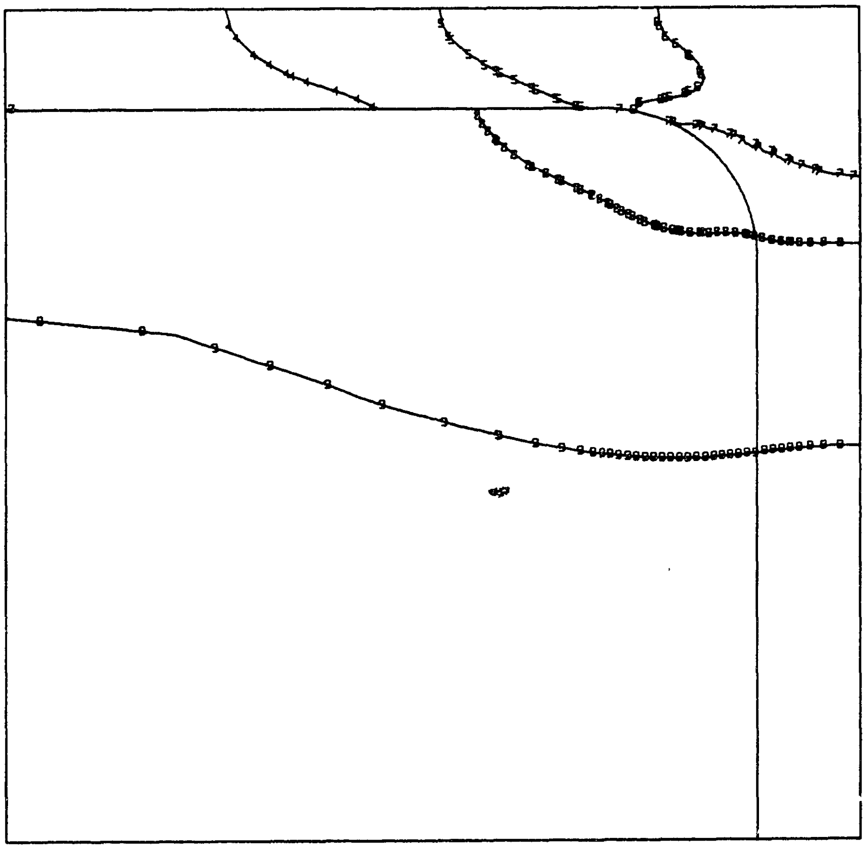


Figure 4.51 Contours of σ_{11} at the end of the analyzed transient for case 4 (TCC).

σ_{22}	Value (MPa)
1	-3.00E+02
2	-2.60E+02
3	-2.20E+02
4	-1.80E+02
5	-1.40E+02
6	-1.00E+02
7	-6.00E+01
8	+1.99E+01
9	+2.00E+01
10	+6.00E+01
11	+1.00E+02

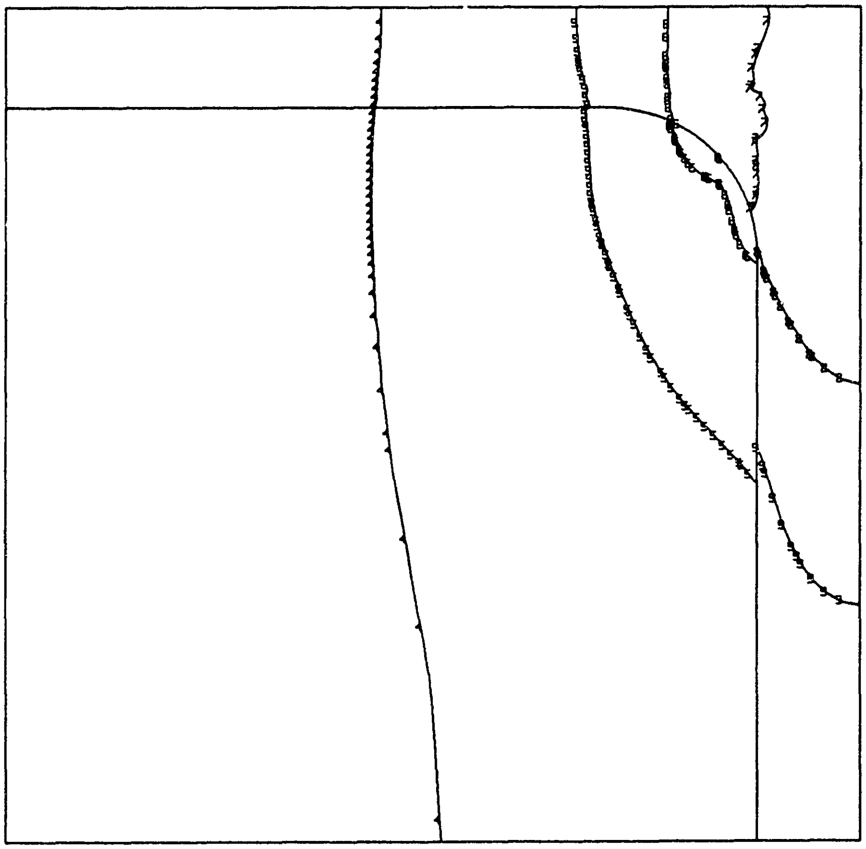


Figure 4.52 Contours of σ_{22} at the end of the analyzed transient for case 4 (TCC).

σ_{11}	Value (MPa)
1	-2.00E+02
2	-1.70E+02
3	-1.40E+02
4	-1.10E+02
5	-8.00E+01
6	-5.00E+01
7	-1.99E+01
8	+1.00E+01
9	+4.00E+01
10	+7.00E+01
11	+1.00E+02

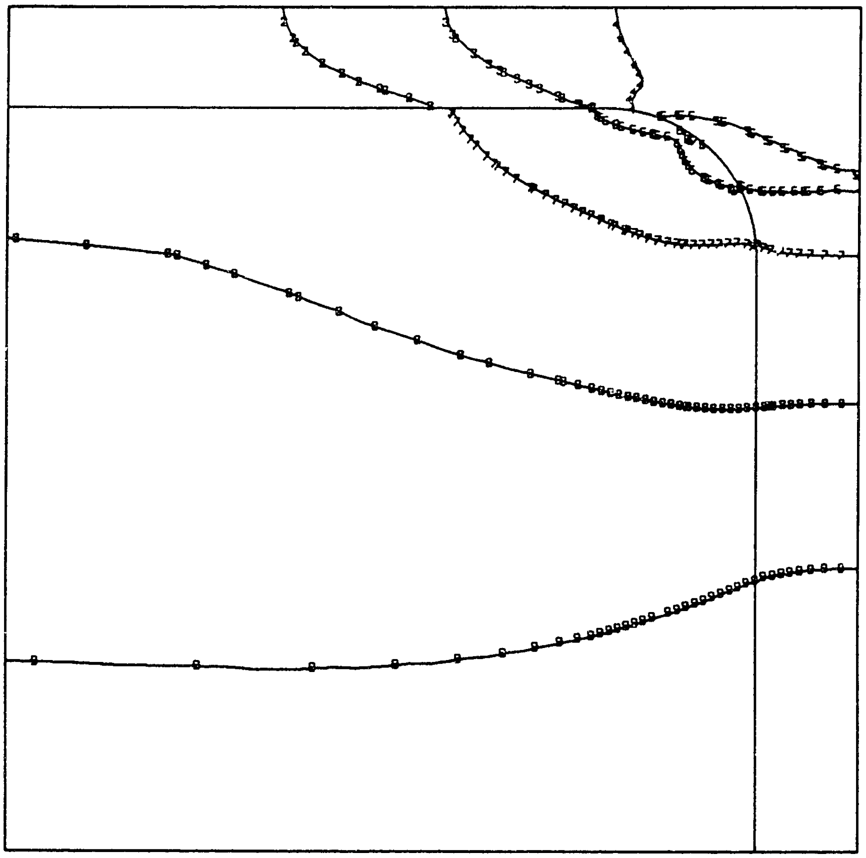


Figure 4.53 Contours of σ_{11} at the end of the analyzed transient for case 10 (TCC_{PMS}).

σ_{22}	Value (MPa)
1	-2.00E+02
2	-1.70E+02
3	-1.40E+02
4	-1.10E+02
5	-8.00E+01
6	-5.00E+01
7	-1.99E+01
8	+1.00E+01
9	+4.00E+01
10	+7.00E+01
11	+1.00E+02

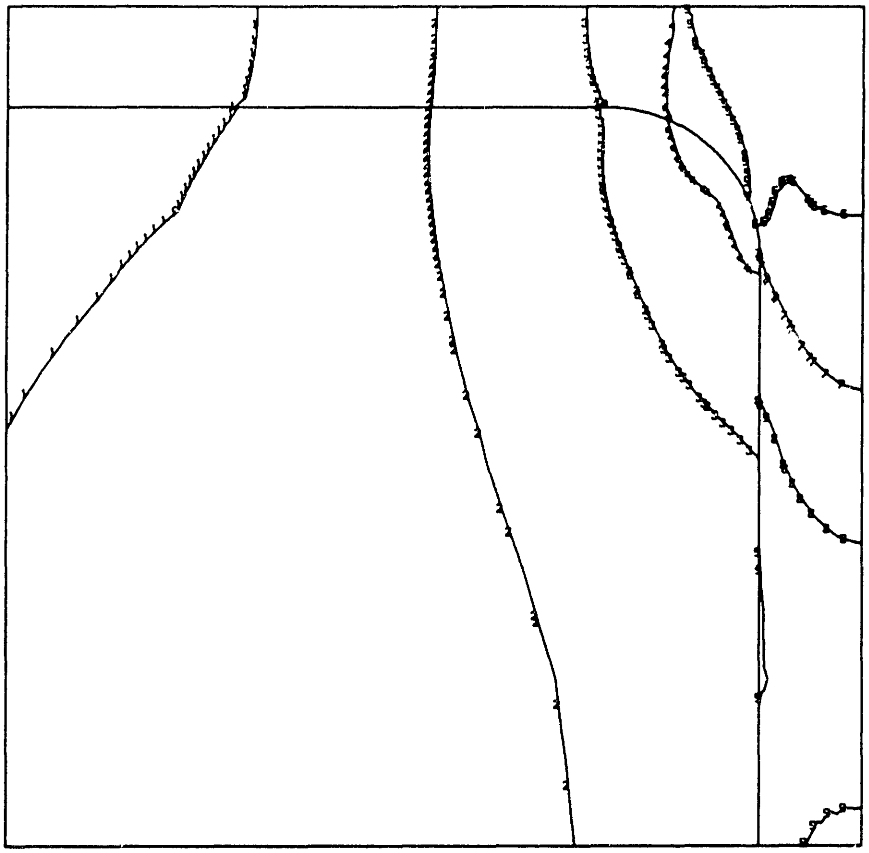


Figure 4.54 Contours of σ_{22} at the end of the analyzed transient for case 10 (TCC$_{PMS}$).

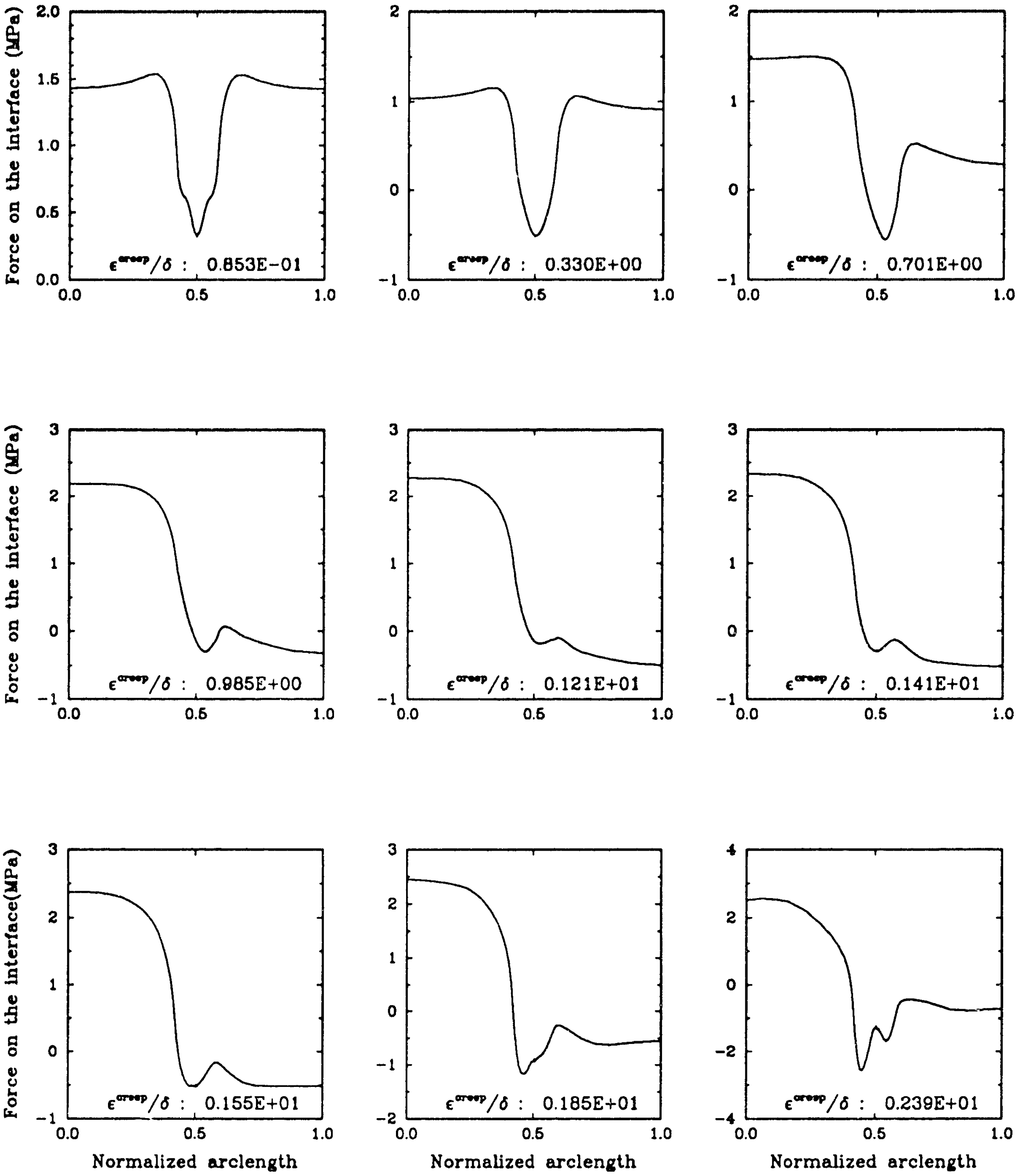


Figure 4.55 Evolution of the τ_n profile during the stress-annealing transient for case 1 (MNMT).

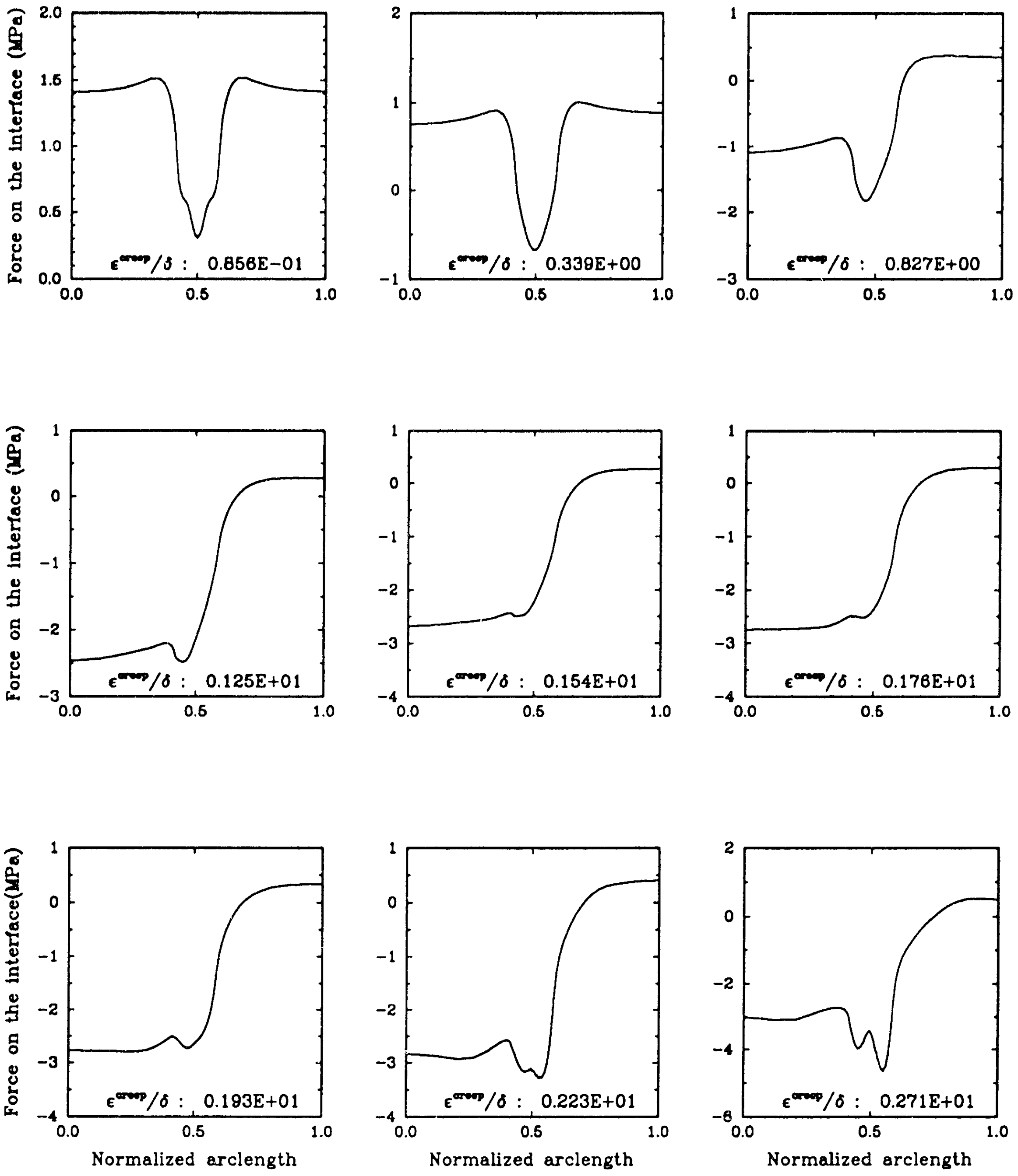


Figure 4.56 Evolution of the τ_n profile during the stress-annealing transient for case 2 (MNMC).

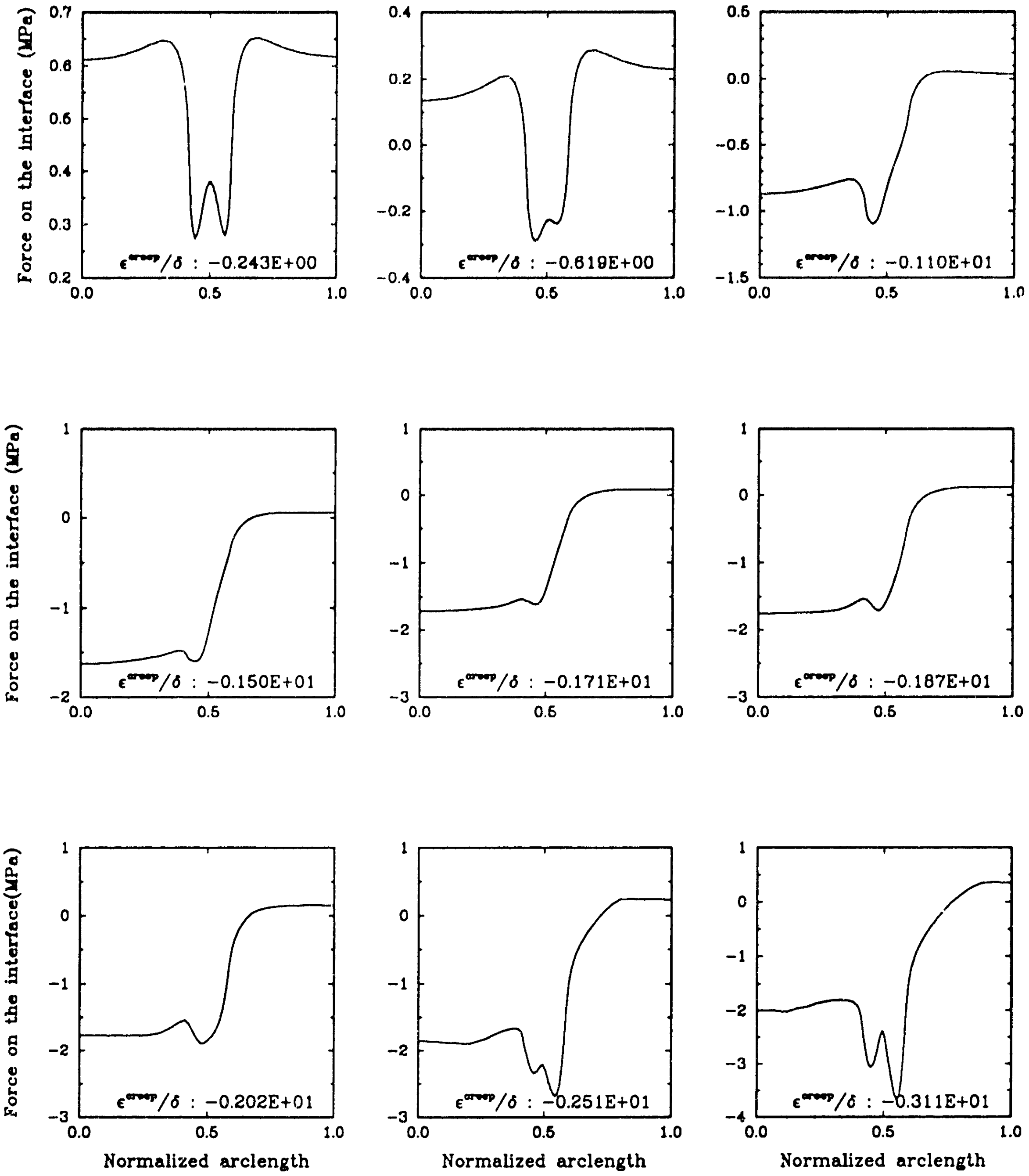


Figure 4.57 Evolution of the τ_n profile during the stress-annealing transient for case 3 (TCT).

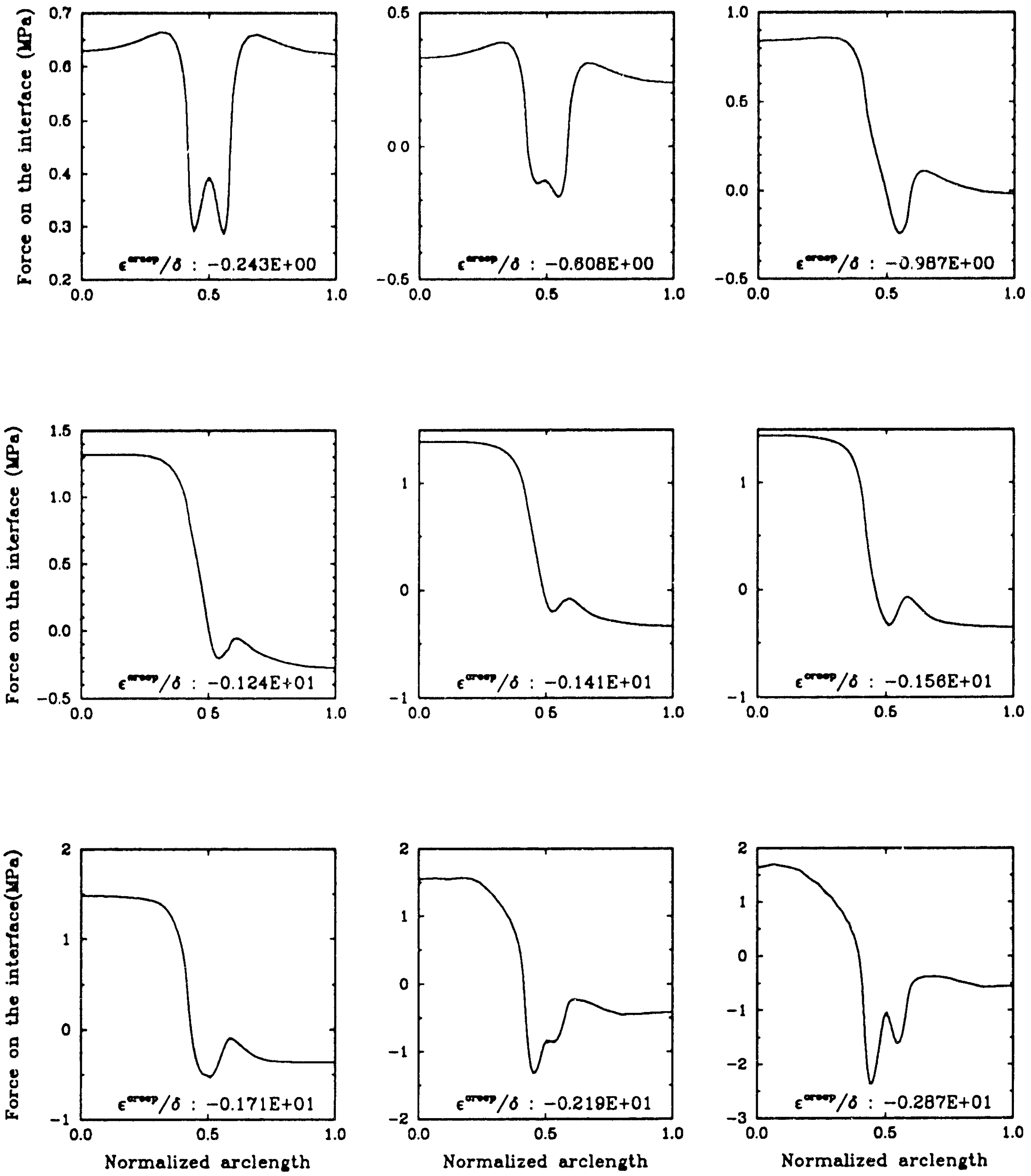


Figure 4.58 Evolution of the τ_n profile during the stress-annealing transient for case 4 (TCC).

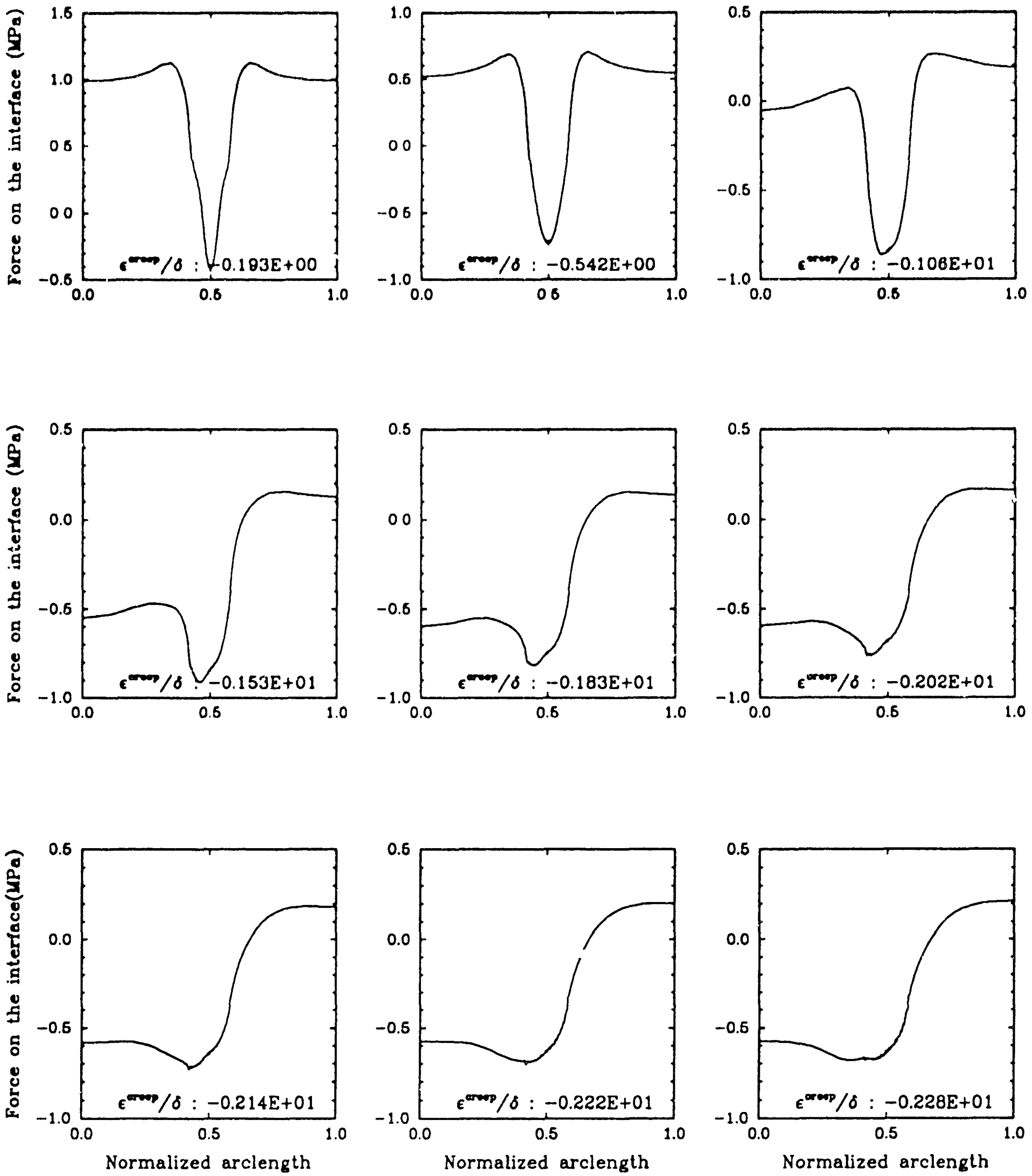


Figure 4.59 Evolution of the τ_n profile during the stress-annealing transient for case 5 (PLT).

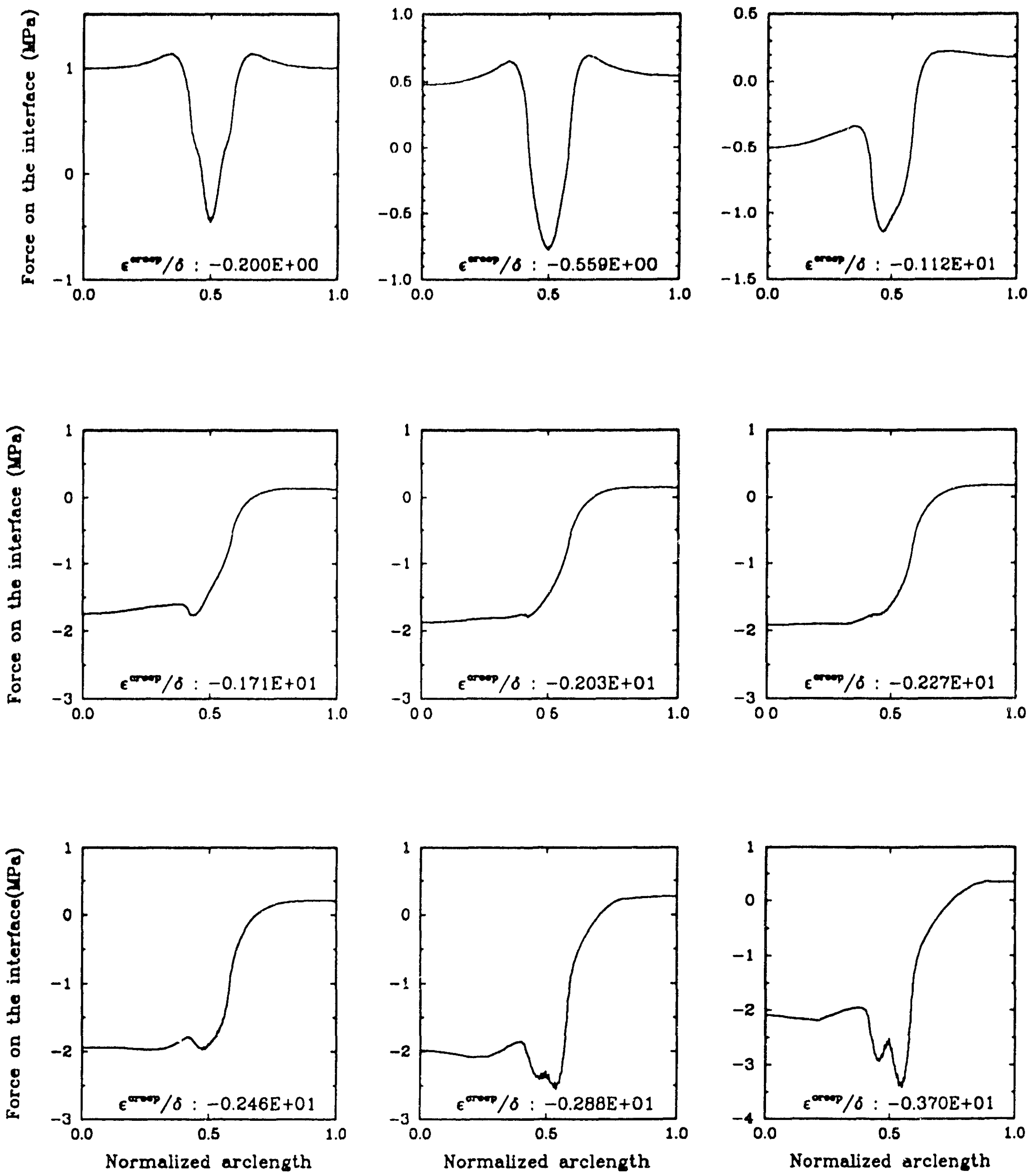


Figure 4.60 Evolution of the τ_n profile during the stress-annealing transient for case 6 (PHT).

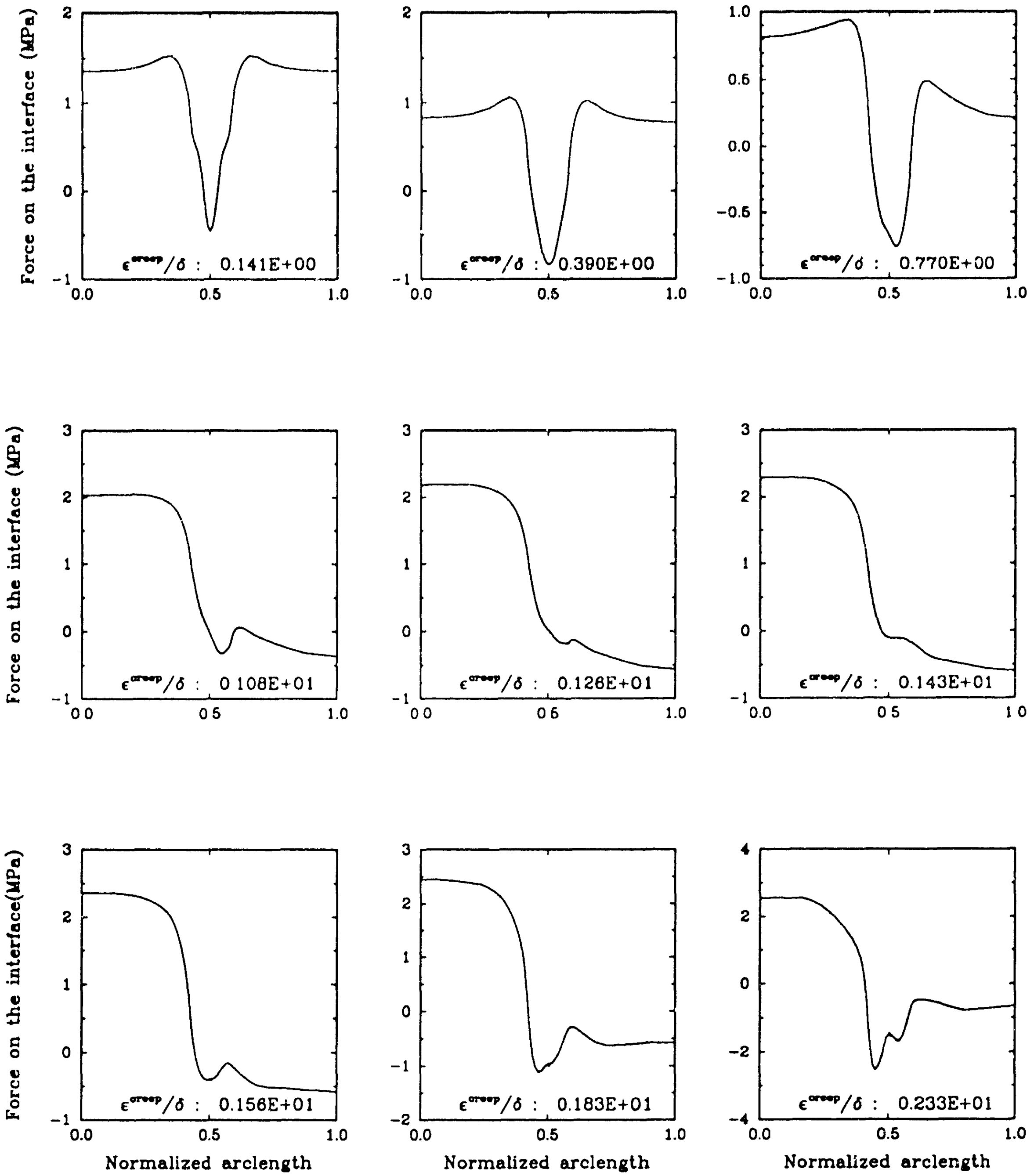


Figure 4.61 Evolution of the τ_n profile during the stress-annealing transient for case 7 (MNMT$_{INV}$).

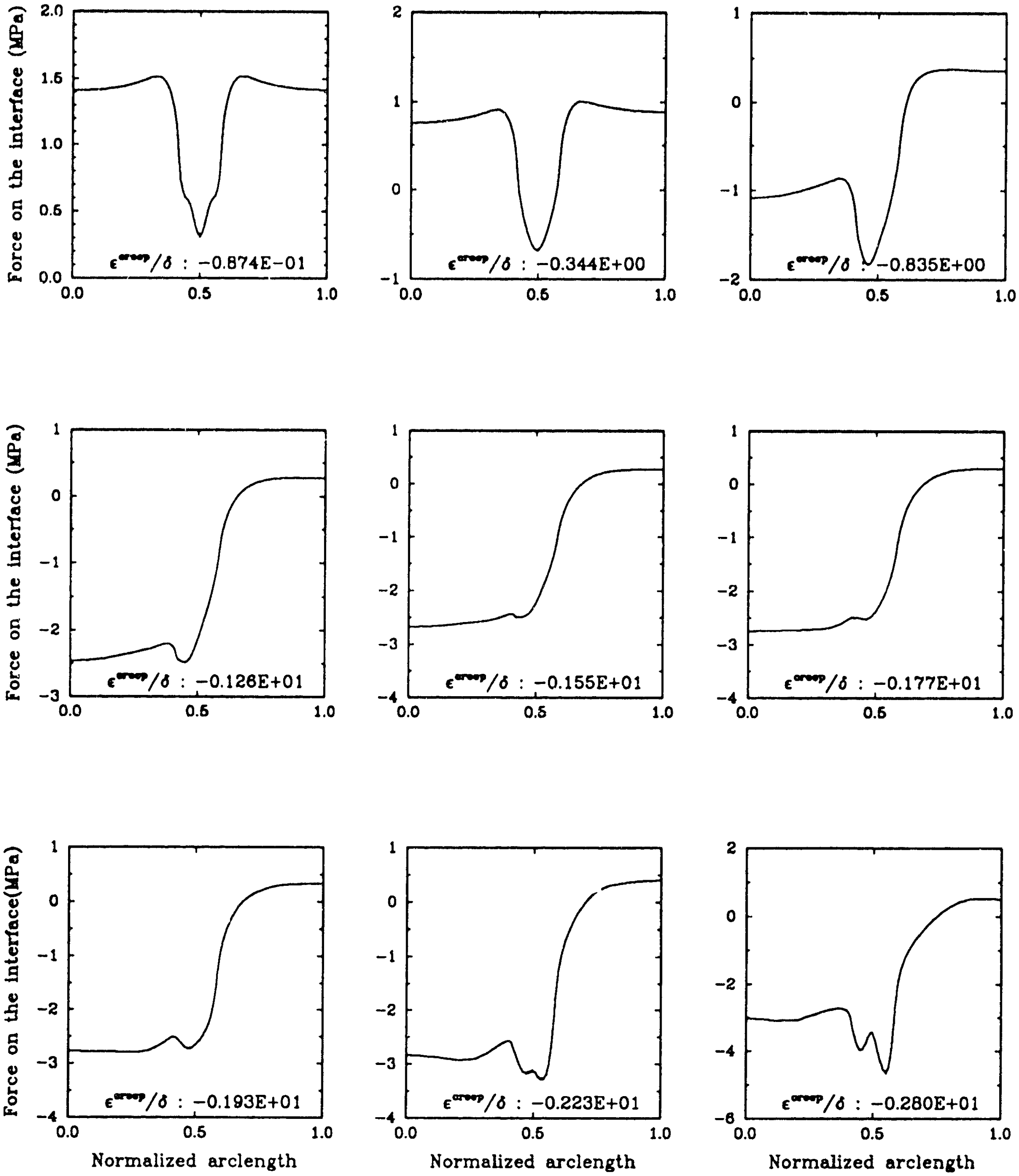


Figure 4.62 Evolution of the τ_n profile during the stress-annealing transient for case 8 (MNMT$_{NMS}$).

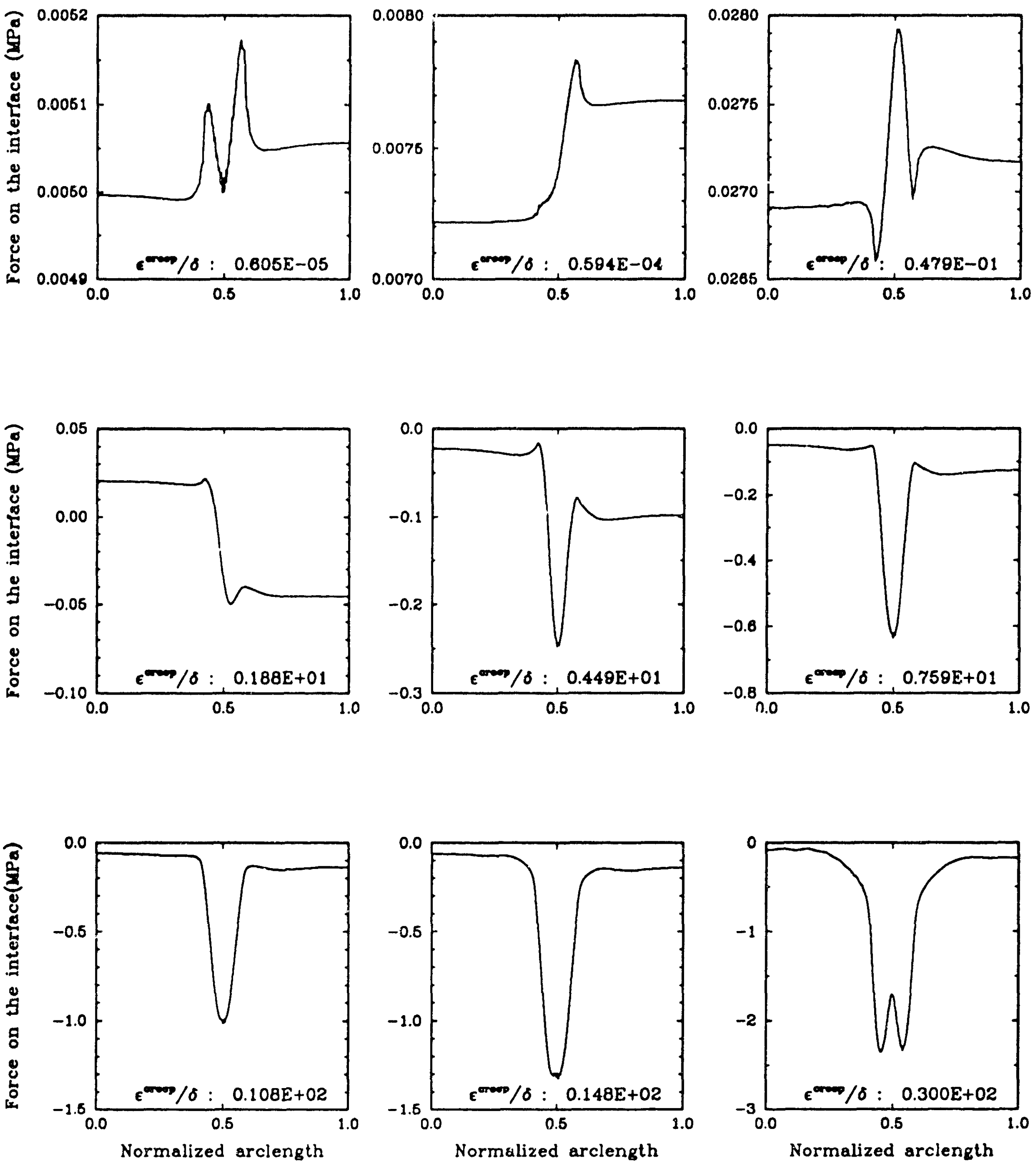


Figure 4.63 Evolution of the τ_n profile during the stress-annealing transient for case 9 (TCT$_{PMS}$).

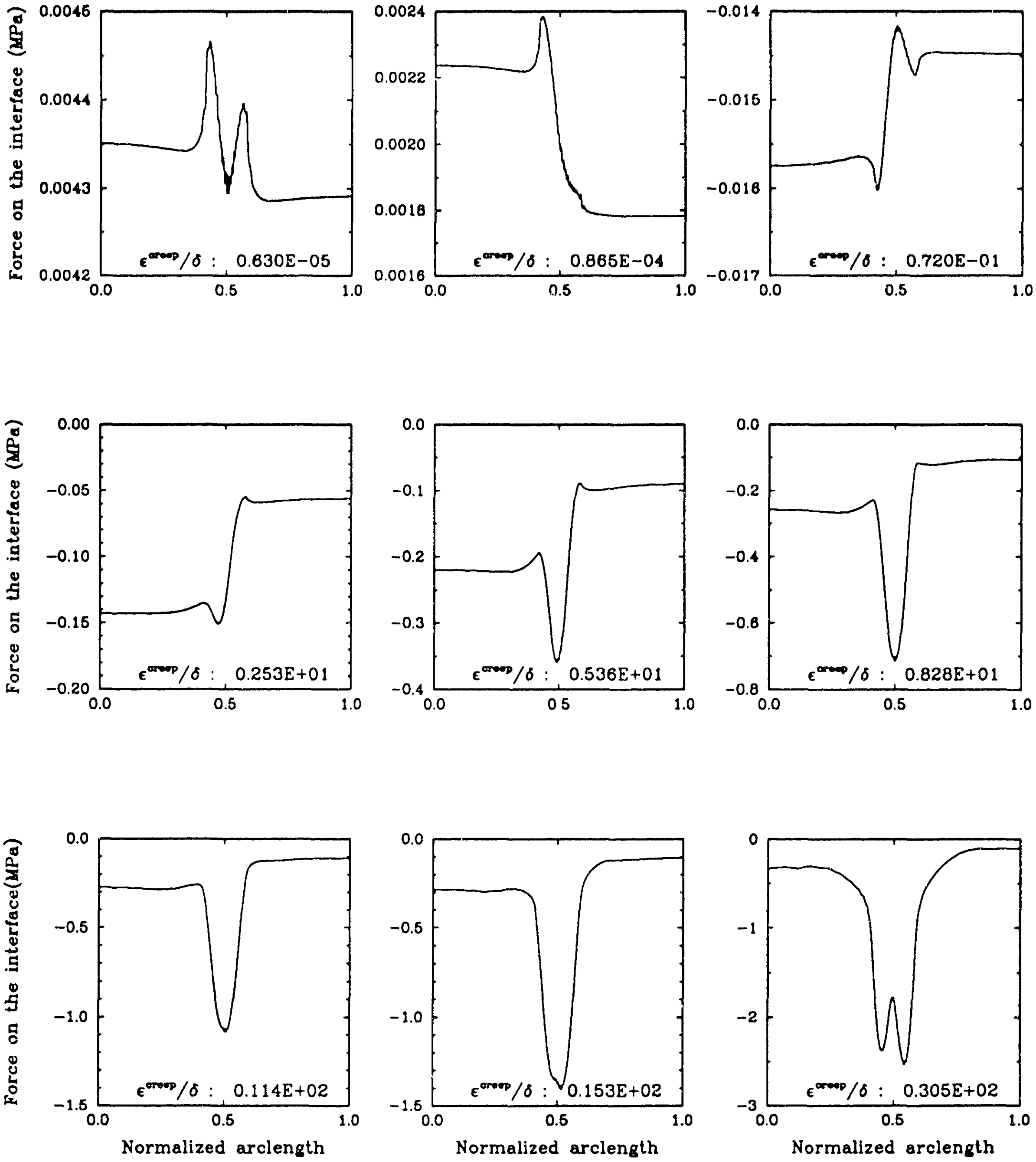


Figure 4.64 Evolution of the τ_n profile during the stress-annealing transient for case 10 (TCC$_{PMS}$).

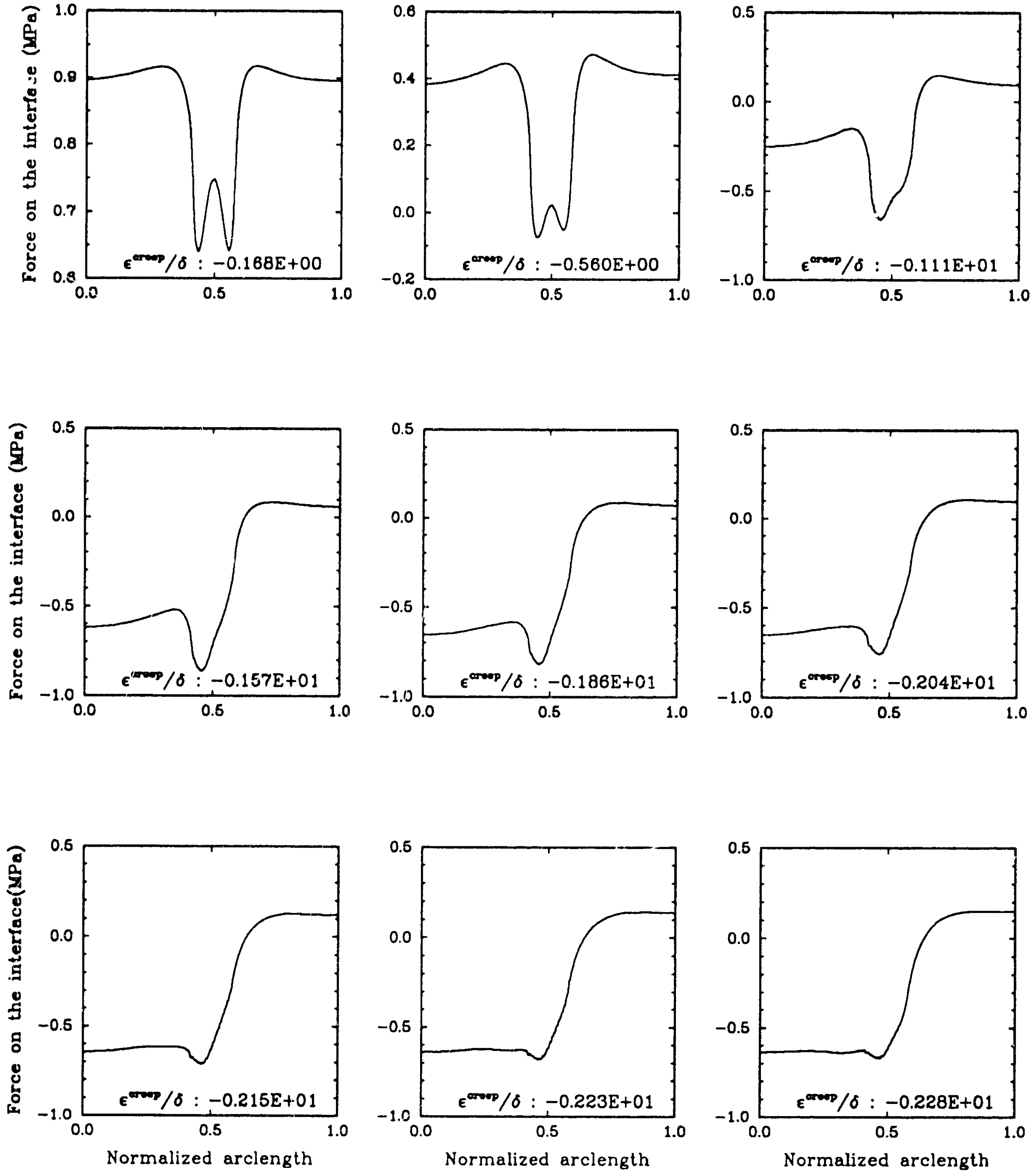


Figure 4.65 Evolution of the τ_n profile during the stress-annealing transient for case 11 (PLT$_{iso}$).

4.6 Discussion

As we have repeatedly mentioned, experimental observations show that the rafting process initiates in the primary stage of the creep transient.

Figure 4.66 shows a typical creep curve with corresponding micrographs illustrating the development of rafting as observed by MacKay and Ebert [22]. Figure 4.67 shows a similar curve obtained by Fredholm and Strudel [23] with the corresponding changes in microstructure.

From these and several other observations [17-24,27], we can infer that the driving force for directional coarsening sets in early in the transient and that the morphology evolution ensues within an extent of time which is determined by the kinetics of the process – in particular by the diffusion properties of the crystal [57]. We can then expect to be able to explain the tendency toward directional coarsening by analyzing the state of strain and stress in the crystal at the end of the primary creep.

The primary stage of creep corresponds to the formation of networks of dislocations which relieve the misfit stresses; in our continuum model, we can identify this stage by comparing the value of the average equivalent creep strain in the matrix $\overline{\epsilon}^{\rm creep}$ as defined in (4.5), with the initial value of the misfit, δ : we can assume that the misfit stresses are relieved when $\overline{\epsilon}^{\rm creep}$ and δ are of the same order of magnitude. Actually, from the results of our finite element analysis, we can conclude that the primary stage of creep is completed when $\overline{\epsilon}^{\rm creep}/\delta \simeq 2$.

Let's now analyze the state of stress and strain at this stage of the transient, and the corresponding effects on the $\tau_{\rm n}$ profile. We recall that the value of $\tau_{\rm n}$ is given by the expressions (2.31) that we repeat here for convenience:

$$\tau_{\rm n} = [W] - t_i[\frac{\partial u_i}{\partial n}]. \qquad (4.6)$$

We will first consider the case of an applied tensile load and we will afterward discuss the case of an applied compressive load.

(a) Applied Tensile Load

It is convenient to independently analyze the conditions on the "top" of the precipitates (corresponding to the channels normal to the applied load) and on the "side"" of the precipitates (corresponding to the channels parallel to the applied load).

- On the *top* of the precipitates, due to the high level of hydrostatic stress in the matrix (see Fig. 4.30(c)), the $t_i[\frac{\partial u_i}{\partial n}]$ term in expression (4.6) largely dominates. The $[W]$ term ranges between one and five percent of the $t_i[\frac{\partial u_i}{\partial n}]$ term for the analyzed alloys.

  The value of the component of the traction vector normal to the interface, $t_{\rm n}$, is large and positive, while the tangential component is negligible. The magnitude of the normal component of $[\frac{\partial \vec{u}}{\partial n}]$, which we will indicate as $[u_{\mathbf{n},\mathbf{n}}]$, is larger than the magnitude of the original misfit since the primary creep flow acts so as to enhance the jump in $u_{\mathbf{n},\mathbf{n}}$ across the interface, initially due to the lattice misfit (see Fig. 4.30(b)). Thus for *negative-misfit* alloys $[u_{\mathbf{n},\mathbf{n}}]$ is positive so that the force on the interface on the top of the precipitate results negative. The magnitude of the force scales with the applied load (which directly affects the magnitude of $t_{\mathbf{n}}$) and with the misfit (which directly affects $[u_{\mathbf{n},\mathbf{n}}]$). As the transient progresses past the primary stage, and the matrix starts creeping under the effect of the applied load, the creep flow keeps acting so as to increase $[u_{\mathbf{n},\mathbf{n}}]$ and therefore $\tau_{\mathbf{n}}$ will assume increasingly larger negative values.

Conversely, for *positive-misfit* alloys $[u_{\mathbf{n},\mathbf{n}}]$ is initially negative so that the force on the interface on the top of the precipitate results positive. The magnitude of $\tau_{\mathbf{n}}$ scales again with the applied load and with the misfit. As the transient progresses past the primary stage, the matrix starts creeping under the effect of the applied load. This process gives a positive contribution to $[u_{\mathbf{n},\mathbf{n}}]$ so that the magnitude of $\tau_{\mathbf{n}}$ starts to decrease and, for $\overline{\epsilon}^{\text{creep}}/\delta \simeq 4, \tau_{\mathbf{n}}$ becomes negative.

- On the *side* of the precipitates, since the traction vector is approximately one order of magnitude smaller than that of the top, the two terms in (4.6) are comparable and the magnitude of the force on the interface is much lower. The $[W]$ term always gives a negative contribution to $\tau_{\mathbf{n}}$ because the state of stress in the precipitate gives rise to an elastic energy level higher than that in the matrix channels, which are essentially in a state of moderate hydrostatic stress.

  The normal component of the traction vector, $t_{\mathbf{n}}$, is negative and the tangential component is negligible.

  For *negative-misfit alloys*, the $[u_{\mathbf{n},\mathbf{n}}]$ term at the end of primary creep is large and positive (for what concerns this term, the primary creep flow acts in the same sense of the misfit) so that the $(-t_i[\frac{\partial u_i}{\partial n}])$ term is positive and counterbalances the $[W]$ term.

  Thus for alloys characterized by a small negative misfit, the $[W]$ term initially dominates and $\tau_{\mathbf{n}}$ is negative, while for alloys with large negative misfit the $(-t_i[\frac{\partial u_i}{\partial n}])$ term dominates and $\tau_{\mathbf{n}}$ is positive (this is normally the case for most commercial alloys). As the transient progresses, the matrix material creeps under compressive stress so that negative contributions are added to $[u_{\mathbf{n},\mathbf{n}}]$ and thus to the force $\tau_{\mathbf{n}}$.

  For *positive-misfit* alloys, the $[u_{\mathbf{n},\mathbf{n}}]$ term at the end of the primary creep is large and negative so that the $(-t_i[\frac{\partial u_i}{\partial n}])$ term is negative as well and adds up with $[W]$ to give a negative value for $\tau_{\mathbf{n}}$. As the transient evolves, negative contributions are added to $[u_{\mathbf{n},\mathbf{n}}]$ so that $\tau_{\mathbf{n}}$ assumes increasingly larger negative values.

(b) Applied Compressive Loads

The patterns according to which τ_n evolves are essentially symmetrical to those for tensile loads due to the fact that the sign of the traction vector is inverted

- on the *top* of the precipitates, the $t_i[\frac{\partial u_i}{\partial n}]$ term dominates; t_n is large and negative while the tangential component of the traction vector is negligible.

  For *negative-misfit* alloys $[u_{n,n}]$ at the end of primary creep is positive so that the force on the interface results positive, and its magnitude scales with the applied load and the initial value of the misfit.

  As the transient evolves, the matrix material creeps under the applied compressive loads and negative contributions are added to $[u_{n,n}]$ so that the magnitude of τ_n starts to decrease and τ_n will eventually become negative.

  For *positive-misfit* alloys $[u_{n,n}]$ at the end of the primary creep is negative so that the force on the interface is negative as well. As the transient progresses past the primary stage of creep, the matrix creeps in compression under the applied load so that negative contributions are added to $[u_{n,n}]$ and τ_n will assume increasingly larger negative values.

- On the *side* of the precipitates the two terms in (4.6) are comparable and the magnitude of τ_n is much lower. The $[W]$ term always gives a negative contribution to τ_n. The normal component of the traction vector, t_n, is positive and the tangential component is negligible. For *negative-misfit* alloys, the $[u_{n,n}]$ term at the end of primary creep is positive so that the $(-t_i[\frac{\partial u_i}{\partial n}])$ term is negative and adds up with $[W]$ to give a negative value for τ_n. As the transient evolves, τ_n will assume increasingly larger negative values due to the positive contributions to $[u_{n,n}]$ as the matrix creep under tensile stress.

  For *positive-misfit* alloys, the $[u_{n,n}]$ term at the end of primary creep is negative so that the $(-t_i[\frac{\partial u_i}{\partial n}])$ term counterbalance the $[W]$ term and τ_n will be positive or negative depending on the value of the initial misfit. As the transient evolves negative contributions will be added to τ_n.

In Fig. 4.68, we give a schematic synoptic diagram showing the levels of τ_n on the top and on the side of the precipitates at the end of the primary stage of the creep transient for the possible combinations of tensile/compressive load and positive/negative misfit.

From this diagram we can see how under a tensile load a negative-misfit alloy will tend to exhibit a "type N" rafting behavior, while a positive-misfit alloy will tend to exhibit a "type P" rafting behavior and vice versa for a compressive load.

These simple patterns are indeed in agreement with all the available experimental data listed in Table 1.1.

The arrows in Fig. 4.68 indicate how the levels of τ_n will decrease as the creep transient evolves toward higher levels of creep strains under the effect of the applied loads. If we consider this further evolution of the τ_n profiles we can conclude that we can always expect to observe a "Type N" rafting behavior for a negative-misfit alloy under a tensile load and for a positive-misfit alloy under a compressive load. However, if we consider the case of negative-misfit alloys under compressive loads and positive-misfit alloys under tensile loads, we can expect to observe a marked tendency toward a "type P" rafting behavior only if we conduct our test so as to maintain the crystal at a low level of creep for the time needed by the kinetics of the process to accomplish, at least partially, the morphology evolution.

In fact, if we compare micrographs of "Type P" rafts and "Type N" rafts, the latter are generally characterized by a higher aspect ratio and a more regular structure.

With regard to the evolution of $\Delta\tau_n$, since the magnitude of τ_n essentially scales with the misfit, δ, and the applied stress, σ, and the sign of $\Delta\tau_n$ changes with the sign of δ and σ as shown in Fig. 4.68, we can expect that plots of the normalized quantity $\Delta\tau_n/\sigma\delta$ versus the magnitude of $\overline{\epsilon}^{creep}/\delta$ will show similar patterns for all the different alloys that we have analyzed. These curves are shown in Fig. 4.69, and it can be noted how all the data correlates within a very narrow band.[1]

This result also agrees with the experimental observations that indicate how the rate of directional coarsening scales with the lattice misfit and the applied stress (see paragraph 1.2): the process is accelerated when the driving force is increased. The hastening of the rafting process observed when the test temperature is increased and when the microstructure is refined, is most probably related to a reduction of the characteristic diffusion time, namely, to an increase in diffusivity and to a shortening of the diffusion path.

In Fig. 4.69 we can notice a substantial increase in $\Delta\tau_n$ within the very first stage of primary creep , when $0.5 \leq |\overline{\epsilon}^{creep}/\delta| \leq 2.0$. After this sharp rise, which corresponds to the period in which the misfit stresses are relieved, the creep flow in the matrix becomes dominated by the applied stress and the behavior of "Type N" evolutions and "Type P " evolutions branches: the driving force for "Type N" coarsening keeps increasing, while the force for "Type P" coarsening will eventually reach a saturation level.

These trends can be already qualitatively observed in the last portion of the curves in Fig 4.69. If we compare correspondent pairs of "Type N" and "Type P " plots (see Fig. 4.70) we can notice that the last section of the "Type P " curves (MNMT, TCC) dips below the last section of the "Type N" curves (MNMC , TCT) which still exhibit an upward curvature.

We have chosen not to continue our analysis beyond this stage because experimental observations show that at the end of primary creep, relevant morphological changes are already occurring: if we extend our analysis, based on the initial cuboidal

[1] Note that we have not included the data for case 9 (TCT_{PMS}) and case 10 (TCC_{PMS}). This is because the misfit that we have chosen for this hypothetical alloy is exceedingly small —the misfit strains and the elastic strains have comparable magnitude. It is obvious how, in the limit of zero misfit, quantities normalized by the misfit itself lose their significance.

shape of the precipitates, to the steady state creep we would not obtain a reliable simulation of this subsequent stage of the process.

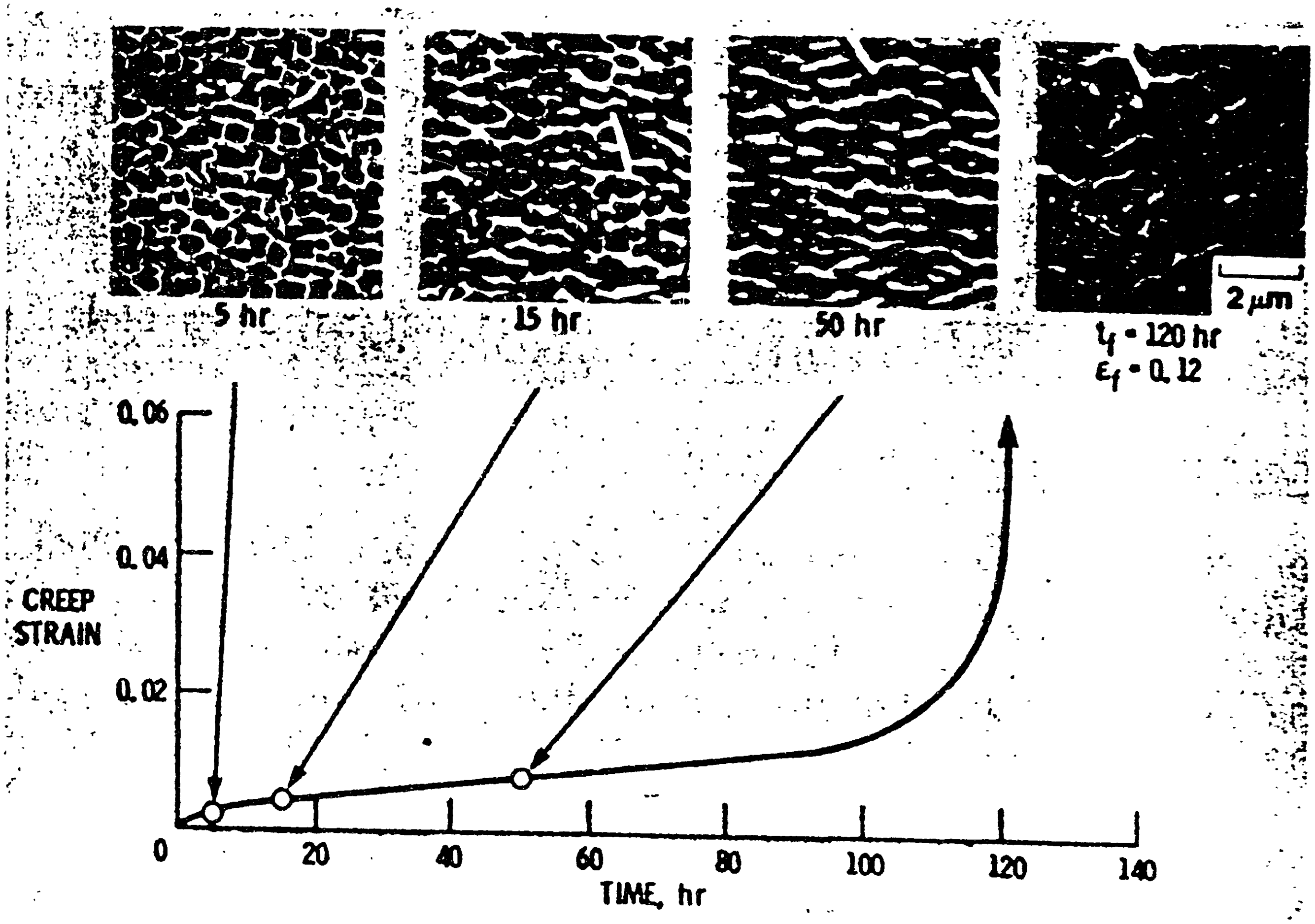


Figure 4.66 A typical creep curve with corresponding micrographs which show the development of directional coarsening [22].

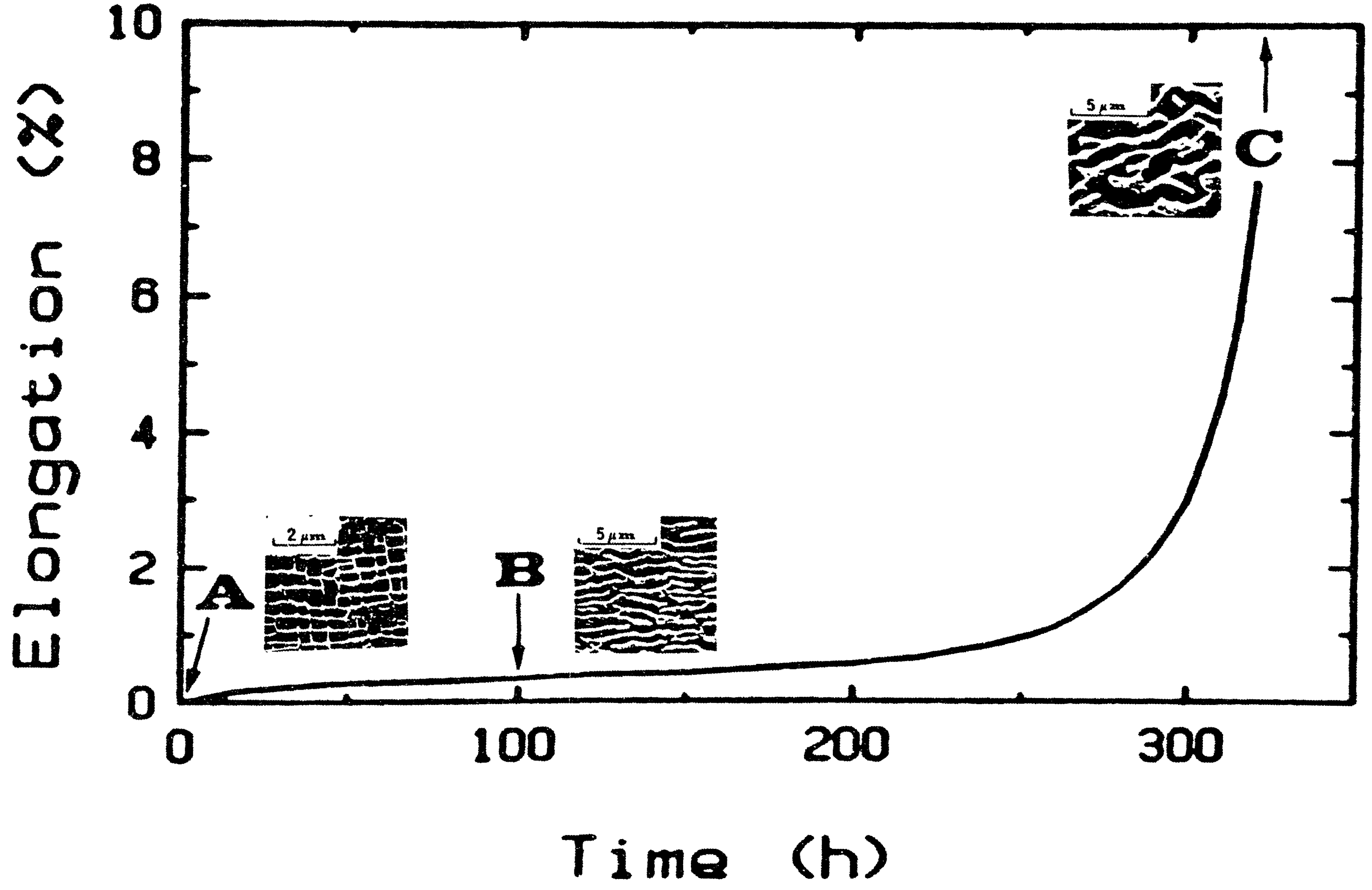


Figure 4.67 Creep curve and correspondent changes in microstructure [23].

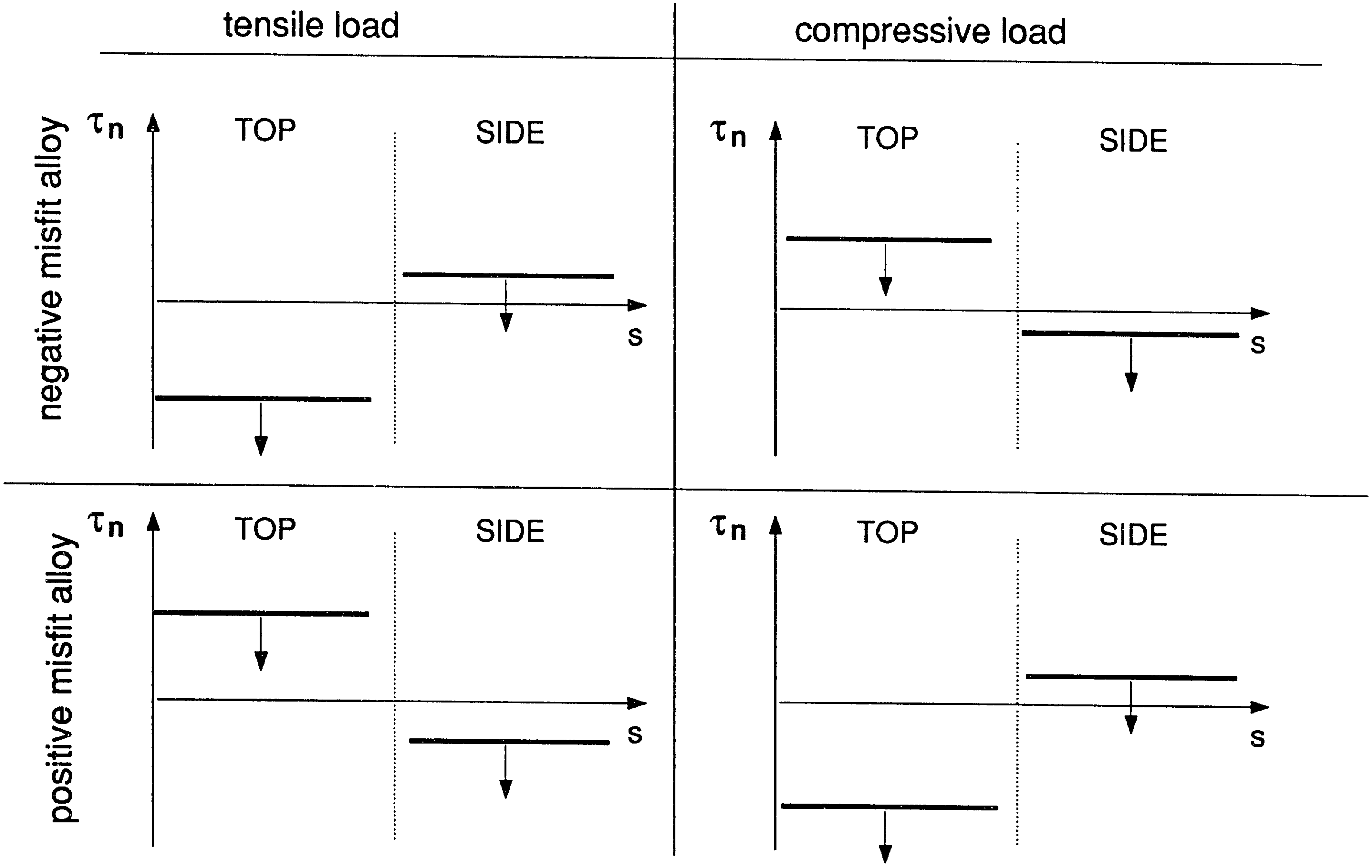


Figure 4.68 A schematic diagram showing the τ_n levels at the end of the primary stage of the creep transient. Vertical arrows indicate sense of expected evolution of τ_n with on-going creep.

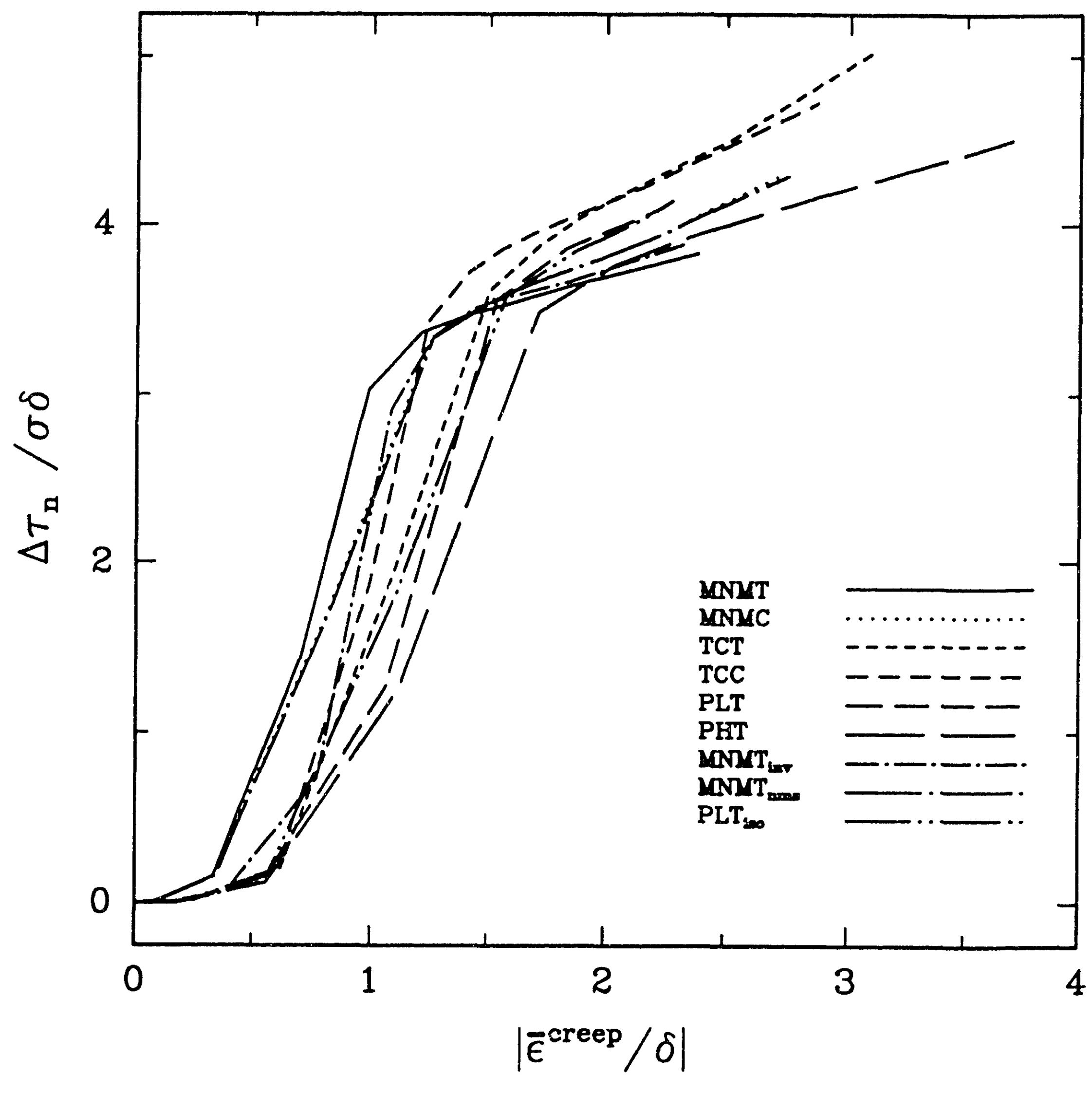


Figure 4.69 Evolution of $\Delta\tau_n/\sigma\delta$ for the analyzed creep transients

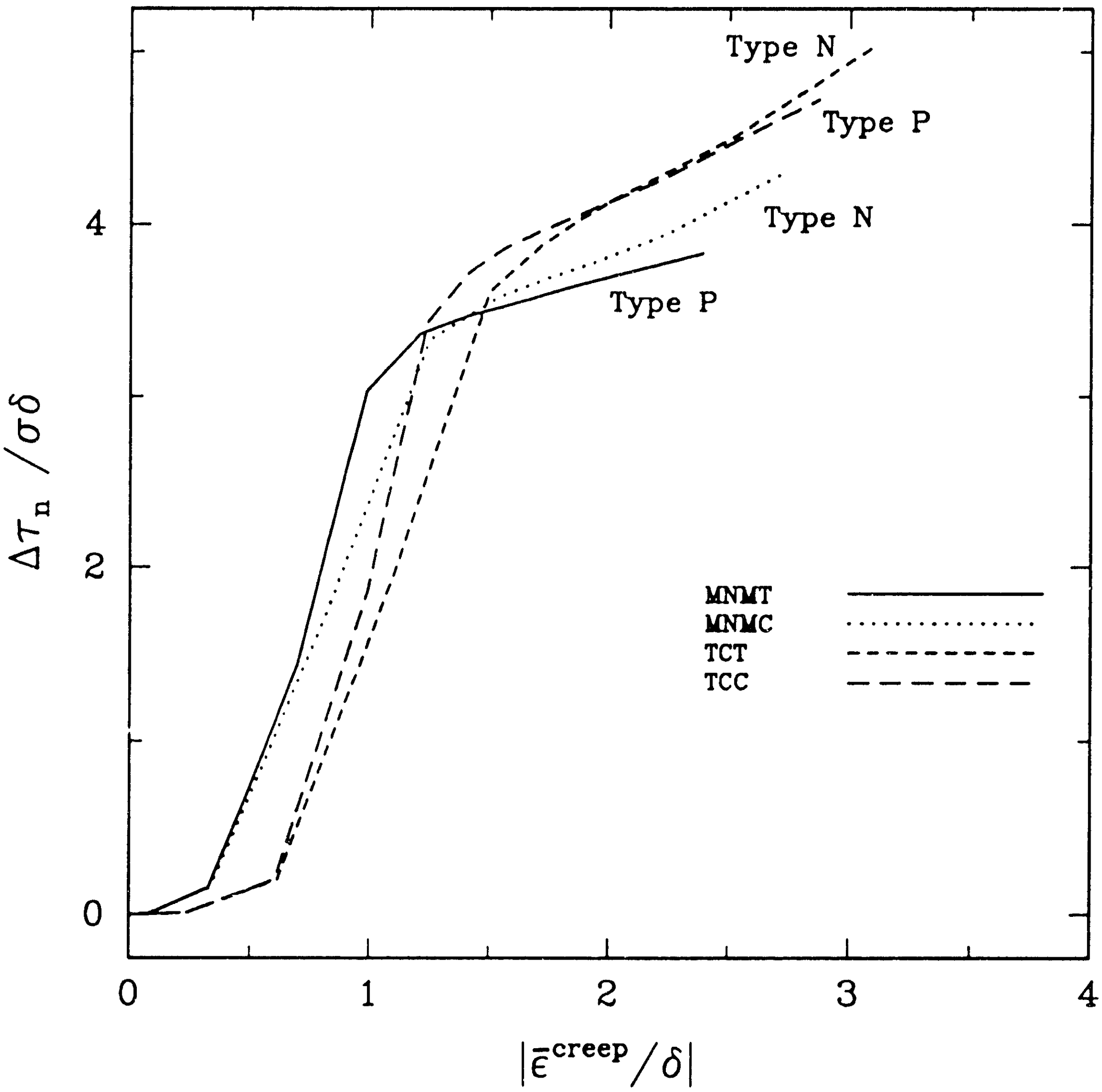


Figure 4.70 Comparison of "Type P" and "Type N" evolution curves for $\Delta\tau_n/\sigma\delta$

CHAPTER 5
CONCLUSIONS

5.1 Summary of Results

We have developed numerical techniques, in the framework of the finite element method, which allows us to evaluate local values of the generalized force acting on a material interface which is work-conjugate with the normal displacement of the interface itself.

This quantity is a direct measure of the tendency for the interface to migrate, and thus of the driving force for morphological evolutions of the microstructure.

We have applied these methods to the study of rafting in $\gamma - \gamma'$ Ni-superalloys.

The flexibility of the proposed methods has readily allowed us to closely model the actual microstructural morphology of the alloys and to account for the effects of applied boundary conditions, lattice misfit, elastic anisotropy and inelastic behavior of the crystals during the stress annealing transients.

We have modeled some experimental alloys, for which we have positively compared the indications of our model with the experimental data, and we have conducted a circumscribed parametric study which has allowed us to formulate a more general interpretation of the rafting phenomenon, which appears to give a satisfactory explanation for all the available experimental observations.

According to the results of our analysis, it is of fundamental importance, in modeling the rafting phenomenon, to consider the effects of the creep flow in the γ-matrix and in particular the evolution of the stress and strain fields during the primary creep stage of the stress-annealing transients.

Earlier attempts to interpret the rafting behavior of these alloys have indeed failed essentially because these effects were neglected.

In summary, the proposed methodology has proved itself successful for our particular application and appears suitable to be used in the analysis of parallel phenomena concerning microstructural evolutions in multi-phase materials.

5.2 Suggestions for Future Study

We can identify two main topics that we have addressed in our research. The first is the development of the numerical techniques *per se*; the second is the analysis of the rafting phenomenon. Regarding the development of numerical techniques for the evaluation of local forces acting on maternal interfaces, the first and most immediate development is the extension of the computer programs to cope with 3-D geometries. A more ambitious and substantial development, more closely related to the study of rafting, would be the implementation of a kinetic model to follow the evolution of the microstructure morphology. In such a model, the evaluation of the driving force would represent only one of several steps in the procedure.

With regard to the analysis of the rafting phenomenon, more involved creep models could be implemented, and the effects of modifications of the volume fraction of the precipitates could be investigated.

Finally, we should recognize that a continuum model for creep is not entirely adequate to model the discrete nature of motion and multiplication of dislocations in the narrow γ' channels.

The development of a discrete model for dislocation mechanics conceived so that it could be interactively superposed to the finite element solutions of continuum elastic behavior, would represent a substantial contribution to the analysis of this process as well as of other phenomena characterized by creep processes with a small length scale.

REFERENCES

1. M. V. Nathal: *Metal. Trans.***18A**, 1961, (1987).
2. G. A. Webster and C. P. Sullivan: *J. Inst. Metals*, **95**, 138, (1967).
3. Y. Nakada and W. C. Leslie: *Trans. ASM*, **60**, 223, (1961).
4. I. L. Mirkin and O. D. Kancheev: *Met. Sci. Heat Treatment*, **1**, 10, (1967).
5. C. P. Sullivan, G. A. Webster and B. J. Piearcey: *J. Inst. Metals*, **96**, 274, (1968).
6. J. R. Mihalisin and D. L. Pasquine: *Proc. International Symposium on Structural Stability in Superalloys*, Seven Springs, PA, **1**, 131 (1968).
7. W. Danesi and M. Donachie: *J. Inst. Metals*, **97**, 107, (1969).
8. R. G. Davies and T. L. Johnson: *Proc. Third. Landing Conf. on Ordered Alloys*, Baton Rouge, LA, 447, (1970).
9. R. A. Stevens and P. E. Flewitt: *Mat. Sci. Eng.*, **37**, 237, (1979).
10. G. A. Webseter and B. J. Pieracy: *Trans. ASM*, **59**, 847, (1966).
11. J. K. Tien and R. P. Gamble: *Met. Trans.*, **3**, 2157, (1972).
12. J. K. Tien and S. M. Copley: *Met. Trans*, **2**, 215, (1971).
13. J. K. Tien and S. M. Copley: *Met. Trans.*, **2**, 543, (1971).
14. A. Pineau: *Acta Met.*, **24**, 550, (1976).
15. T. Miyazaki, K. Nakamura and H. Mori: *J. Mat. Sci.*, **14**, 1827, (1979).
16. C. Carry and J. L. Strudel: *Acta. Met.*, **25**, 767, (1977).
17. C. Carry and J. L. Strudel: *Acta. Met.*, **26**, 859, (1978).
18. D. D. Pearson, F. D. Lemkey and B. H. Kear: *Proc. Fourth International Symposium on Superalloys*, Metals Park, OH, 513, (1980).
19. D. D. Pearson, B. H. Kear and F. D. Lemkey: **Creep and Fracture of Engineering Materials and Structure**, (B. Wilshire and D. R. Owen, Ed.), Pineridge Press, Ltd., Swansea, U. K., 213, (1983).
20. M. V. Nethal and L. J. Ebert: *Scripta Met.*, **17**, 115, a, (1983).
21. P. Caron and T. Khan: *Mat. Sci. Eng.*, **61**, 173, (1983).
22. R. A. MacKay and L. J. Ebert: **Superalloys 1984**, (M. Gell, et al., Ed.), 135, (1984).
23. A. Fredholm and J. L. Strudel: **Superalloys 1984**, (M. Gell, et al., Ed.), 211, (1984).
24. R. A. MacKay and L. J. Ebert: *Met. Trans.*, **16A**, 1968, (1985).
25. T. M. Pollock: "Creep Deformation in Nickel-base Superalloys Single Crystal", Ph.D. Thesis, Dept. Mat. Sci. Eng., MIT, (1989).

26. J. M. Oblak and B. H. Kear: *Trans. Quart. ASM*, **61**, 519, (1968).

27. M. V. Nathal, R. A. MacKay, R. G. Garlick: *Mat. Sci. Eng.*, **75**, 195, (1985).

28. D. A. Grose and G. S. Ansell: *Metall. Trans*, **A12**, 1631, (1981).

29. M. P. Arbuzov and I. A. Zelenkov: *Phys. Met. Metalloved*, **16**, 65, (1963).

30. **Superalloys II**, C.T. Sims, N. S. Stoloff, W. C. Hagel Ed., (1987).

31. J. K. Lee, D. M. Barnett and H. I. Aaronson: *Met. Trans.*, **8A**, 963, (1977).

32. A. F. Jankowski, E. M. Wingo and T. Tsakalakos: *Computer Simulation of Microstructural Evolution*, (D. J. Srolovitz, Ed.), MS-AIME, 125, (1985).

33. G. Faivre: *Phys. Stat. Sol*; **35**, 249, (1964).

34. W. C. Johnson: "The Elastic and Diffusional Interaction of Spherical Inhomogeneities in a Uniaxial Stress Field", in **Micromechanics Inhomogenitiy. Toshio Mura anniversary Volume**, (G. Weng, Ed.), Springer Verlag (1989).

35. J. C. Chang: "Elastic Energy Changes Accompanying Gamma-Prime Rafting in Nickel-Base Superalloys", Ph.D. Thesis, Dept. Mat. Sci. Eng., MIT, (1989).

36. J. D. Eshelby: *Phil. Trans.*, **A244**: 87 (1951).

37. J. D. Eshelby: "Energy Relations and the Energy Momentum Tensor in Continuum Mechanics", in **Inelastic Behavior of Solids**, (M. F. Kanninen, et al., Ed.), 77-115, McGraw-Hill, New York (1970).

38. T. Mura: **Micromechanics of Defects in Solids**, Martinus Nijhoff Publishers, Dordrecht (1987).

39. ABAQUS version 4.7, Hibbitt, Karlsson and Sorensen, Inc., Providence, R. I.

40. J. R. Rice: "Mathematical Analysis in the Mechanics of Fracture" in **Fracture: An Advanced Treatise**, (H. Liebowitz, Ed.), Academic Press, New York (1968).

41. J. R. Rice: *J. Appl. Mech.*, **35**, 379, (1968).

42. J. W. Hutchinson: *J. Mech. Phys. Solids*, **16**, 13, (1968).

43. J. R. Rice and G. F. Rosengren: *J. Mech. Phys. Solids*, **16**, 1, (1968).

44. C. F. Shih, B. Moran, T. Nakamura: *Int. J. Fracture*, **30**, 79, (1986).

45. J. D. Eshelby: "Calculation of Energy Release Rate" in **Prospects of Fracture Mechanics**, (Sih, Van Elst and Broek, Ed.), Noordhoff (1974).

46. T. Nakamura, C. F. Shih, L. B. Freund, *Int. J. Fracture*, **27**, 229, (1985).

47. D. M. Parks: *Int. J. Fracture*, **10**, 487, (1974).

48. D. M. Parks: *Comp. Meth. Appl. Mech. Eng.*, **12**, 353, (1977).

49. D. M. Parks: "Virtual Crack Extension: A General Finite Element Technique for the *J*-Integral Evaluation" in **Numerical Methods in Fracture Mechanics**, (Luxmore A. R. and Owen D. R. J., Ed.), University College of Wales, Swansea (1978).

50. T. K. Hellen: *Int. J. Numer. Method. Eng.*, **9**, 187, (1975).

51. H. G. deLorenzi: *Int. J. Fracture*, **19**, 183, (1982).

52. H. G. deLorenzi: "Energy Release Rate Calculations by the Finite Element Method", General Electric Company, TIS, Rep. 82 CRD205 (1982).

53. F. Z. Li, C. F. Shih, A. Needleman: *Engineering Fracture Mechanics*, **21**, 405, (1985).

54. B. Moran and C. F. Shih: *Engineering Fracture Mechanics*, **27**, 615, (1987).

55. J. N. Goodier: *Trans. ASME*, **55**, 39, (1933).

56. W. R. Johnson, C. R. Barrett, W.D. Nix: *Metall. Trans.*, **3A**, 693, (1972).

57. F. G. Haubensak: S. M. Thesis, Dept. Mat. Sci. Eng., M.I.T., (1990).

58. F. A. McClintock and A. S. Argon: **Mechanical Behavior of Materials**, Addison-Wesley, (1966).

APPENDIX I: THE COMPUTER PROGRAM POSTABQ

```
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          P R O G R A M   P O S T A B Q                         *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      PROGRAM POSTABQ
C
      INCLUDE 'post_common'
C
C     Common for abaqus routines
C
      COMMON / NAME / FILE_NAME
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513), JRRAY(2,513),LRUNIT(2,1)
      EQUIVALENCE(ARRAY(1),JRRAY(1,1))
C
C     File Handling
C
      CHARACTER*20 FILN
      CHARACTER*32 FILINP
      CHARACTER*25 FILCHK
      CHARACTER*32 FILOUT
      CHARACTER*25 FILGEO
      CHARACTER*32 FILDAT
      CHARACTER*32 FILP01
      CHARACTER*32 FILP02
      CHARACTER*32 FILP03
      CHARACTER*32 FILP04
      CHARACTER*32 FILP05
      CHARACTER*32 FILP06
      CHARACTER*32 FILP07
      CHARACTER*32 FILP08
      CHARACTER*32 FILP09
      CHARACTER*32 FILP10
      CHARACTER*32 FILP11
      CHARACTER*32 FILP12
      CHARACTER*32 FILP13
      CHARACTER*32 FILP14
      CHARACTER*32 FILP15
      CHARACTER*32 FILP16
      CHARACTER*32 FILP17
      CHARACTER*32 FILP18
      CHARACTER*32 FILP19
      CHARACTER*25 FILE_NAME
      CHARACTER*6  TINP
      CHARACTER*6  TABQ
      CHARACTER*6  TCHK
      CHARACTER*6  TOUT
      CHARACTER*6  TGEO
      CHARACTER*6  TDAT
      CHARACTER*6  TP01
      CHARACTER*6  TP02
      CHARACTER*6  TP03
      CHARACTER*6  TP04
      CHARACTER*6  TP05
      CHARACTER*6  TP06
      CHARACTER*6  TP07
      CHARACTER*6  TP08
      CHARACTER*6  TP09
      CHARACTER*6  TP10
      CHARACTER*6  TP11
      CHARACTER*6  TP12
      CHARACTER*6  TP13
      CHARACTER*6  TP14
      CHARACTER*6  TP15
      CHARACTER*6  TP16
      CHARACTER*6  TP17
      CHARACTER*6  TP18
      CHARACTER*6  TP19
      CHARACTER*1  RESP
C
      DATA TINP /'.p.inp'/
      DATA TABQ /'.p.abq'/
      DATA TCHK /'.p.chk'/
      DATA TOUT /'.p.out'/
      DATA TGEO /'.p.geo'/
      DATA TDAT /'.p.dat'/
      DATA TP01 /'.p.p01'/
      DATA TP02 /'.p.p02'/
      DATA TP03 /'.p.p03'/
      DATA TP04 /'.p.p04'/
      DATA TP05 /'.p.p05'/
      DATA TP06 /'.p.p06'/
      DATA TP07 /'.p.p07'/
      DATA TP08 /'.p.p08'/
      DATA TP09 /'.p.p09'/
      DATA TP10 /'.p.p10'/
      DATA TP11 /'.p.p11'/
      DATA TP12 /'.p.p12'/
      DATA TP13 /'.p.p13'/
      DATA TP14 /'.p.p14'/
      DATA TP15 /'.p.p15'/
      DATA TP16 /'.p.p16'/
      DATA TP17 /'.p.p17'/
      DATA TP18 /'.p.p18'/
      DATA TP19 /'.p.p19'/
C
C
      PRINT *,'* * *           PROGRAM POSTABQ               * * *'
      PRINT *,' '
      PRINT *,'           POSTPROCESSOR FOR ABAQUS FILE 8'
      PRINT *,''
      PRINT *,'ABAQUS OUTPUT (FILE 8)  MUST BE NAMED Job_name.p.abq'
      PRINT *,'POSTABQ INPUT FILE      MUST BE NAMED Job_name.p.inp'
      PRINT *,'POSTABQ DATA FILE       MUST BF NAMED Job_name.p.dat'
      PRINT *,'POSTABQ OUTPUT FILE     WILL BE NAMED Job_name.p.out'
      PRINT *,'POSTABQ CHECK FILE      WILL BE NAMED Job_name.p.chk'
      PRINT *,'POSTABQ GEOM. OUT. FILE WILL BE NAMED Job_name.p.geo'
      PRINT *,'POSTABQ PLOT FILES      WILL BE NAMED Job_name.p**'
      PRINT *,''
C
C
C
      PRINT 1000,' Please enter the Job_name (MAX 20 Char.)'
      READ(*,2000) INM,FILN
C
C
      FILE_NAME(1:INM) = FILN(1:INM)
      FILE_NAME(INM+1:INM+6) = TABQ(1:6)
C
      FILINP(1:INM) = FILN(1:INM)
      FILINP(INM+1:INM+6) = TINP(1:6)
C
      FILDAT(1:INM) = FILN(1:INM)
      FILDAT(INM+1:INM+6) = TDAT(1:6)
C
      FILCHK(1:INM) = FILN(1:INM)
      FILCHK(INM+1:INM+6) = TCHK(1:6)
C
      FILOUT(1:INM) = FILN(1:INM)
      FILOUT(INM+1:INM+6) = TOUT(1:6)
C
      FILGEO(1:INM) = FILN(1:INM)
      FILGEO(INM+1:INM+6) = TGEO(1:6)
C
      FILP01(1:INM) = FILN(1:INM)
      FILP01(INM+1:INM+6) = TP01(1:6)
      FILP02(1:INM) = FILN(1:INM)
      FILP02(INM+1:INM+6) = TP02(1:6)
      FILP03(1:INM) = FILN(1:INM)
      FILP03(INM+1:INM+6) = TP03(1:6)
      FILP04(1:INM) = FILN(1:INM)
      FILP04(INM+1:INM+6) = TP04(1:6)
      FILP05(1:INM) = FILN(1:INM)
      FILP05(INM+1:INM+6) = TP05(1:6)
      FILP06(1:INM) = FILN(1:INM)
      FILP06(INM+1:INM+6) = TP06(1:6)
      FILP07(1:INM) = FILN(1:INM)
      FILP07(INM+1:INM+6) = TP07(1:6)
      FILP08(1:INM) = FILN(1:INM)
      FILP08(INM+1:INM+6) = TP08(1:6)
      FILP09(1:INM) = FILN(1:INM)
      FILP09(INM+1:INM+6) = TP09(1:6)
      FILP10(1:INM) = FILN(1:INM)
      FILP10(INM+1:INM+6) = TP10(1:6)
      FILP11(1:INM) = FILN(1:INM)
      FILP11(INM+1:INM+6) = TP11(1:6)
      FILP12(1:INM) = FILN(1:INM)
      FILP12(INM+1:INM+6) = TP12(1:6)
      FILP13(1:INM) = FILN(1:INM)
      FILP13(INM+1:INM+6) = TP13(1:6)
      FILP14(1:INM) = FILN(1:INM)
      FILP14(INM+1:INM+6) = TP14(1:6)
      FILP15(1:INM) = FILN(1:INM)
      FILP15(INM+1:INM+6) = TP15(1:6)
      FILP16(1:INM) = FILN(1:INM)
      FILP16(INM+1:INM+6) = TP16(1:6)
      FILP17(1:INM) = FILN(1:INM)
      FILP17(INM+1:INM+6) = TP17(1:6)
      FILP18(1:INM) = FILN(1:INM)
      FILP18(INM+1:INM+6) = TP18(1:6)
      FILP19(1:INM) = FILN(1:INM)
      FILP19(INM+1:INM+6) = TP19(1:6)
C
      OPEN(UNIT = 10,FILE = FILINP,STATUS = 'OLD',ERR=10)
      GO TO 20
 10   PRINT*,' '
      PRINT*,' '
      PRINT*,'Unable to open file : ',FILINP
      PRINT*,' '
      PRINT*,' '
      PRINT 1000,' Do you want to try again? [Y]'
      READ (*,2000)INUTIL, RESP
      IF (RESP.EQ.'N'.OR.RESP.EQ.'n') STOP
      PRINT 1000,' Choose another Job_name '
      READ (*,2000) INM,FILN

C
      FILE_NAME(1:INM) = FILN(1:INM)
      FILE_NAME(INM+1:INM+6) = TABQ(1:6)
      FILE_NAME(INM+7:25) = ' '
C
      FILINP(1:INM) = FILN(1:INM)
      FILINP(INM+1:INM+6) = TINP(1:6)
      FILINP(INM+7:32) = ' '
C
      FILDAT(1:INM) = FILN(1:INM)
      FILDAT(INM+1:INM+6) = TDAT(1:6)
      FILDAT(INM+7:32) = ' '
C
      FILCHK(1:INM) = FILN(1:INM)
      FILCHK(INM+1:INM+6) = TCHK(1:6)
      FILCHK(INM+7:25) = ' '
C
      FILOUT(1:INM) = FILN(1:INM)
      FILOUT(INM+1:INM+6) = TOUT(1:6)
      FILOUT(INM+7:32) = ' '
C
      FILGEO(1:INM) = FILN(1:INM)
      FILGEO(INM+1:INM+6) = TGEO(1:6)
      FILGEO(INM+7:25) = ' '
C
      FILP01(1:INM) = FILN(1:INM)
      FILP01(INM+1:INM+6) = TP01(1:6)
      FILP01(INM+7:32) = ' '
      FILP02(1:INM) = FILN(1:INM)
      FILP02(INM+1:INM+6) = TP02(1:6)
      FILP02(INM+7:32) = ' '
      FILP03(1:INM) = FILN(1:INM)
      FILP03(INM+1:INM+6) = TP03(1:6)
```

```
      FILP03(INM+7:32) = ' '
      FILP04(1:INM) = FILN(1:INM)
      FILP04(INM+1:INM+6) = TP04(1:6)
      FILP04(INM+7:32) = ' '
      FILP05(1:INM) = FILN(1:INM)
      FILP05(INM+1:INM+6) = TP05(1:6)
      FILP05(INM+7:32) = ' '
      FILP06(1:INM) = FILN(1:INM)
      FILP06(INM+1:INM+6) = TP06(1:6)
      FILP06(INM+7:32) = ' '
      FILP07(1:INM) = FILN(1:INM)
      FILP07(INM+1:INM+6) = TP07(1:6)
      FILP07(INM+7:32) = ' '
      FILP08(1:INM) = FILN(1:INM)
      FILP08(INM+1:INM+6) = TP08(1:6)
      FILP08(INM+7:32) = ' '
      FILP09(1:INM) = FILN(1:INM)
      FILP09(INM+1:INM+6) = TP09(1:6)
      FILP09(INM+7:32) = ' '
      FILP10(1:INM) = FILN(1:INM)
      FILP10(INM+1:INM+6) = TP10(1:6)
      FILP10(INM+7:32) = ' '
      FILP11(1:INM) = FILN(1:INM)
      FILP11(INM+1:INM+6) = TP11(1:6)
      FILP11(INM+7:32) = ' '
      FILP12(1:INM) = FILN(1:INM)
      FILP12(INM+1:INM+6) = TP12(1:6)
      FILP12(INM+7:32) = ' '
      FILP13(1:INM) = FILN(1:INM)
      FILP13(INM+1:INM+6) = TP13(1:6)
      FILP13(INM+7:32) = ' '
      FILP14(1:INM) = FILN(1:INM)
      FILP14(INM+1:INM+6) = TP14(1:6)
      FILP14(INM+7:32) = ' '
      FILP15(1:INM) = FILN(1:INM)
      FILP15(INM+1:INM+6) = TP15(1:6)
      FILP15(INM+7:32) = ' '
      FILP16(1:INM) = FILN(1:INM)
      FILP16(INM+1:INM+6) = TP16(1:6)
      FILP16(INM+7:32) = ' '
      FILP17(1:INM) = FILN(1:INM)
      FILP17(INM+1:INM+6) = TP17(1:6)
      FILP17(INM+7:32) = ' '
      FILP18(1:INM) = FILN(1:INM)
      FILP18(INM+1:INM+6) = TP18(1:6)
      FILP18(INM+7:32) = ' '
      FILP19(1:INM) = FILN(1:INM)
      FILP19(INM+1:INM+6) = TP19(1:6)
      FILP19(INM+7:32 ) = ' '
C
C
      OPEN(UNIT = 10,FILE = FILINP,STATUS = 'OLD',ERR=10)
C
 20    CONTINUE
C
      OPEN(UNIT = 11,FILE = FILDAT)
      OPEN(UNIT = 13,FILE = FILCHK)
      OPEN(UNIT = 14,FILE = FILOUT)
      OPEN(UNIT = 15,FILE = FILGEO)
      OPEN(UNIT = 16,FILE = FILP01)
      OPEN(UNIT = 17,FILE = FILP02)
      OPEN(UNIT = 18,FILE = FILP03)
      OPEN(UNIT = 19,FILE = FILP04)
      OPEN(UNIT = 20,FILE = FILP05)
      OPEN(UNIT = 21,FILE = FILP06)
      OPEN(UNIT = 22,FILE = FILP07)
      OPEN(UNIT = 23,FILE = FILP08)
      OPEN(UNIT = 24,FILE = FILP09)
      OPEN(UNIT = 25,FILE = FILP10)
      OPEN(UNIT = 26,FILE = FILP11)
      OPEN(UNIT = 27,FILE = FILP12)
      GPEN(UNIT = 28,FILE = FILP13)
      OPEN(UNIT = 30,FILE = FILP15)
      OPEN(UNIT = 31,FILE = FILP16)
      OPEN(UNIT = 32,FILE = FILP17)
      OPEN(UNIT = 33,FILE = FILP18)
      OPEN(UNIT = 34,FILE = FILP19)
C
C
C
      NORI = 10
      NORD = 11
      NOWC = 13
      NOWO = 14
      NOWG = 15
      NOWP(1) = 16
      NOWP(2) = 17
      NOWP(3) = 18
      NOWP(4) = 19
      NOWP(5) = 20
      NOWP(6) = 21
      NOWP(7) = 22
      NOWP(8) = 23
      NOWP(9) = 24
      NOWP(10) =25
      NOWP(11) =26
      NOWP(12) =27
      NOWP(13) =28
      NOWP(14) =29
      NOWP(15) =30
      NOWP(16) =31
      NOWP(17) =32
      NOWP(18) =33
      NOWP(19) =34
C
C
      NNELT = 8
      NDIM = 2
      KERROR = 0
C
C     Accessing file 8
C
      NRU = 1
      LRUNIT(1,1) = 8
      LRUNIT(2,1) = 2
      CALL INITPF(NRU,LRUNIT,LOUTF)
      JUNIT = 8
      CALL DBRNU(JUNIT)
C
C     Geometry input. Connectivity matrices
C
      CALL GEOINP
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     Path geometry. Curvilinear coordinate and normal to the path.
C
      CALL GEOPAT
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     Shape function matrices
C
```

```
      CALL PRESFN
C
      IF(KERROR.GT.0) GO TO 200
C
      IF(KPSTOP.EQ.1) GO TO 300
C
C
C     Input the control variables of the procedure
C
      CALL CTRINP
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     processing the variables for all the step/increment required by user
C
      DO 150 NOUT = 1,NTOUT
C
C            reset to zero flags and variables
C
             CALL RESETV
C
      WRITE(NOWO,4000) NKSTEP(NOUT),NKINCR(NOUT)
C
             DO 100 IVAR = 1, NVAR
C
                    CALL PROGES(NOUT,IVAR)
C
                    IF(KERROR.GT.0) GO TO 200
C
 100         CONTINUE
C
C
C     Executing the user routine
C
      IF (JSUB.EQ.0) GO TO 300
C
      CALL USRSUB(NOUT,FILN)
C
      IF(KERROR.GT.0) GO TO 200
C
C
 150  CONTINUE
C
      STOP
C
C
 200  CONTINUE
C
      WRITE (NOWC,3000) KERROR
      WRITE (NOWO,3000) KERROR
C
 300  CONTINUE
C
 1000 FORMAT($,A,’ : ’)
 2000 FORMAT(Q,A)
 3000 FORMAT(1H1,//,10X,’* * *  ERROR DETECTED. PROGRAM STOP : KERROR’,
     &’= ’,I5,’ * * *’)
 4000 FORMAT(1H1,////,20X,’* * * STEP’,I3,’ INCR.’,I4,’ * * *’,//)
C
      STOP
      END
C
C
C
C     NOTE : Routines follow in alphabetical order
C
C
```

```
C
C     **************************************************************
C     *                                                            *
C     *                                                            *
C     *          F U N C T I O N     A L C A L 1                   *
C     *                                                            *
C     *                                                            *
C     **************************************************************
C
C
      FUNCTION ALCAL1(A,B,C)
C
C     Evaluate the integral for -1<X<0  of SQRT(C*X**2 + B*X + A)
C
      IF ( C ) 100,200,300
C
C
C
C     Negative C
C
 100  CONTINUE
C
      TERM1 = B*(SQRT(A))-(B-2*C)*SQRT(A-B+C)
      DELTA = B**2 - 4*A*C
      TERM2 = (DELTA/(2*SQRT(-C)))*(ASIN(B/(SQRT(DELTA)))-
     &                           ASIN((B-2*C)/(SQRT(DELTA))))
C
      ALCAL1 = (TERM1 +TERM2) /(4*C)
C
      GO TO 400
C
C
C
C     Zero  C
C
 200  CONTINUE
C
      IF(B.EQ.0) ALCAL1 = SQRT(A)
C
      IF(B.NE.0) ALCAL1 = (2*(A**1.5-(A-B)**1.5))/(3*B)
C
      GO TO 400
C
C
C
```

```
C     Positive C
C
 300  CONTINUE
C
      TERM1 = B*(SQRT(A))-(B-2*C)*SQRT(A-B+C)
      TERM2 = ((4*A*C-B**2)/(2*SQRT(C)))*ALOG((2*SQRT(C*A)+B)/
     &                                  (2*SQRT(C*(A-B+C))-2*C+B))
C
      ALCAL1 = (TERM1+TERM2) /(4*C)
C
C
 400  CONTINUE
C
      RETURN
      END
```

```
C
C
C
C     ********************************************************************
C     *                                                                  *
C     *                                                                  *
C     *           F U N C T I O N      A L C A L 2                       *
C     *                                                                  *
C     *                                                                  *
C     ********************************************************************
C
C
      FUNCTION ALCAL2(A,B,C)
C
C     Evaluate the integral for 0<X<1  of SQRT(C*X**2 + B*X + A)
C
      IF ( C ) 100,200,300
C
C
C
C     Negative C
C
 100  CONTINUE
C
      TERM1 = (B+2*C)*SQRT(A+B+C)-B*(SQRT(A))
      DELTA = B**2 - 4*A*C
      TERM2 = (DELTA/(2*SQRT(-C)))*(ASIN((B+2*C)/(SQRT(DELTA)))-
     &                              ASIN(B/(SQRT(DELTA))))
C
      ALCAL2 = (TERM1 +TERM2) /(4*C)
C
      GO TO 400
C
C
C
C     Zero  C
C
 200  CONTINUE
C
      IF(B.EQ.0) ALCAL2 = SQRT(A)
C
      IF(B.NE.0) ALCAL2 = (2*((A+B)**1.5-A**1.5))/(3*B)
C
      GO TO 400
C
C
C
C     Positive C
C
 300  CONTINUE
C
      TERM1 = (B+2*C)*SQRT(A+B+C)-B*(SQRT(A))
      TERM2 = ((4*A*C-B**2)/(2*SQRT(C)))*ALOG((2*SQRT(C*(A+B+C))+2*C+B)/
     &                                         (2*SQRT(C*A)+B))
C
      ALCAL2 = (TERM1+TERM2) /(4*C)
C
C
 400  CONTINUE
C
      RETURN
      END
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   A V E R A G                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE AVERAG(KER,ID,IC,IN,VARINP,VAROUT)
C
C
C     Parameters
C          I/       VARINP(ID,IC): input array
C          O/       VAROUT(ID)    : output array
C          I/       ID,IC: dimensions of the input array VARINP.
C                       NOTE: Max number of rows  (ID max) = 1000
C          I/       IN    : number of columns of VARINP to be considered
C                          in evaluating the mean value:
C
C                        VAROUT(I) = mean(VARINP(I,J), J=1,IN)
C
C          I/O      KER  :  error code. KER=1 : error in sbr. averag
C
C
      DIMENSION VARINP(ID,IC),VAROUT(ID), SUM(1000)
C
C     Check max number of rows
C
      IF (ID.LE.1000) GO TO 10
      KER = 1
      WRITE (13,1000) ID
      GO TO 400
 10   CONTINUE
C
      DO 100 I = 1,ID
 100  SUM(I) = 0.
C
C
      DO 200 N = 1,IN
      DO 200 I = 1,ID
C
 200  SUM(I) = SUM(I)+VARINP(I,N)
C
C
      DO 300 I = 1,ID
C
 300  VAROUT(I) = SUM(I)/FLOAT(IN)
C
C
 400  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. AVERAG * * *',//,
     &               17X,'MAXIMUM NUMBER OF COMPONENTS EXCEEDED',//,
     &               20X,'          I = ',I4)
C
      RETURN
      END
```

```
C
C
C     ***************************************************************
C     *                                                             *
C     *                                                             *
C     *          S U B R O U T I N E   C C I N C 2                  *
C     *                                                             *
C     *                                                             *
C     ***************************************************************
C
C
      SUBROUTINE CCINC2(X,AL)
C
C
C     Evaluates the arclength AL(2) between three side-nodes of a second
C     order  2D isoparametric element.
C
C     Parameters
C           I/    X(2,3) : cartesian coordinates of the three nodes
C           O/    AL(2)  : arclength
C                               AL(1) = node1-node2
C                               AL(2) = node2-node3
C
      INCLUDE 'post_common'
      DIMENSION X(2,3),AL(2),XX(2,3,3)
C
      DO 100 N = 1,2
      DO 100 I = 1,3
      DO 100 J = 1,3
C
  100 XX(N,I,J) = X(N,I)*X(N,J)
C
C     Evaluating the constants A,B,C
C
      A = 0.
      B = 0.
      C = 0.
C
      DO 200 I = 1,2
C
      A = A + 0.25*XX(I,1,1) + 0.25*XX(I,3,3) - 0.5*XX(I,1,3)
C
      B = B -      XX(I,1,1) +      XX(I,3,3) +   2*XX(I,1,2)
     &      -    2*XX(I,2,3)
C
      C = C +      XX(I,1,1) +    4*XX(I,2,2) +     XX(I,3,3)
     &      -    4*XX(I,1,2) +    2*XX(I,1,3) -   4*XX(I,2,3)
C
  200 CONTINUE
C
C
C     Check the values of A,B,C (MUST BE C*X**2+B*X+A > 0 FOR -1<X<1)
C
      RLARG = 0.
      RSMAL = 0.
C
      DELTA = B**2 - 4*A*C
      IF(DELTA.LE.0.OR.C.LE.0) GO TO 220
      RLARG = (B+SQRT(DELTA))/(2*C)
      RSMAL = (B-SQRT(DELTA))/(2*C)
C
  220 CONTINUE
C
      IF( (A.LT.0)                                           .OR.
     &    ((-C-B).GT.A)                                      .OR.
     &    (C.GT.0.AND.DELTA.GT.0.AND.B.GT.0.AND.RSMAL.LT.1)  .OR.
     &    (C.GT.0.AND.DELTA.GT.0.AND.B.LT.0.AND.RLARG.GT.-1)) GO TO 250
      GO TO 300
C
  250 CONTINUE
      WRITE (NOWC,1000) A,B,C,X
      KERROR = KERROR +1
      GO TO 400
C
  300 CONTINUE
C
      AL(1) = ALCAL1(A,B,C)
      AL(2) = ALCAL2(A,B,C)
C
  400 CONTINUE
C
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CCINC2  * * *',//,
     &             20X,' BAD VALUES FOR CONSTANTS A,B,C',//,
     &             20X,' A = ',E12.5,' B = ',E12.5,' C = ',E12.5,//,
     &             20X,'NODE COORDINATES ARE :',/
     &             20X,'                X1              X2   ',
     &            /,20X,'      NODE1',2(6X,F10.5),/,
     &             20X,'      NODE2',2(6X,F10.5),/,
     &             20X,'      NODE3',2(6X,F10.5),/)
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   C H K S E T                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      SUBROUTINE CHKSET(ISET,NAB,NLC)
C
C     Find the local identity number
C
C     Parameters
C           I/    ISET : Set flag
C                             ISET = 0  : Element set
C                             ISET = 1  : Node input set
C                             ISET = 2  : Node set
C           I/    NAB  : Abaqus number
C           O/    NLC  : Local number: NLC =
C                                          IELC (ISET = 0)
C                                          NNIS (ISET = 1)
C                                          NNLC (ISET = 2)
C                                    NLC = 0 if the node/elt doesn't
C                                            belong to the set
C
      INCLUDE 'post_common'
C
      NLC = 0
C
      GO TO (100,200) ISET
C
C
C     Element set
C
      DO 50 IE = 1,NTELT
            IF (IELTOP(1,IE).NE.NAB)GO TO 50
            NLC = IE
            GO TO 300
 50   CONTINUE
      GO TO 300
C
C
```

```
C
      RETURN
      END
```

```
C     Node input set
C
 100  CONTINUE
C
      DO 150 IN = 1,NTOTIS
            IF(NCONN(1,IN).NE.NAB) GO TO 150
            NLC = IN
            GO TO 300
 150  CONTINUE
      GO TO 300
C
C
C     Node set
C
 200  CONTINUE
C
      DO 250 IN = 1,NTNOD
            IF(NABAQ(IN).NE.NAB) GO TO 250
            NLC = IN
            GO TO 300
 250  CONTINUE
C
 300  CONTINUE
C
      RETURN
      END
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   C K P E R M                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      SUBROUTINE CKPERM(ID,IMOVE,INEW,B,X)
C
C     Check if in the matrix AN rows have been permuted. If yes(IMOVE=1)
C     it permutes the Right Hand Side B according to the order given by
C     vector INEW
C
      DIMENSION INEW(ID),B(ID),X(ID)
C
      IF(IMOVE.NE.1) GO TO 500
C
        DO 100 NR = 1,ID
 100    X(NR) = B(INEW(NR))
        DO 200 NR = 1,ID
 200    B(NR) = X(NR)
C
 500  CONTINUE
C
      RETURN
      END
```

```
C
C
C     **********************************************************************
C     *                                                                    *
C     *                                                                    *
C     *          S U B R O U T I N E   C O N N E C                         *
C     *                                                                    *
C     *                                                                    *
C     **********************************************************************
C
C
      SUBROUTINE CONNEC(IFISNO,IDONE)
C
C
C     Evaluates the connectivity matrices NCONN & IELTOP
C     Evaluates the total number of nodes NTNOD and element NTELT
C
C     Parameters
C         I/      IFISNO(10000) :local input-set-number for abaqus nodes
C         O/      IDONE (10000) :local number for abaqus nodes
C
      INCLUDE 'post_common'
      DOUBLE PRECISION ARRAY
      DIMENSION IFISNO(10000), IDONE(10000)
      DIMENSION ARRAY(513), JRRAY(2,513)
      EQUIVALENCE (ARRAY(1), JRRAY(1,1))
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C
C     Scanning file 8
C
C
      NTNOD = 0
      NTELT =1
C
      DO 500 K =1,99999
C
            CALL DBFILE(0,ARRAY,JRCD)
            IF(JRCD.NE.0) GO TO 600
C
            LR   = JRRAY(1,1)
            KEY  = JRRAY(1,2)
```

```
            IF(KEY.NE.1900) GO TO 500
            IEAB = JRRAY(1,3)
C
C           Check if any of the nodes of IEAB is a node of the input set
C
            IADD = 1
            DO 200 NN = 1,NNELT
                  NNAB = JRRAY(1,NN+4)
                  IF(IFISNO(NNAB).EQ.0) GO TO 200
                  IADD = IADD*0
C
C                 NNAB is a node of the input set -> fill NCONN
C
                  NNIS = IFISNO(NNAB)
                  KP = 4 + NCONN(3,NNIS)*3
                  NCONN(3,NNIS) = NCONN(3,NNIS) + 1
C
C                 check max number of element to which NNIS belongs
C
                  IF(NCONN(3,NNIS).LE.6) GO TO 100
                  KERROR = KERROR +1
                  WRITE(NOWC,1000) NNAB
                  GO TO 700
 100              CONTINUE
C
C                 filling NCONN
C
                  NCONN(KP,NNIS)   = IEAB
                  NCONN(KP+1,NNIS) = NTELT
                  NCONN(KP+2,NNIS) = NN
 200        CONTINUE
C
C           Check if the elt is on the path -> if yes fill IELTOP
C
            IF (IADD.GT.0) GO TO 500
            IELTOP(1,NTELT) = IEAB
C
            DO 400 NN = 1,NNELT
                  NNAB = JRPAY(1,NN+4)
C
C                 Check if the node has been already numbered in the
C                 local list. If not -> put it in the list.
C
                  IF(IDONE(NNAB).NE.0) GO TO 300
                  NTNOD = NTNOD+1
                  IDONE(NNAB) = NTNOD
                  NABAQ(NTNOD) = NNAB
C
                  IF(IFISNO(NNAB).EQ.0) GO TO 300
                  NNIS = IFISNO(NNAB)
                  NCONN(2,NNIS) = NTNOD
C
 300              CONTINUE
C
                  IA = 2+(NN-1)*2
                  IELTOP(IA,NTELT)   = NNAB
                  IELTOP(IA+1,NTELT) = IDONE(NNAB)
 400        CONTINUE
            NTELT = NTELT +1
 500  CONTINUE
 600  CONTINUE
C
      NTELT = NTELT -1
C
C     Check max number of elt's (100) and nodes (800)
C

      IF(NTELT.LE.100.AND.NTNOD.LE.800) GO TO 700
      KERROR = KERROR + 1
      WRITE(NOWC,2000) NTELT,NTNOD
C
 700  CONTINUE
C
      IF (IOUTG.EQ.0) GO TO 900
C
      WRITE(NOWG,3000)
C
      DO 750 NNIS = 1,NTOTIS
      IMAX = NCONN(3,NNIS)*3+3
 750  WRITE(NOWG,4000) NNIS,(NCONN(I,NNIS),I=1,IMAX)
C
      WRITE(NOWG,5000)
C
      DO 800 IELC = 1,NTELT
      IMAX = NNELT*2 + 1
 800  WRITE(NOWG,6000) IELC,(IELTOP(I,IELC),I = 1,IMAX)
C
      WRITE(NOWG,7000)
C
      DO 850 NNLC = 1,NTNOD
```

```
 850  WRITE(NOWG,8000) NNLC,NABAQ(NNLC)
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CONNEC  * * *',//,
     & 10X,'MAX NUMBER OF ELEMENTS CONNECTED TO ONE NODE EXCEEDED',//,
     & 10X,'MORE THAN 6 ELEMENTS ARE CONNECTED TO NODE ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CONNEC  * * *',//,
     & 10X,'MAX NUMBER OF ELEMENTS and/or NODES EXCEEDED',//,
     & 10X,'TOTAL NUMBER OF ELEMENTS = ',I4,'  (MAX : 100)',//,
     & 10X,'TOTAL NUMBER OF NODES = ',I4,' (MAX: 800)')
 3000 FORMAT(1H1,///,40X,'* * * N C O N N * * *',//,
     & 1X,'NNIS NNAB NNLC NBE IAB1 LC1 P1 IAB2 LC2 P2 IAB3 LC3 P3 ',
     &     'IAB4 LC4 P4 IAB5 LC5 P5 IAB6 LC6 P6',/)
 4000 FORMAT(1X,I3,2X,I4,1X,I3,1X,I3,6(1X,I4,1X,I3,1X,I2))
 5000 FORMAT(1H1,///,40X,'* * * I E L T O P * * *',//,
     & 1X,'IELC IEAB NAB1 NLC1 NAB2 NLC2 NAB3 NLC3 NAB4 NLC4 NAB5 ',
     &     'NLC5 NAB6 NLC6 NAB7 NLC7 NAB8 NLC8',/)
 6000 FORMAT(18(1X,I4))
 7000 FORMAT(1H1,///,40X,'* * * N A B A Q * * *',//,
     & 1X,'NNLC NNAB ',/)
 8000 FORMAT(2(1X,I4))
C
      RETURN
      END
```

```
C
C
C     *******************************************************************
C     *                                                                 *
C     *                                                                 *
C     *            S U B R O U T I N E   C T R I N P                    *
C     *                                                                 *
C     *                                                                 *
C     *******************************************************************
C
C
      SUBROUTINE CTRINP
C
C     Reads the contro. variables of the procedure
C
      INCLUDE 'post_common'
C
      READ(NORI,6000) NVAR,JSUB
C
      DO 100 NV = 1,NVAR
C
             READ(NORI,7000) IVAR,IV,K
C
             IVDIM(IVAR) = IV
             KAB  (IVAR) = K
C
             READ(NORI,8000) (KPKP(J,IVAR),J=1,8)
C
             DO 10 J =1,8
                   IF(KPKP(J,IVAR).NE.0) GO TO 10
                   NPROC(IVAR) = J-1
                   GO TO 20
 10          CONTINUE
 20          CONTINUE
C
C            Check the consistency of the procedure
C
             IF(KPKP(1,IVAR).EQ.1.OR.KPKP(1,IVAR).EQ.2) GO TO 30
                KERROR = KERROR +1
                WRITE(NOWC,1000) IVAR,KPKP(1,IVAR)
 30          CONTINUE
C
             IGRST(IVAR) = 0
             IGRAD = 0
             DO 60 J = 2,NPROC(IVAR)
                   IF(KPKP(J,IVAR).NE.1.AND.KPKP(J,IVAR).NE.2) GO TO 40
                      KERROR = KERROR +1
                      WRITE(NOWC,2000) J,IVAR,KPKP(J,IVAR)
 40                CONTINUE
C
             IF(KPKP(J,IVAR).NE.3) GO TO 60
                   IF(IGRAD.EQ.0) GO TO 50
                      KERROR = KERROR +1
                      WRITE(NOWC,3000) IVAR
 50                CONTINUE
                   IGRAD = 1
                   IGRST(IVAR) = J
 60          CONTINUE
      IF(IGRST(IVAR).LE.1)  GO TO 100
        DO 80 J = 1,IGRST(IVAR)-1
              IF(KPKP(J,IVAR).NE.4) GO TO 70
                KERROR = KERROR + 1
                WRITE (NOWC,4000) IVAR
 70           CONTINUE
              IF(KPKP(J,IVAR).NE.5) GO TO 80
                KERROR = KERROR + 1
                WRITE (NOWC,5000) IVAR
 80            CONTINUE
 100  CONTINUE
C
C
C
      READ(NORI,9000) NTOUT
C
      DO 200  NOUT = 1,NTOUT
 200         READ(NORI,6000)NKSTEP(NOUT),NKINCR(NOUT)
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     & 10X,'THE FIRST STEP OF THE PROCEDURE MUST BE A READING STEP',//,
     & 10X,'BAD FIRST STEP FOR VARIABLE N. ',I4,'FIST STEP = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     & 10X,'ONLY THE FIRST STEP OF THE PROC. CAN BE A READING STEP',//,
     & 10X,'BAD STEP N.',I4,' FOR VARIABLE N. ',I4,' STEP = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     & 10X,'ONLY ONE GRADIENT STEP IS ALLOWED FOR EACH VARIABLE',//,
     & 10X,'MORE THAN ONE GRADIENT STEP FOR VARIABLE N. ',I4)
 4000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     & 10X,'ROTATION STEP IS NOT ALLOWED FOR ALL NODES',//,
     & 10X,'A ROT. STEP PRECED. GRADIENT STEP FOR VARIABLE N. ',I4)
 5000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     &  10X,'NORM.DOT PROD. STEP IS NOT ALLOWED FOR ALL NODES',//,
     &  10X,'A  N.D.P. STEP PRECED. GRADIENT STEP FOR VARIABLE N. ',I4)
 6000 FORMAT(2(I5))
 7000 FORMAT(3(I5))
 8000 FORMAT(8(I5))
 9000 FORMAT(I5)
C
      RETURN
      END
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           F U N C T I O N     D L I S C O                     *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
C
      FUNCTION DLISCO(K,J,I)
C
C     Evaluates the dimensionless isoparametric coordinates of the nodes
C
C     Parameters
C                K : node number
C                J : required coordinate
C                                           1 : G
C                                           2 : H
C                                           3 : R
C                I : Element type
C                                   8 : 2D - 8 nodes  isop. element
C
      INCLUDE 'post_common'
C
      DLISCO = 0.
C
      IF ( I.EQ.8) GO TO 100
         KERROR = KERROR + 1
         WRITE(NOWC,1000) I
         GO TO 400
  100 CONTINUE
C
      GO TO (200,300) J
         KERROR = KERROR + 1
         WRITE(NOWC,2000) J
         GO TO 400
C
C     First coordinate : G
C
  200 CONTINUE
C
      GO TO (210,220,230,240,250,260,270,280) K
```

```
C
          KERROR = KERROR + 1
          WRITE(NOWC,3000) K
          GO TO 400
C
 210  CONTINUE
      DLISCO = -1.
      GO TO 400
C
 220  CONTINUE
      DLISCO =  1.
      GO TO 400
C
 230  CONTINUE
      DLISCO =  1.
      GO TO 400
C
 240  CONTINUE
      DLISCO = -1.
      GO TO 400
C
 250  CONTINUE
      DLISCO =  0.
      GO TO 400
C
 260  CONTINUE
      DLISCO =  1.
      GO TO 4C0
C
 270  CONTINUE
      DLISCO =  0.
      GO TO 400
C
 280  CONTINUE
      DLISCO = -1.
      GO TO 400
C
C     Second coordinate : H
C
 300  CONTINUE
C
      GO TO (310,320,330,340,350,360,370,380) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,3000) K
```

```
          GO TO 400
C
 310  CONTINUE
      DLISCO = -1.
      GU TO 400
C
 320  CONTINUE
      DLISCO = -1.
      GO TO 400
C
 330  CONTINUE
      DLISCO =  1.
      GO TO 400
C
 340  CONTINUE
      DLISCO =  1.
      GO TO 400
C
 350  CONTINUE
      DLISCO = -1.
      GO TO 400
C
 360  CONTINUE
      DLISCO =  0.
      GO TO 400
C
 370  CONTINUE
      DLISCO =  1.
      GO TO 400
C
 380  CONTINUE
      DLISCO =  0.
      GO TO 400
C
 400  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
     &  10X,'THE REQUIRED-COORDINATE-CODE MUST BE 1 (G) OR 2(H)',//,
     &  10X,'REQUIRED COORDINATE CODE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
     &  10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
     &  10X,'NODE NUMBER = ',I4)
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   D O T P R O                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE DOTPRO(ND,ID1,ID2,JD1,JD2,JDPR,AR1,AR2,ARPR)
C
C     Evaluate the dot-product of vectors and tensors
C
C     Parameters
C          I/    ND  : dimensions of space
C          I/    ID1 : dimension-flag for ARRAY1
C                                            1: vector -> ARRAY1(ND)
C                                            2: tensor -> ARRAY1(ND,ND)
C          I/    ID2 : dimension-flag for ARRAY2
C                                            1: vector -> ARRAY2(ND)
C                                            2: tensor -> ARRAY2(ND,ND)
C          I/O   VECT: input/output vector.
C                       Contains:
C                                 ARRAY1,ARRAY2,ARRAY1*ARRAY2
C                      The components are in the order:
C                      Vector -> a(1),a(2),....a(ND)
C                      Tensor -> A(1,1),A(2,1),..A(ND,1),A(1,2)..A(ND,ND)
C
      DIMENSION VECT(1000),AR1(JD1),AR2(JD2),ARPR(JDPR)
C
C
      IVD = JD1+JD2+JDPR
      DO 5  J=1,IVD
 5    VECT(J) = 0.
      DO 10 J=1,JD1
 10   VECT(J) = AR1(J)
      DO 20 J=1,JD2
      JV = JD1+J
 20   VECT(JV) = AR2(J)
      N1  = 1
      N2  = 1
      NPR = 1
C
      IF(ID1.EQ.2) N1 = ND
C
      RETURN
      END
      IF(ID2.EQ.2) N2 = ND
      IF(ID1.EQ.2.AND.ID2.EQ.2) NPR = ND
C
      NP1 = ND*N1
      NP2 = ND*N1 + ND*N2
C
      DO 200 NR = 1,N1
      DO 200 NC = 1,N2
C
            SUM = 0.
            DO 100 N = 1,ND
                  IF = NR+(N-1)*N1
                  IS = NP1+N+(NC-1)*ND
 100        SUM =SUM+VECT(IF)*VECT(IS)
C
            IPR = NP2 +NR+(NC-1)*NPR
 200  VECT(IPR) =SUM
C
      DO 300 J = 1,JDPR
      JV = JD1+JD2+J
 300    ARPR(J) = VECT(JV)
      RETURN
      END
```

```
C
C
C
C     ***************************************************************
C     *                                                             *
C     *                                                             *
C     *          S U B R O U T I N E   D S V R I N                  *
C     *                                                             *
C     *                                                             *
C     ***************************************************************
C
      SUBROUTINE DSVRIN(IV,ID,IA,VAR,ALL)
C
C     Read variable data from  file 'postdat'
C
C     Parameters
C           I/   IV  : Variable ID number
C           I/   ID  : Dimension flag of the input field
C                                                     0 : scalar field
C                                                     1 : vector field
C                                                     2 : tensor field
C           I/   IA  : All-flag:
C                          IA = 0 -> values required only at path-nodes
C                          IA = 1 -> values required at all nodes
C           O/   VAR : Variable values at path nodes
C           O/   ALL : Variable values at all  nodes
C
      INCLUDE 'post_common'
C
      DIMENSION VAR(9,200),ALL(9,800),VECT(9)
      DIMENSION JV(200),JA(800)
C
      DO 10 K=1,200
 10   JV(K) = 0
C
      DO 20 K = 1,800
 20   JA(K) = 0
C
      DO 40 I = 1,9
      DO  30 N = 1,NTOTIS
 30   VAR(I,N) = 0.
      DO 40 N = 1,NTNOD
 40   ALL(I,N) = 0.
C
      REWIND(UNIT=NORD)
```

```
C
      NCOMP = 1
      IF(ID.EQ.1) NCOMP = 3
      IF(ID.EQ.2) NCOMP = 9
C
C     Scanning data file
C
      DO 400 K = 1,99999
             READ(NORD,3000,END = 500) I,NAB,(VECT(I),I=1,NCOMP)
      IF(I.NE.IV) GO TO 400
C
C           Check if the node belongs to the input set:if YES->Fill VAR
C
             CALL CHKSET(1,NAB,NNIS)
             IF(NNIS.EQ.0) GO TO 200
               JV(NNIS) = 1
               DO 100 I = 1,NCOMP
 100           VAR(I,NNIS) = VECT(I)
 200         CONTINUE
C
C            Check if CURALL is required and if NNAB belongs to the node
C            set: If YES -> Fill ALL
C
      IF(IA.EQ.0) GO TO 400
              CALL CHKSET(2,NAB,NNLC)
      IF(NNLC.EQ.0) GO TO 400
                JA(NNLC) = 1
                DO 300 I = 1,NCOMP
 300            ALL(I,NNLC) = VECT(I)
 400  CONTINUE
C
 500  CONTINUE
C
C
C     Check if all nodes have been found
C
      DO 600 K =1,NTOTiS
             IF(JV(K).EQ.1) GO TO 600
             KERROR = KERROR +1
             WRITE(NOWC,1000) K,IV
 600  CONTINUE
C
         IF(IA.LE.0) GO TO 800
      DO 700 K = 1,NTNOD
             IF(JA(K).EQ.1) GO TO 700
                     KERROR = KERROR + 1
                     WRITE( NOWC,2000 ) K,IV
 700  CONTINUE
C
 800  CONTINUE
C

 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. DSVRIN  * * *',//,
     &  10X,'NODE NUMBER',I4,'OF THE INPUT SET HAS NOT BEEN FOUND',
     &  ' FOR IVAR = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. DSVRIN  * * *',//,
     &  10X,'NODE NUMBER',I4,'OF THE NODE SET HAS NOT BEEN FOUND',
     &  ' FOR IVAR = ',I4)
 3000 FORMAT(2(I5),9(F10.0))
C
      RETURN
      END
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   F I L L I N                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE FILLIN(KEY,LR,ARRAY,NLC,CUR)
C
C     Fills CUR(j,NLC) with the values given by ARRAY according to rules
C     defined by KEY
C
C     Parameters
C         I/      KEY  : Abaqus  file-8-reading key
C         I/      LR   : Record length of ARRAY
C         I/      ARRAY: Input record read from file 8
C         I/      NLC  : local identification number : number of the
C                        column of CUR to be filled with ARRAY
C         I/O     CUR  : internal variable input array
C
      INCLUDE 'post_common'
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),CUR(9,800)
C
      DO 10 I = 1,9
 10   CUR(I,NLC) = 0.
C
      IF(KEY.GE.100) GO TO 400
C
C     Element variable
C
      IF(KEY.EQ.2.OR.KEY.EQ.14) GO TO 100
      IF(KEY.EQ.11) GO TO 200
      IF(KEY.GE.21.AND .KEY.LE.25) GO TO 300
C
      KERROR = KERROR + 1
      WRITE (NOWC,1000) KEY
      GO TO 700
C
C     1 component-variable
C
 100  CONTINUE
C
      CUR(1,NLC) = ARRAY(3)
      GO TO 700
C
C     Stress tensor
C
 200  CONTINUE
C
      CUR(1,NLC) = ARRAY(3)
      CUR(2,NLC) = ARRAY(4)
      CUR(3,NLC) = ARRAY(5)
      CUR(4,NLC) = ARRAY(6)
      CUR(7,NLC) = ARRAY(6)
      IF(NDIM.LT.3) GO TO 700
      CUR(5,NLC) = ARRAY(7)
      CUR(8,NLC) = ARRAY(7)
      CUR(6,NLC) = ARRAY(8)
      CUR(9,NLC) = ARRAY(8)
      GO TO 700
C
C     Strain tensor
C
 300  CONTINUE
C
      CUR(1,NLC) = ARRAY(3)
      CUR(2,NLC) = ARRAY(4)
      CUR(3,NLC) = ARRAY(5)
      CUR(4,NLC) = ARRAY(6)*0.5
      CUR(7,NLC) = ARRAY(6)*0.5
      IF(NDIM.LT.3) GO TO 700
      CUR(5,NLC) = ARRAY(7)*0.5
      CUR(8,NLC) = ARRAY(7)*0.5
      CUR(6,NLC) = ARRAY(8)*0.5
      CUR(9,NLC) = ARRAY(8)*0.5
      GO TO 700
C
C     Node variable
C
 400  CONTINUE
C
      DO 500 J = 4,LR
            IC = J-3
            IF(IC.LT.9) GO TO 500
               KERROR = KERROR + 1
               WRITE(NOWC,2000)
```

```
                GO TO 700
 500  CUR(IC,NLC) = ARRAY(J)
C
 700  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FILLIN  * * *',//,
     &  10X,'ONLY KEY = 2,11,14,21,22,23,24,25 OR KEY GT. 100',
     &  ' ARE IMPLEMENTED OPTIONS',//,10X,'KEY = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FILLIN  * * *',//,
     &  10X,'FOR NODE VARIABLES ONLY 9 COMPONENTS CAN BE READ ')
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   F 8 V R I N                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE F8VRIN(NOUT,KREQ,IA,CURVAR,CURALL)
C
C     Read variable data from Abaqus-file 8
C
C     Parameters
C          I/  KREQ  : Abaqus file 8-read-key for the required variable
C          I/  IA    : All-flag:
C                          IA = 0 -> values required only at path-nodes
C                          IA = 1 -> values required at all nodes
C          O/   CURVAR: Variable values at path nodes
C          O/   CURALL: Variable values at all  nodes
C
      INCLUDE 'post_common'
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),JRRAY(2,513),CURVAR(9,200),CURALL(9,800)
      DIMENSION JV(200),JA(800)
      EQUIVALENCE (ARRAY(1),JRRAY(1,1))
C
      DO 10 K=1,200
 10   JV(K) = 0
C
      DO 20 K = 1,800
 20   JA(K) = 0
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C     Scanning file 8
C
      DO 330 K = 1,999999
             CALL DBFILE(0,ARRAY,JRCD)
         IF(JRCD.NE.0) GO TO 350
```

```
            LR   = JRRAY(1,1)
            KEY  = JRRAY(1,2)
C
C           Finding the right step/increment
C
            IF (KEY.NE.2000) GO TO 330
            NST = JRRAY(1,8)
            NIN = JRRAY(1,9)
            IF(NKSTEP(NOUT).NE.NST.OR.NKINCR(NOUT).NE.NIN) GO TO 330
C
      JF = 1
C
      DO 300 KK = 1,999999
            CALL DBFILE(0,ARRAY,JRCD)
         IF(JRCD.NE.0) GO TO 350
            LR   = JRRAY(1,1)
            KEY  = JRRAY(1,2)
         IF(KEY.EQ.2001) GO TO 350
            NNMB = JRRAY(1,3)
C
C           Check if a node or an element variable is required
C
            IF(KREQ.GE.100) GO TO 100
C
C              Element variable is required
C
               ILOC = JRRAY(1,6)
         IF(KEY.NE.1.OR.ILOC.NE.4) GO TO 300
C
C              The subsequent record of file 8 contains nodal averaged
C              values at node NNMB
C
               CALL DBFILE(0,ARRAY,JRCD)
               LR = JRRAY(1,1)
               KEY = JRRAY(1,2)
C
C           start from here if nodal variable is required
C
 100        CONTINUE
C
         IF(KEY.NE.KREQ) GO TO 300
C
C           Check if the node belongs to the input set: if YES ->
C                                                          Fill CURVAR
            CALL CHKSET(1,NNMB,NNIS)
```

```
            IF(NNIS.EQ.0) GO TO 200
               JV(NNIS) = 1
               CALL FILLIN(KREQ,LR,ARRAY,NNIS,CURVAR)
 200        CONTINUE
C
C           Check if CURALL is required and if NNAB belongs to the node
C           set: If YES -> Fill CURALL
C
         IF(IA.LE.0) GO TO 300
            CALL CHKSET(2,NNMB,NNLC)
         IF(NNLC.EQ.0) GO TO 300
            JA(NNLC) = 1
            CALL FILLIN(KREQ,LR,ARRAY,NNLC,CURALL)
 300  CONTINUE
 330  CONTINUE
 350  CONTINUE
C
C     Check if the required step/incr has been found
C
      IF(JF.EQ.1) GO TO 390
            KERROR = KERROR +1
            WRITE(NOWC,3000) NKINCR(NOUT),NKSTEP(NOUT)
 390  CONTINUE
C
C
C     Check if all nodes have been found
C
      DO 400 K =1,NTOTIS
            IF(JV(K).EQ.1) GO TO 400
            KERROR = KERROR +1
            WRITE(NOWC,1000) K,KREQ
 400  CONTINUE
C
         IF(IA.LE.0) GO TO 600
      DO 500 K = 1,NTNOD
            IF(JA(K).EQ.1) GO TO 500
            KERROR = KERROR + 1
            WRITE( NOWC,2000 ) K,KREQ
 500  CONTINUE
C
 600  CONTINUE
C
1000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8VRIN  * * *',//,
     &  10X,'NODE NUMBER',I4,'OF THE INPUT SET HAS NOT BEEN FOUND',
     &  ' FOR KEY = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8VRIN  * * *',//,
     &  10X,'NODE NUMBER',I4,'OF THE NODE SET HAS NOT BEEN FOUND',
     &  ' FOR KEY = ',I4)
C
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8VRIN  * * *',//,
     &  10X,'INCREMENT ',I4,' OF STEP',I4,'  HAS NOT BEEN FOUND')
C
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           S U B R O U T I N E   G E O I N P                   *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE GEOINP
C
C     Manages the geometric input
C
      INCLUDE 'post_common'
C
      DIMENSION IFISNO(10000),IDONE(10000)
C
      READ(NORI,1000) ITYPE,KPSTOP,IOUTG
C
C     Path definition
C
      CALL PATDEF(IFISNO)
C
      IF (KERROR.GT.0) GO TO 100

C
C
C     Connectivity matrices
C
      CALL CONNEC(IFISNO,IDONE)
C
      IF (KERROR.GT.0) GO TO 100

C
C
C     Node cartesian components
C
      CALL NCARCO(IDONE)
C
C
 100  CONTINUE
C
 1000 FORMAT(3(I5))
```

```
      RETURN
      END
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   G E O P A T                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      SUBROUTINE GEOPAT
C
C     Evaluates curvilinear coordinate , normal and tangent vector to
C     the path.
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ==================================================================
C
      INCLUDE 'post_common'
C
      DIMENSION N(3), X(2,3), T(2,3), AL(2), TOLD(2)
C
      TOLD(1) = 0.
      TOLD(2) = 0.
      SCURV(1) = 0.
      NELT = (NTOTIS-1)/2
C
C     Following the path
C
      DO 500 NE = 1,NELT
             DO 100 I =1,3
                    NNIS = (NE-1)*2 + I
                    N(I) = NCONN(2,NNIS)
             DO 100 J =1,2
 100         X(J,I) = COORDS(J,N(I))
C
C            Evaluating the cart. comp. of the tang. at the three nodes
C
             CALL TANG2D(X,T)
C
C            Average value of the tangent vector at node 1
C
             IF(NE.EQ.1) GO TO 200
               T(1,1) = (T(1,1)+TOLD(1))/2.
```

```
            T(2,1) = (T(2,1)+TOLD(2))/2.
 200       CONTINUE
C
           TOLD(1) = T(1,3)
           TOLD(2) = T(2,3)
C
C          Cartesian component of the normal at nodes 1,2,3
C
           DO 300 I =1,3
                 NNIS = (NE-1)*2+I
                 PATANG(1,NNIS) = T(1,I)
                 PATANG(2,NNIS) = T(2,I)
                 PATANG(3,NNIS) = 0.
                 PATNOR(1,NNIS) = -T(2,I)
                 PATNOR(2,NNIS) =  T(1,I)
                 PATNOR(3,NNIS) =  0.
 300       CONTINUE
C
C          Evaluating the arclength (1-2) and (2-3) : AL(2)
C
           CALL CCINC2(X,AL)
C
           IF (KERROR.GT.0) GO TO 600
C
C          Curvilinear coordinate
C
           DO 400 I =2,3
                 NNIS = (NE-1)*2 +I
                 SCURV(NNIS) = SCURV(NNIS-1) +AL(I-1)
 400       CONTINUE
 500  CONTINUE
 600  CONTINUE
C
      IF (IOUTG.EQ.0) GO TO 800
C
      WRITE(NOWG,1000)
C
      DO 700 NNIS = 1,NTOTIS
 700  WRITE(NOWG,2000) NNIS,SCURV(NNIS),(PATANG(I,NNIS),I = 1,3),
     &                (PATNOR(I,NNIS),I = 1,3)
C
 800  CONTINUE
C
 1000 FORMAT(1H1,////,40X,'* * * P A T H   G E O M E T R Y * * *',//
     &1X,'NNIS  CURV. COOR   T(1)   T(2)   T(3)   N(1)   N(2)   N(3)'//)
```

```
 2000 FORMAT(1X,I4,1X,F10.5,6(1X,F6.3))
C
      RETURN
      END
```

```
C
C
C     ***************************************************************
C     *                                                             *
C     *                                                             *
C     *          S U B R O U T I N E   G R A D I E                  *
C     *                                                             *
C     *                                                             *
C     ***************************************************************
C
      SUBROUTINE GRADIE(IDIM,F,OU)
C
C     Evaluates the gradient of a scalar field or of a vector field   at
C     the nodes of the path.
C
C     Parameters
C           I/    IDIM : Dimension flag of the input field
C                                                 0 : scalar field
C                                                 1 : vector field
C           I/    F    : Input field at all nodes
C           O/    OU   : Output gradient field at the nodes of the path
C
      INCLUDE 'post_common'
C
      DIMENSION F(9,800), OU(9,200), AM(3,3),AU(3,3),AL(3,3),B(3),X(3)
      DIMENSION GR(3,3,6), GRM(3,3), INEW(3)
C
C     Local array :
C            AM : coeff. matrix -> AM(I,J) = [d N(k)/d c(I)] * X(k)(J)
C
C            AL,AU : Triang. fact. matr. AM = AL*AU
C
C            B : RHS -> Bj(I) = [d N(k)/d c(I)] * Fj(k)
C
C            X : Unknown vect -> Xj(I) = d Fj/d x(I)
C
C            GR : Gr. at the node  ev. for each elt ->
C                                      GR(J,I,L) = d Fj/ d x(I)
C                                                    for elt L
C
C            GRM : mean gr. value at the node -> GRM(J,I) = d Fj/ d x(I)
C
C                           N(k)    : Kth shape function
C                           c(I)    : Ith isop. coord.
C                           X(k)(J): Jth cart coord of the Kth node
C                           x(I)    : Ith cartesian coordinate
C                           Fj(k)   :Jth comp. of input field at node k
C
C
C     NOTE : max numb. of element at a node : 6 [GR(3,3,6)]
C     ================================================
C
      DO 10 N = 1,200
      DO 10 I = 1,9
   10 OU(I,N) = 0.
C
      DO 20 I = 1,3
      DO 20 J = 1,3
      GRM(I,J) = 0.
      DO 20 N = 1,6
   20 GR(I,J,N) = 0.
C
      NCOMP = NDIM
      IF(IDIM.EQ.0) NCOMP = 1
C
C     Following the path
C
      DO 400 NP = 1,NTOTIS
            DO 100 NE = 1,NCONN(3,NP)
                  NIELC = 5+(NE-1)*3
                  NKP   = 6+(NE-1)*3
                  IELC  = NCONN(NIELC,NP)
                  KP    = NCONN(NKP  ,NP)
C
C                 Evaluate the coefficient matrix AM
C
                  CALL GRAMAT(IELC,KP,AM)
C
C                 LU factorization
C
                  KER = KERROR
                  CALL LUFACT(KER,3,IMOVE,INEW,AM,AU,AL)
                  KERROR = KER
                  IF(KERROR.GT.0) GO TO 500
C
C           Solving the system for all the components of F
C
            DO 100 ID = 1,NCOMP
C
C                 Evaluating the RHS
```

```
C
                 CALL GRARHS (IELC,NP,F,ID,B)
C
C                solving the system
C
                 CALL SOLVER (3,IMOVE,INEW,AU,AL,B,X)
C
C           Storing the gradient
C
            DO 100 J = 1,NDIM
C
 100        GR(ID,J,NE) = X(J)
C
C           Average value of the gr. over all the elements at the  node
C
            KER = KERROR
            CALL AVERAG(KER,9,6,NCONN(3,NP),GR,GRM)
                 KERROR = KER
                 IF(KERROR.GT.0) GO TO 500
C
C           Storing the result in the output array
C
          IF (IDIM.EQ.1) GO TO 300
C
C           F is a scalar field -> OU is a vector with NDIM components
C
            OU(1,NP) = GRM(1,1)
            OU(2,NP) = GRM(1,2)
            OU(3,NP) = GRM(1,3)
C
      GO TO 400
C
C           F is a vector field -> OU is a tensor with NDIM*NDIM comp's
C
 300      CONTINUE
C
            OU(1,NP) = GRM(1,1)
            OU(2,NP) = GRM(2,2)
            OU(3,NP) = GRM(3,3)
            OU(4,NP) = GRM(1,2)
            OU(5,NP) = GRM(1,3)
            OU(6,NP) = GRM(2,3)
            OU(7,NP) = GRM(2,1)
            OU(8,NP) = GRM(3,1)
            OU(9,NP) = GRM(3,2)
C
 400  CONTINUE
 500  CONTINUE
C
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   G R A M A T                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE GRAMAT(IELC,KP,AM)
C
C     Evaluate the coefficient matrix AM for node KP of element IELC
C
C     Paramweters
C          I/      IELC : local elt id number
C          I/      KP   : position of the node in the elt
C          O/      AM   : coeff. matrix ->
C                                    AM(I,J) = [d N(k)/d c(I)] * X(k)(J)
C
C                       N(k)   : Kth shape function
C                       c(I)   : Ith isop. coord.
C                       X(k)(J): Jth cart coord of the Kth node
C
      INCLUDE 'post_common'
      DIMENSION AM(3,3)
C
      DO 20 NR = 1,3
      DO 10 NC = 1,3
 10   AM(NR,NC) = 0.
 20   AM(NR,NR) = 1.
C
      DO 200 NR = 1,NDIM
      DO 200 NC = 1,NDIM
            SUM = 0.
            DO 100 K = 1, NNELT
                  INLC = 3 +(K-1)*2
                  NNLC = IELTOP(INLC,IELC)
 100        SUM = SUM+ SFDER(KP,K,NR) * COORDS(NC,NNLC)
 200  AM(NR,NC) = SUM
C
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   G R A R H S                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
C
      SUBROUTINE GRARHS(IELC,KP,F,ID,B)
C
C     Evaluates the Right Hand Side of the system to be solved in order
C     to find the gradient of the IDth component of the field F
C
C     Parameters
C          I/      IELC : local elt id number
C          I/      KP   : position of the node in the elt
C          I/      F    : input field at each node
C          I/      ID   : component of F to be considered
C          O/      B    : RHS -> B(I) = [d N(k)/d c(I)] * F(ID)(k)
C
C                           N(k)   : Kth shape function
C                           c(I)   : Ith isop. coord.
C                           F(ID)(k):IDth comp. of input field at node k
C
      INCLUDE 'post_common'
      DIMENSION F(9,800),B(3)
C
      DO 10 I = 1,3
 10   B(I) = 0.
C
      DO 200 J = 1,NDIM
            SUM = 0.
            DO 100 K = 1,NNELT
                  INLC = 3+(K-1)*2
                  NNLC = IELTOP(INLC,IELC)
 100        SUM = SUM + SFDER(KP,K,J) * F(ID,NNLC)
 200  B(J) = SUM
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *           S U B R O U T I N E   L U F A C T                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE LUFACT(KER,ID,IMOVE,INEW,AM,AU,AL)
C
C     Factorize a square matrix AM(ID,ID) into a lower triang. matrix
C     AL(ID,ID) and an upper triang. matrix AU(ID,ID) . AM = AL*AU
C     If elements on the main diagonal of AM are equal to zero,
C     rows of AM are permuted and a track is kept in array INEW.
C     Then (AM)perm. is factorized.
C     Parameters
C           I/O    KER : error count.
C           I/     ID : matrix dimension
C           O/     IMOVE : row permutation code (=1 : row permuted )
C           O/     INEW : perm. record :INEW(J)= old # of new row #J
C           I/     AM : square input matrix
C           O/     AL : lower triang matrix
C           O/     AU : upper triang.matrix
C
      DIMENSION AM(ID,ID),AU(ID,ID),AL(ID,ID),INEW(ID)
C
C     Check if matrix AM has zero elements on the main diagonal
C     and permutation of rows if it is needed.
C
      CALL RWPERM (ID,IMOVE,INEW,AM,AU,AL)
C
C     Check if AM has any zero pivot
C
      IF(IMOVE.GE.0) GO TO 5
        KER = KER+1
        GO TO 200
 5    CONTINUE
C
      PIVMIN = 1.E-6
C
      DO 20 NC = 1,ID
      DO 10 NR = 1,ID
```

```
       AU(NR,NC) = AM(NR,NC)
 10    AL(NR,NC) = 0.
 20    AL(NC,NC) = 1.
C
      DO 100 NFR = 1,ID-1
            PIVOT = AU(NFR,NFR)
            IF(ABS(PIVOT).GT.PIVMIN) GO TO 50
              KER = KER +1
              WRITE (13,1000) NFR,PIVOT
              WRITE (13,*) AM
              IF(IMOVE.GT.0) WRITE(13,2000)
              IF(IMOVE.GT.0) WRITE(13,*) INEW
              GO TO 200
 50         CONTINUE
       DO 100 NSR = NFR+1,ID
             AL(NSR,NFR) = AU(NSR,NFR)/PIVOT
       DO 100 NC = NFR.ID
 100  AU(NSR,NC) = AU(NSR,NC)-AU(NFR,NC)*AL(NSR,NFR)
C
 200  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. LUFACT  * * *',//,
     &  10X,'ROW ',I4,' PIVOT TOO SMALL!!!! PIVOT = ',E12.5,//,
     &  10X,'INPUT MATRIX :',//)
 2000 FORMAT(///,20X,'* * * N O T E ! ! ! * * *',//,
     &  10X,'ROWS HAVE BEEN PERMUTED . PERMUTATION RECORD VECTOR ',
     &  10X,'INEW FOLLOWS: (INEW(J) = OLD # OF ACTUAL ROW # J)',/)
C
      RETURN
      END
```

```
C
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   N C A R C O                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE NCARCO(IDONE)
C
C     Reads the cartesian coordinates of the nodes
C
C     Parameters
C         I/      IDONE (10000) :local number for abaqus nodes
C
      INCLUDE 'post_common'
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),JRRAY(2,513),IDONE (10000)
      EQUIVALENCE (ARRAY(1), JRRAY(1,1))
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C     Scanning file 8
C
      NCHECK = 0
      DO 200 K = 1,99999
            CALL DBFILE(0,ARRAY,JRCD)
            IF (JRCD.NE.0) GO TO 300
            LR = JRRAY(1,1)
            KEY = JRRAY(1,2)
            IF(KEY.NE.1901) GO TO 200
            NNAB = JRRAY(1,3)
            NNLC = IDONE(NNAB)
            IF(NNLC.EQ.0) GO TO 200
            NCHECK = NCHECK + 1
            DO 100 ND = 1, NDIM
 100        COORDS(ND,NNLC) = ARRAY(3+ND)
 200  CONTINUE
```

```
 300  CONTINUE
C
C     Check if all the nodes have been found
C
      IF(NCHECK.EQ.NTNOD) GO TO 400
        KERROR = KERROR +1
        WRITE(NOWC,1000) NCHECK,NTNOD
 400  CONTINUE
C
      IF(IOUTG.EQ.0) GO TO 600
C
      WRITE(NOWG,2000)
C
      DO 500 NNLC = 1,NTNOD
 500  WRITE(NOWG,3000) NNLC,NABAQ(NNLC),(COORDS(I,NNLC),I =1,3)
C
 600  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. NCARCO  * * *',//,
     &  10X,'TOTAL NUMBER OF READ NODES :',I4,'IS NOT CONSISTENT WITH ',
     &  'THE TOTAL NUMBER OF NODES : ',I4)
 2000 FORMAT(1H1,////,40X,'* * *   C C O R D I N A T E S * * *',//
     &  'NNLC NNAB    X1         X2         X3',/)
 3000 FORMAT (2(1X,I4),3(2X,F6.3,2X))
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   N O R M P R                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE NORMPR(ID,OLD,CUR)
C
C     Find the dot product of the input field OLD with the vector normal
C     to the path. CUR = OLD * n
C     If OLD is a vector field (ID = 1)->CUR : normal component
C     If OLD is a tensor field (ID = 2)->CUR : surface (traction) vector
C
C     Parameters
C          I/    ID : dimension-flag for OLD ->  1: vector ,  2: tensor
C          I/    OLD : input field at the path nodes
C          O/    CUR : output normal component/vector
C
      INCLUDE 'post_common'
      DIMENSION OLD(9,200),CUR(9,200),V(3),T(3,3),VP(3),AN(3)
C
      DO 600 NP = 1,NTOTIS
            DO 100 NR = 1,9
 100        CUR(NR,NP) = 0.
C
            DO 200 I = 1,3
            VP(I) = 0.
 200        AN(I) = PATNOR (I,NP)
C
            IF(ID.EQ.2) GO TO 400
C
C           OLD is a vector field
C
            DO 300 I = 1,3
 300        V(I) = OLD(I,NP)
C
            CALL DOTPRO(3,1,1,3,3,1,V,AN,PR)
            CUR(1,NP) = PR
      GO TO 600
C
```

```
C           OLD is a tensor field
C
 400        CONTINUE
C
            T(1,1) = OLD(1,NP)
            T(2,2) = OLD(2,NP)
            T(3,3) = OLD(3,NP)
            T(1,2) = OLD(4,NP)
            T(1,3) = OLD(5,NP)
            T(2,3) = OLD(6,NP)
            T(2,1) = OLD(7,NP)
            T(3,1) = OLD(8,NP)
            T(3,2) = OLD(9,NP)
C
            CALL DOTPRO(3,2,1,9,3,3,T,AN,VP)
C
            DO 500 I = 1,3
 500        CUR(I,NP) = VP(I)
 600  CONTINUE
C
      RETURN
      END
```

```
C
C
C     *******************************************************************
C     *                                                                 *
C     *                                                                 *
C     *           S U B R O U T I N E   P A T D E F                     *
C     *                                                                 *
C     *                                                                 *
C     *******************************************************************
C
      SUBROUTINE PATDEF(IFISNO)
C
C     Reads input set nodes. Find total number of input set nodes
C
C     Parameters
C         O/      IFISNO(10000) :local input-set-number for abaqus nodes
C
      INCLUDE 'post_common'
      DIMENSION IFISNO(10000),NINP(15)
C
      NNIS = 0
      READ(NORI,1000) NSET
      DO 100 NS = 1,NSET
            READ(NORI,2000) IN,(NINP(K),K=1,15)
            IF(IN.NE.0)THEN
              NF = NINP(1)
              NL = NINP(2)
              NT = 1+(NL-NF)/IN
              DO 130 NN = 1,NT
                    NNIS = NNIS+1
                    NNAB = NF + (NN-1)*IN
                    NCONN(1,NNIS) = NNAB
                    IFISNO(NNAB) = NNIS
 130          CONTINUE
            ELSE
              DO 160 NN=1,15
                    IF(NINP(NN).EQ.0) GO TO 160
                    NNAB = NINP(NN)
                    NNIS = NNIS +1
                    NCONN(1,NNIS) = NNAB
                    IFISNO(NNAB) = NNIS
 160          CONTINUE
            ENDIF
 100  CONTINUE
      NTOTIS = NNIS
C
C     Check the max number of nodes
C
      IF (NTOTIS.LE.200) GO TO 200
        KERROR = KERROR +1
        WRITE(NOWC,3000) NTOTIS
 200  CONTINUE
C
 1000 FORMAT(I5)
 2000 FORMAT(16(I5))
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. PATDEF  * * *',//,
     &  10X,'MAX NUMBER OF INPUT SET NODES  EXCEEDED',//,
     &  10X,'TOTAL NUMBER OF I.S. NODES = ',I4,' (MAX: 200)')
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          F U N C T I O N    P D S H F N                        *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      FUNCTION PDSHFN(G,H,R,K,J,I)
C
C     Evaluate partial derivative of sh. function dN(k)/dc(j) at(G,H,R)
C
C     Parameters
C               G : 1st|   isoparametric coordinate of the location at
C               H : 2nd|   which the partial derivative must be
C               R : 3rd|   evaluated
C
C               K : number of the shape function
C               J : isoparametric coordinate with respect to which
C                   the shape function must be differentiated
C               I : Element type
C                                8 : 2D - 8 nodes  isop. element
C
      INCLUDE 'post_common'
C
      PDSHFN = 0.
C
      IF(I.EQ.8) GO TO 10
         KERROR = KERROR+1
         WRITE (NOWC,1000) I
         GO TO 900
 10   CONTINUE
C
      IF(J.EQ.1.OR.J.EQ.2) GO TO 20
         KERROR = KERROR+1
         WRITE (NOWC,2000) J
         GO TO 900
 20   CONTINUE
C
      GO TO (100,200,300,400,500,600,700,800) K
        KERROR = KERROR +1
        WRITE (NOWI,3000) K
        GO TO 900
C
 100  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H+G)
      GO TO 900
C
 200  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H-G)
      GO TO 900
C
 300  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H+G)
      GO TO 900
C
 400  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H-G)
      GO TO 900
C
 500  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1-H)*G
      IF(J.EQ.2) PDSHFN = -0.5*(1-G)*(1+G)
      GO TO 900
C
 600  CONTINUE
      IF(J.EQ.1) PDSHFN =  0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1+G)*H
      GO TO 900
C
 700  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1+H)*G
      IF(J.EQ.2) PDSHFN =  0.5*(1-G)*(1+G)
      GO TO 900
C
 800  CONTINUE
      IF(J.EQ.1) PDSHFN = - 0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1-G)*H
C
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'THE REQUIRED-COORDINATE-CODE MUST BE 1 (G) OR 2(H)',//,
     &  10X,'REQUIRED COORDINATE CODE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
     &  10X,'NODE NUMBER = ',I4)
C
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   P R E S F N                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE PRESFN
C
C     Evaluate partial derivative of the shape function at the nodes
C     (SFDER)
C
      INCLUDE 'post_common'
      DIMENSION C(3)
C
      DO 50 I = 1,3
 50   C(I) = 0.
C
      IT = ITYPE
C
      DO 200 I = 1,NNELT
             DO 100 M = 1,NDIM
 100         C(M) = DLISCO(I,M,IT)
      DO 200 K = 1,NNELT
      DO 200 J = 1,NDIM
             SFDER(I,K,J) = PDSHFN(C(1),C(2),C(3),K,J,IT)
 200  CONTINUE
C
C
      IF (IOUTG.EQ.0) GO TO 400
         WRITE(NOWG,1000)
         DO 300 I = 1,NNELT
         DO 300 K = 1,NNELT
         DO 300 J = 1,NDIM
                WRITE(NOWG,2000) I,K,J,SFDER(I,K,J)
 300  CONTINUE
 400  CONTINUE
C
 1000 FORMAT(1H1,////,40X,'* * * SHAPE FUNCTION DERIVATIVE * * *',//,
     & 1X,'NODE  SH.FN  J   D(SH.NF)/D(COORD.J)',//)
 2000 FORMAT(1X,I4,3X,I2,4X,I1,4X,F10.5)
```

```
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   P R O G E S                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE PROGES(NOUT,IVAR)
C
C     Process variable IVAR according to the input procedure (KPKP)
C
C     Parameter
C          I/  NOUT  : Serial number of the current step/increment
C
C          I/    IVAR : ID number of the variable to be processed
C
C     Local variables
C           CURVAR(9,J) : value of the variable at the current step
C                         at node J of the input set
C           OLDVAR(9,J) : value of the variable at the preceeding step
C                         at node J of the input set
C           CURALL(9,N) : value of the variable at the current step
C                         at node N of the node set
C           OLDALL(9,N) : value of the variable at the preceeding step
C                         at node N of the node set
C
C           IVCDIM      _\ dimension flag of the current and of the
C           IVODIM       / old variable :
C                                   0 -> scalar
C                                   1 -> vector
C                                   2 -> tensor
C           the components of the variable are stored as :
C
C                           scalar    vector      tensor
C
C             ROW 1           a        v(1)       t(1,1)
C             ROW 2           /        v(2)       t(2,2)
C             ROW 3           /        v(3)       t(3,3)
C             ROW 4           /        /          t(1,2)
C             ROW 5           /        /          t(1,3)
C             ROW 6           /        /          t(2,3)
```

```
C           ROW 7           /         /           t(2,1)
C           ROW 8           /         /           t(3,1)
C           ROW 9           /         /           t(3,2)
C
      INCLUDE 'post_common'
C
      DIMENSION IALL(7),CURVAR(9,200),OLDVAR(9,200)
      DIMENSION         CURALL(9,800),OLDALL(9,800)
C
      DO 5 IST  = 1,7
 5    IALL(IST) = 0
C
      IF(IGRST(IVAR).LE.1)  GO TO 20
        DO 10 J = 1,IGRST(IVAR)-1
 10     IALL(J) = 1
 20   CONTINUE
c
      DO 900 NSTEP = 1,NPROC(IVAR)
            IA = IALL(NSTEP)
C
            GO TO (100,200,300,400,500,600) KPKP(NSTEP,IVAR)
C
C           Read from file 8
C
 100        CONTINUE
C
            IVCDIM = IVDIM(IVAR)
            KEY = KAB(IVAR)
            CALL F8VRIN(NOUT,KEY,IA,CURVAR,CURALL)
            IF(KERROR.GT.0) GO TO 999
          GO TO 700
C
C           read variable from data set
C
 200        CONTINUE
C
            IVCDIM = IVDIM(IVAR)
            ID     = IVCDIM
            IV     = IVAR
            CALL DSVRIN(IV,ID,IA,CURVAR,CURALL)
            IF(KERROR.GT.0) GO TO 999
          GO TO 700
C
C           Evaluate the gradient of OLDALL : CURVAR = grad(OLDALL)
C
 300        CONTINUE
C
            IF(IVODIM.LE.1) GO TO 350
              KERROR = KERROR + 1
              WRITE (NOWC,1000) IVAR,NSTEP
          GO TO 700
 350        CONTINUE
C
            IVCDIM = IVODIM  + 1
            CALL GRADIE(IVODIM,OLDALL,CURVAR)
            IF(KERROR.GT.0) GO TO 999
          GO TO 700
C
C           Rotate the reference frame
C
 400        CONTINUE
C
            IF(IVODIM.GT.0) GO TO 450
              KERROR = KERROR + 1
              WRITE (NOWC,2000) IVAR,NSTEP
          GO TO 700
 450        CONTINUE
C
            IVCDIM = IVODIM
            CALL ROTATE(IVCDIM,OLDVAR,CURVAR)
            IF(KERROR.GT.0) GO TO 999
          GO TO 700
C
C           Evaluate the dot product with the normal to the path
C
 500        CONTINUE
C
            IF(IVODIM.GT.0) GO TO 550
              KERROR = KERROR + 1
              WRITE (NOWC,3000) IVAR,NSTEP
          GO TO 700
 550        CONTINUE
C
            IVCDIM = IVODIM-1
            CALL NORMPR(IVODIM,OLDVAR,CURVAR)
            IF(KERROR.GT.0) GO TO 999
          GO TO 700
C
C           Write CURVAR on file 'postout'
C
```

```
 600        CONTINUE
C
            CALL VPRINT(NOUT,IVAR,NSTEP,IA,IVCDIM,CURVAR,CURALL)
C
 700      CONTINUE
C
            IVODIM = IVCDIM
            DO 800 NC = 1,9
                  DO 730 NNIS = 1,NTOTIS
 730              OLDVAR(NC,NNIS) = CURVAR(NC,NNIS)
            IF(IA.NE.1) GO TO 800
                  DO 760 NNLC = 1,NTNOD
 760              OLDALL(NC,NNLC)= CURALL(NC,NNLC)
C
 800        CONTINUE
 900  CONTINUE
C
      DO 950 NNIS = 1,NTOTIS
      DO 950 NC = 1,9
 950  OUTVAR(IVAR,NC,NNIS) = CURVAR(NC,NNIS)
C
 999  CONTINUE
C
1000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. PROGES  * * *',//,
     &  10X,'A GRADIENT STEP IS NOT ALLOWED ON A TENSOR VARIABLE',//,
     &  10X,'BAD PROCEDURE FOR VARIABLE N. ',I4,' STEP N. ',I4)
2000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. PROGES  * * *',//,
     &  10X,'A ROTATION STEP IS NOT ALLOWED ON A SCALAR VARIABLE',//,
     &  10X,'BAD PROCEDURE FOR VARIABLE N. ',I4,' STEP N. ',I4)
3000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. PROGES  * * *',//,
     &  10X,'A NORM. DOT. PROD. STEP IS NOT ALLOWED ON A SCALAR VR.',//,
     &  10X,'BAD PROCEDURE FOR VARIABLE N. ',I4,' STEP N. ',I4)
C
      RETURN
      END
```

```
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   R E S E T V                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE RESETV
C
C     Reset to zero the memory used in the loop over the step/increment
C
      INCLUDE 'post_common'
C
      DO 200 N = 1,200
      DO 200 J=1,9
      DO 200 K=1,10
              OUTVAR(K,J,N) = 0.
 200  CONTINUE
C
      RETURN
      END
```

```
C
C
C     ***************************************************************
C     *                                                             *
C     *                                                             *
C     *          S U B R O U T I N E   R O T A T E                  *
C     *                                                             *
C     *                                                             *
C     ***************************************************************
C
      SUBROUTINE ROTATE(ID,OLD,CUR)
C
C     Find the components of vector and tensors in a rotated frame with
C                 e1 = unit vector tangent to the path (t)
C                 e2 = unit vector normal to the path  (n)
C                 e3 = e1xe2                           (z)
C     Parameters
C        I/    ID  : Dimension flag of the input field
C                                                     1 : vector field
C                                                     2 : tensor field
C        I/    OLD : Input  field ->coords in the main cartesian frame
C        O/    CUR : Output field ->coords in the t,n,z frame
C
      INCLUDE 'post_common'
      DIMENSION OLD(9,200),CUR(9,200),Q(3,3),Z(3),V(3),T(3,3),TI(3,3)
      DIMENSION VP(3),TP(3,3),QT(3,3)
C
      DO 600 NP = 1,NTOTIS
            DO 100 NR = i,9
 100        CUR(NR,NP) = O.
C
C           Components of the third axis z normal to t & n  (z = txn)
C
            Z(1) = PATANG(2,NP)*PATNOR(3,NP)-PATANG(3,NP)*PATNOR(2,NP)
            Z(2) = PATANG(3,NP)*PATNOR(1,NP)-PATANG(1,NP)*PATNOR(3,NP)
            Z(3) = PATANG(1,NP)*PATNOR(2,NP)-PATANG(2,NP)*PATNOR(1,NP)
C
C           Store the components of the transformation matrix Q and the
C           transpose QT
C
            DO 200 NC = 1,3
                  Q(1,NC) = PATANG (NC,NP)
                  Q(2,NC) = PATNOR (NC,NP)
                  Q(3,NC) = Z (NC)
                  QT(NC,1) = Q(1,NC)
                  QT(NC,2) = Q(2,NC)
                  QT(NC,3) = Q(3,NC)
 200        CONTINUE
C
C
          IF(ID.EQ.2) GO TO 500
C
C           OLD is a vector
C
            DO 300 I = 1,3
 300        V(I) = OLD(I,NP)
C
            CALL DOTPRO(3,1,2,3,9,3,V,Q,VP)
C
            DO 400 I = 1,3
 400        CUR(I,NP) = VP(I)
C
      GO TO 600
C
 500      CONTINUE
C
C           OLD is a tensor
C
            T(1,1) = OLD(1,NP)
            T(2,2) = OLD(2,NP)
            T(3,3) = OLD(3,NP)
            T(1,2) = OLD(4,NP)
            T(1,3) = OLD(5,NP)
            T(2,3) = OLD(6,NP)
            T(2,1) = OLD(7,NP)
            T(3,1) = OLD(8,NP)
            T(3,2) = OLD(9,NP)
C
            CALL DOTPRO(3,2,2,9,9,9,T,QT,TI)
            CALL DOTPRO(3,2,2,9,9,9,Q,TI,TP)
C
            CUR(1,NP) = TP(1,1)
            CUR(2,NP) = TP(2,2)
            CUR(3,NP) = TP(3,3)
            CUR(4,NP) = TP(1,2)
            CUR(5,NP) = TP(1,3)
            CUR(6,NP) = TP(2,3)
            CUR(7,NP) = TP(2,1)
            CUR(8,NP) = TP(3,1)
            CUR(9,NP) = TP(3,2)
```

```
C
 600  CONTINUE
C
      RETURN
      END
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   R W P E R M                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE RWPERM(ID,IMOVE,INEW,AM,AP,AN)
C
C     Change the order of the rows of AM if ZERO elements are on the
C     main diagonal.
C
C     Parameters:
C           I/     ID : matrix dimension
C           0/     IMOVE : row permutation code (=1 : row permuted )
C           0/     INEW : perm. record :INEW(J)= old # of new row #J
C           I/0    AM : square input/output matrix
C            /     AP : matrix used locally to keep record of non-zero
C                         pivots
C            /     AN : matrix used locally to store the new perm. matr.
C
      DIMENSION AM(ID,ID),AP(ID,ID),AN(ID,ID),INEW(ID)
C
      DO 10 NR = 1,ID
            INEW(NR) = 0
      DO 10 NC = 1,ID
            AP(NR,NC) = 0.
            AN(NR,NC) = 0.
 10   CONTINUE
C
      IMOVE = 0
      PIVMIN = 1.E-06
C
C     Check if matrix AM has zero elements on the main diagonal
C
      DO 50 NR = 1,ID
            IF(ABS(AM(NR,NR)).GE.PIVMIN) GO TO 50
            IMOVE = 1
 50   CONTINUE
C
      IF(IMOVE.EQ.0) GO TO 900
```

```
C
C     Permutation required
C
C     Prepare matrix AP with the position of non-zero pivots and
C     vector INEW with the # of rows which have non-zero pivots
C
      DO 100 NPIV = 1,ID
      DO 100 NR   = 1,ID
             IF(ABS(AM(NR,NPIV)).LT.PIVMIN) GO TO 100
               IPOS  = INEW(NPIV)+1
               INEW(NPIV) = IPOS
               AP(NPIV,IPOS) = FLOAT(NR)
 100  CONTINUE
C
C     Permutation
C
      NP = 0
C
 200  CONTINUE
C
      DO 700 NPIV = 1,ID
C
         IF(INEW(NPIV).NE.NP) GO TO 700
C
         IF(NP.EQ.0) THEN
           IMOVE = -1
           WRITE(13,1000) NPIV
           WRITE(13,*) AM
           GO TO 900
         ENDIF
C
         PIVMAX = PIVMIN
C
         DO 300 NCP = 1,NP
            NR = INT(AP(NPIV,NCP))
            PIVOT = ABS(AM(NR,NPIV))
            IF(PIVOT.GE.PIVMAX) THEN
               PIVMAX = PIVOT
               NRCH   = NR
            ENDIF
 300     CONTINUE
C
         INEW(NPIV) = -NRCH
         DO 350 NC = 1,ID
 350     AN(NPIV,NC) = AM(NRCH,NC)
```

```
C
C        Packing of AP
C
         DO 600 NR = 1,ID
            IF(INEW(NR).LT.0) GO TO 600
            DO 500 NC = 1,ID
               IF(INT(AP(NR,NC)).EQ.NRCH) THEN
                 INEW(NR) = INEW(NR) -1
                 DO 400 NLC = NC,ID-1
 400             AP(NR,NLC) = AP(NR,NLC+1)
                 AP(NR,ID)  = 0.
                 GO TO 600
               ENDIF
 500        CONTINUE
 600     CONTINUE
C
         NP = NP-1
         GO TO 200
 700  CONTINUE
C
      NP = NP+1
C
C     Check if all the pivot have been found
C
      INEG = 1
      DO 750 NR = 1, ID
 750  IF(INEW(NR).GT.0) INEG = 0
C
      IF(INEG.EQ.1) GO TO 800
C
      IF(NP.LE.ID) GO TO 200
C
      IMOVE = -1
      WRITE(13,2000)
      WRITE(13,*) INEW
      WRITE(13,3000)
      WRITE(13,*) AM
      WRITE(13,4000)
      WRITE(13,*) AP
      WRITE(13,5000)
      WRITE(13,*) AN
C
 800  CONTINUE
C
      DO 850 NR = 1,ID
```

```
         INEW(NR) = -INEW(NR)
      DO 850 NC = 1,ID
         AM(NR,NC) = AN(NR,NC)
 850  CONTINUE
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. RWPERM  * * *',//,
     & 20X,'NO PIVOT AVAILABLE FOR ROW N.',I4,//,
     & 20X,'INPUT MATRIX AM FOLLOWS ',//)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. RWPERM  * * *',//,
     & 20X,'NP GREATER THAN ID.  NP = ',I4,' ID = ',I4,//,
     & 20X,'PERMUTATION RECORD VECTOR INEW  FOLLOWS ',//)
 3000 FORMAT(1X,//,20X,'INPUT MATRIX AM FOLLOWS',//)
 4000 FORMAT(1X,//,20X,'PIVOT MATRIX AP FOLLOWS',//)
 5000 FORMAT(1X,//,20X,'OUTPUT MATRIX AN FOLLOWS',//)
      RETURN
      END
```

```
C
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   S O L V E R                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE SOLVER(ID,IMOVE,INEW,AU,AL,B,X)
C
C     Solve the systems AL*C = B and AU*X = C
C     If matrix AM has been permuted, the RHS is permuted too
C
C     Parameters
C           I/     ID : matrix/vector dimension
C           I/     IMOVE : row permutation code (=1 : row permuted )
C           I/     INEW : perm. record :INEW(J)= old # of new row #J
C           I/     AL : lower triang matrix AL(ID,ID)
C           I/     AU : upper triang.matrix AU(ID,ID)
C           I/     B  : RHS of the system AL*AU * X = B
C           O/     X  : unknown vector
C
      DIMENSION AU(ID,ID), AL(ID,ID), B(ID), X(ID),INEW(ID)
C
C     Check if rows of AM have been permuted. If YES, permute B
C
      CALL CKPERM(ID,IMOVE,INEW,B,X)
C
      DO 10 NC = 1,ID
 10   X(NC) = 0.
C
C     Evaluate X  -> AL*X = B
C
      DO 200 NR = 1,ID
            SUM = 0.
         IF(NR.EQ.1) GO TO 200
            DO 100 NC = 1,NR-1
 100        SUM = SUM + X(NC) * AL(NR,NC)
 200  X(NR) = B(NR) -SUM
C
C     Copy X in B
```

```
C
      DO 250 I = 1,ID
      B(I) = X(I)
  250 X(I) = 0.

C     Evaluate X  -> AU*X = B
C
      DO 400 NR = ID,1,-1
             SUM = 0.
         IF (NR.EQ.ID) GO TO 400
             DO 300 NC = NR+1,ID
  300        SUM = SUM + X(NC) * AU(NR,NC)
  400 X(NR) = (B(NR)-SUM)/AU(NR,NR)
C
C
      RETURN
      END
```

```
C
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   T A N G 2 D                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE TANG2D(X,T)
C
C     Evaluates the components of the unit tangent vector at the three
C     subsequent nodes on a side af a 2nd ord. isop 2D elt
C
C     Parameters
C          I/    X : Cartesian coordinate of the nodes
C          O/    T : Cartesian components of the unit tangent vector
C                    at the nodes
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ================================================================
C
      DIMENSION X(2,3),T(2,3),DXDS(2,3),SQMOD(3)
C
C     First node
C
      SQMOD(1) = 0.
      DO 100 I = 1,2
            DXDS(I,1) = -1.5 * X(I,1) + 2 * X(I,2) - 0.5 * X(I,3)
  100 SQMOD(1) = SQMOD(1) + DXDS(I,1)**2
C
C     second node
C
      SQMOD(2) = 0.
      DO 200 I = 1,2
            DXDS(I,2) = -0.5 * X(I,1) + 0.5 * X(I,3)
  200 SQMOD(2) = SQMOD(2) + DXDS(I,2)**2
c
c     third node
C
      SQMOD(3) = 0.
      DO 300 I = 1,2
            DXDS(I,3) = 0.5 * X(I,1) - 2 * X(I,2) + 1.5 * X(I,3)
```

```
 300  SQMOD(3) = SQMOD(3) + DXDS(I,3)**2
C
      DO 400 N = 1,3
      DO 400 I = 1,2
 400  T(I,N)  = DXDS(I,N)/SQRT(SQMOD(N))
C
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *         S U B R O U T I N E   U S R S U B                     *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE USRSUB(NOUT,FILN)
C
C     Parameter
C
C          I/  NOUT  : Serial number of the current step/increment
C          I/  FILN  : Job name
C
C
C     Evaluate the energy momentum tensor with:
C                                                OUTVAR(1,J,NNIS) = W
C                                                OUTVAR(2,J,NNIS) = t
C                                                OUTVAR(3,J,NNIS) = du/dn
C
      INCLUDE 'post_common'
      DIMENSION ENMTN(200),T(3),DUDN(3)
      CHARACTER*20 FILN
C
      WRITE(NORD,1000) NKSTEP(NOUT),NKINCR(NOUT)
      WRITE(NOWP(NOUT),4000) FILN,NKSTEP(NOUT),NKINCR(NOUT)
C
C
C
C
      DO 100 NNIS = 1,NTOTIS
C
            DO 10 K = 1,3
                  T(K)   = OUTVAR(2,K,NNIS)
                  DUDN(K)= OUTVAR(3,K,NNIS)
 10         CONTINUE
            CALL DOTPRO(3,1,1,3,3,1,T,DUDN,PR)
C
         WRITE(NORD,3000) OUTVAR(1,1,NNIS),PR
C
            ENMTN(NNIS) = OUTVAR(1,1,NNIS) - PR
```

```
             SCNOR=SCURV(NNIS)/SCURV(NTOTIS)
             WRITE(NOWP(NOUT),6000) SCNOR,ENMTN(NNIS)
 100  CONTINUE
C
 1000 FORMAT(1H1,////,20X,'* * * STEP',I3,' INCR.',I4,' * * *',//,
     &  10X,'ELAST.ENER.',10X,'  T*dU/dn  ',//)
 2000 FORMAT(12X,F8.4,T35,E12.5)
 3000 FORMAT(10X,E12.5,10X,E12.5)
 4000 FORMAT(1X,'normalized arclength',/,1X,'normal component of the',
     &       ' EMT (MPa)',/,'TYPE 3',/,'LABL 2',/,'.5 .5 .1',/
     &       ,A,/,'.5 .2 .1',/,'NSTEP : ',I2,'  NINCR :',I3,/,'END')
 6000 FORMAT(1X,F8.4,2X,E12.5)
      RETURN
      END
```

```
C
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           S U B R O U T I N E   V P R I N T                   *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE VPRINT(NOUT,IV,NST,IA,IDIM,VAR,ALL)
C
C     Print the current value of variable IVAR in file 'postout'
C
C     Parameters
C          I/  NOUT  : Serial number of the current abqstep/increment
C           I/   IV  : Variable ID number
C           I/   NST : Step number
C           I/   IDIM: Dimension flag of the output field
C                                                      0 : scalar field
C                                                      1 : vector field
C                                                      2 : tensor field
C           I/   IA  : All-flag
C                     IA = 0 -> values to be printed only at path-nodes
C                     IA = 1 -> values to be printed at all nodes
C           I/   VAR : Variable values at path nodes
C           I/   ALL : Variable values at all  nodes
C
      INCLUDE 'post_common'
C
      DIMENSION VAR(9,200),ALL(9,800)
C
      WRITE (NOWO,1000) IV,NST
C
      DO 100, NNIS = 1,NTOTIS
C
 100  WRITE (NOWO,2000) NNIS,NCONN(1,NNIS),SCURV(NNIS),
     &                  (VAR(I,NNIS),I = 1,9)
C
C      IF(IA.EQ.0) GO TO 300
C
      WRITE(NOWO,3000) IV,NST
C
      DO 200 NNLC = 1,NTNOD
```

```
C
 200  WRITE(NOWO,4000) NNLC,NABAQ(NNLC),(ALL(I,NNLC),I = 1,9)
C
 300  CONTINUE
C
 1000 FORMAT(1H1,/////,1X,T10,'* * *  V A R I A B L E   N .  ',I2,
     &       '    S T E P   N .  ',I2,'   * * *',//,
     &       1X,T30,'OUTPUT VARIABLE AT PATH NODES',//,
     &       1X,'NNIS NNAB CURV. COORD ',5X,'11',9X ,'22',9X ,'33',9X ,
     &       '12',9X ,'13',9X ,'23',9X ,'21',9X ,'31',9X ,'32',//)
 2000 FORMAT(2(I5),F10.5,2X,9(E11.4))
 3000 FORMAT(1H1,/////,1X,T10,'* * *  V A R I A B L E   N .  ',I2,
     &       '    S T E P   N .  ',I2,'   * * *',//,
     &       1X,T30,'OUTPUT VARIABLE AT ALL NODES',//,
     &       1X,'NNLC NNAB',10X,'11',10X,'22',10X,'33',10X,
     &       '12',10X,'13',10X,'23',10X,'21',10X,'31',10X,'32',//)
 4000 FORMAT(2(I5),5X,9(E12.5))
C
      RETURN
      END
C
```

```
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     * C O M M O N   B L O K S   O F   P R O G R A M   P O S T A B Q  *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
C     Common variables
C
      COMMON / CONN / NCONN(21,200),IELTOP(17,100),NABAQ(800)
      COMMON / COOR / COORDS(3,800)
      COMMON / CTRL / NVAR,JSUB,ITYPE,IOUTG,KPSTOP
      COMMON / CTRV / KPKP(8,10),IVDIM(10),KAB(10),NPROC(10),IGRST(10)
      COMMON / ERRO / KERROR
      COMMON / NUMB / NNELT,NDIM,NTOTIS,NTNOD,NTELT
      COMMON / PATH / PATNOR(3,200),PATANG(3,200),SCURV(200)
      COMMON / SHFN / SFDER(8,8,2)
      COMMON / STEP / NKSTEP(19),NKINCR(19),NTOUT
      COMMON / UNIT / NORI,NOWD,NOWC,NOWO,NOWG,NOWP(19)
      COMMON / VARI / OUTVAR(10,9,200)
C
```

APPENDIX II: THE COMPUTER PROGRAM DOMAIN

```
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *         P R O G R A M   D O M A I N                            *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      PROGRAM DOMAIN §
C
      INCLUDE 'domain_common'
C
C     Common for abaqus routines
C
      COMMON / NAME / FILE_NAME
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513), JRRAY(2,513),LRUNIT(2,1)
      EQUIVALENCE(ARRAY(1),JRRAY(1,1))
C
C     File Handling
C
      CHARACTER*20 FILN
      CHARACTER*25 FILINP
      CHARACTER*25 FILCHK
      CHARACTER*25 FILOUT
      CHARACTER*25 FILGEO
      CHARACTER*25 FILP01
      CHARACTER*25 FILP02
      CHARACTER*25 FILP03
      CHARACTER*25 FILP04
      CHARACTER*25 FILP05
      CHARACTER*25 FILP06
      CHARACTER*25 FILP07
      CHARACTER*25 FILP08
      CHARACTER*25 FILP09
      CHARACTER*25 FILP10
      CHARACTER*25 FILP11
      CHARACTER*25 FILP12
      CHARACTER*25 FILP13
      CHARACTER*25 FILP14
      CHARACTER*25 FILP15
      CHARACTER*25 FILP16
      CHARACTER*25 FILP17
      CHARACTER*25 FILP18
      CHARACTER*25 FILP19
      CHARACTER*25 FILE_NAME
      CHARACTER*4  TINP
      CHARACTER*4  TABQ
      CHARACTER*4  TCHK
      CHARACTER*4  TOUT
      CHARACTER*4  TGEO
      CHARACTER*4  TP01
      CHARACTER*4  TP02
      CHARACTER*4  TP03
      CHARACTER*4  TP04
      CHARACTER*4  TP05
      CHARACTER*4  TP06
      CHARACTER*4  TP07
      CHARACTER*4  TP08
      CHARACTER*4  TP09
      CHARACTER*4  TP10
      CHARACTER*4  TP11
      CHARACTER*4  TP12
      CHARACTER*4  TP13
      CHARACTER*4  TP14
      CHARACTER*4  TP15
      CHARACTER*4  TP16
      CHARACTER*4  TP17
      CHARACTER*4  TP18
      CHARACTER*4  TP19
      CHARACTER*1  RESP
C
      DATA TINP /'.inp'/
      DATA TABQ /'.abq'/
      DATA TCHK /'.chk'/
      DATA TOUT /'.out'/
      DATA TGEO /'.geo'/
      DATA TP01 /'.p01'/
      DATA TP02 /'.p02'/
      DATA TP03 /'.p03'/
      DATA TP04 /'.p04'/
      DATA TP05 /'.p05'/
      DATA TP06 /'.p06'/
      DATA TP07 /'.p07'/
      DATA TP08 /'.p08'/
      DATA TP09 /'.p09'/
      DATA TP10 /'.p10'/
      DATA TP11 /'.p11'/
      DATA TP12 /'.p12'/
      DATA TP13 /'.p13'/
      DATA TP14 /'.p14'/
      DATA TP15 /'.p15'/
      DATA TP16 /'.p16'/
      DATA TP17 /'.p17'/
      DATA TP18 /'.p18'/
      DATA TP19 /'.p19'/
C
      PRINT *,''
      PRINT *,'* * *            PROGRAM DOMAIN                * * *'
      PRINT *,' '
      PRINT *,'  ABAQUS OUTPUT (FILE 8)  MUST BE NAMED  Job_name.abq'
      PRINT *,'  DOMAIN INPUT  FILE      MUST BE NAMED  Job_name.inp'
      PRINT *,'  DOMAIN OUTPUT FILE      WILL BE NAMED  Job_name.out'
      PRINT *,'  DOMAIN CHECK  FILE      WILL BE NAMED  Job_name.chk'
      PRINT *,'  DOMAIN GEOM.  FILE      WILL BE NAMED  Job_name.geo'
      PRINT *,'  DOMAIN PLOT   FILES     WILL BE NAMED  Job_name.p**'
      PRINT *,''
C
      PRINT 1000,' Please enter the Job_name (MAX 20 Char.)'
      READ(*,2000) INM,FILN
C

      FILINP(1:INM) = FILN(1:INM)
      FILINP(INM+1:INM+4) = TINP(1:4)
C
      FILE_NAME(1:INM) = FILN(1:INM)
      FILE_NAME(INM+1:INM+4) = TABQ(1:4)
C
      FILCHK(1:INM) = FILN(1:INM)
      FILCHK(INM+1:INM+4) = TCHK(1:4)
C
      FILOUT(1:INM) = FILN(1:INM)
      FILOUT(INM+1:INM+4) = TOUT(1:4)
C
      FILGEO(1:INM) = FILN(1:INM)
      FILGEO(INM+1:INM+4) = TGEO(1:4)
C
      FILP01(1:INM) = FILN(1:INM)
      FILP01(INM+1:INM+4) = TP01(1:4)
      FILP02(1:INM) = FILN(1:INM)
      FILP02(INM+1:INM+4) = TP02(1:4)
      FILP03(1:INM) = FILN(1:INM)
      FILP03(INM+1:INM+4) = TP03(1:4)
      FILP04(1:INM) = FILN(1:INM)
      FILP04(INM+1:INM+4) = TP04(1:4)
      FILP05(1:INM) = FILN(1:INM)
      FILP05(INM+1:INM+4) = TP05(1:4)
      FILP06(1:INM) = FILN(1:INM)
      FILP06(INM+1:INM+4) = TP06(1:4)
      FILP07(1:INM) = FILN(1:INM)
      FILP07(INM+1:INM+4) = TP07(1:4)
      FILP08(1:INM) = FILN(1:INM)
      FILP08(INM+1:INM+4) = TP08(1:4)
      FILP09(1:INM) = FILN(1:INM)
      FILP09(INM+1:INM+4) = TP09(1:4)
      FILP10(1:INM) = FILN(1:INM)
      FILP10(INM+1:INM+4) = TP10(1:4)
      FILP11(1:INM) = FILN(1:INM)
      FILP11(INM+1:INM+4) = TP11(1:4)
      FILP12(1:INM) = FILN(1:INM)
      FILP12(INM+1:INM+4) = TP12(1:4)
      FILP13(1:INM) = FILN(1:INM)
      FILP13(INM+1:INM+4) = TP13(1:4)
      FILP14(1:INM) = FILN(1:INM)
      FILP14(INM+1:INM+4) = TP14(1:4)
      FILP15(1:INM) = FILN(1:INM)
      FILP15(INM+1:INM+4) = TP15(1:4)
      FILP16(1:INM) = FILN(1:INM)
      FILP16(INM+1:INM+4) = TP16(1:4)
      FILP17(1:INM) = FILN(1:INM)
      FILP17(INM+1:INM+4) = TP17(1:4)
      FILP18(1:INM) = FILN(1:INM)
      FILP18(INM+1:INM+4) = TP18(1:4)
      FILP19(1:INM) = FILN(1:INM)
      FILP19(INM+1:INM+4) = TP19(1:4)
C
      OPEN(UNIT = 10,FILE = FILINP,STATUS = 'OLD',ERR=10)
      GO TO 20
 10   PRINT*,' '
      PRINT*,' '
      PRINT*,'Unable to open file : ',FILINP
      PRINT*,' '
      PRINT*,' '
      PRINT 1000,' Do you want to try again? [Y]'
      READ (*,2000)INUTIL, RESP
      IF (RESP.EQ.'N'.OR.RESP.EQ.'n') STOP
      PRINT 1000,' Choose another Job_name '
      READ (*,2000) INM,FILN
      FILINP(1:INN) = FILN(1:INN)
      FILINP(INN+1:INN+4) = TINP(1:4)
      FILINP(INN+5:25) = ' '
C
      FILE_NAME(1:INN) = FILN(1:INN)
      FILE_NAME(INN+1:INN+4) = TABQ(1:4)
      FILE_NAME(INN+5:25) = ' '
C
      FILCHK(1:INN) = FILN(1:INN)
      FILCHK(INN+1:INN+4) = TCHK(1:4)
      FILCHK(INN+5:25) = ' '
C
      FILOUT(1:INN) = FILN(1:INN)
      FILOUT(INN+1:INN+4) = TOUT(1:4)
      FILOUT(INN+5:25) = ' '
C
      FILGEO(1:INN) = FILN(1:INN)
      FILGEO(INN+1:INN+4) = TGEO(1:4)
      FILGEO(INN+5:25) = ' '
C
      FILP01(1:INN) = FILN(1:INN)
      FILP01(INN+1:INN+4) = TP01(1:4)
      FILP01(INN+5:25) = ' '
      FILP02(1:INN) = FILN(1:INN)
      FILP02(INN+1:INN+4) = TP02(1:4)
      FILP02(INN+5:25) = ' '
      FILP03(1:INN) = FILN(1:INN)
      FILP03(INN+1:INN+4) = TP03(1:4)
      FILP03(INN+5:25) = ' '
      FILP04(1:INN) = FILN(1:INN)
      FILP04(INN+1:INN+4) = TP04(1:4)
      FILP04(INN+5:25) = ' '
      FILP05(1:INN) = FILN(1:INN)
      FILP05(INN+1:INN+4) = TP05(1:4)
      FILP05(INN+5:25) = ' '
      FILP06(1:INN) = FILN(1:INN)
      FILP06(INN+1:INN+4) = TP06(1:4)
      FILP06(INN+5:25) = ' '
      FILP07(1:INN) = FILN(1:INN)
      FILP07(INN+1:INN+4) = TP07(1:4)
      FILP07(INN+5:25) = ' '
      FILP08(1:INN) = FILN(1:INN)
      FILP08(INN+1:INN+4) = TP08(1:4)
      FILP08(INN+5:25) = ' '
      FILP09(1:INN) = FILN(1:INN)
      FILP09(INN+1:INN+4) = TP09(1:4)
      FILP09(INN+5:25) = ' '
      FILP10(1:INN) = FILN(1:INN)
      FILP10(INN+1:INN+4) = TP10(1:4)
      FILP10(INN+5:25) = ' '
      FILP11(1:INN) = FILN(1:INN)
      FILP11(INN+1:INN+4) = TP11(1:4)
      FILP11(INN+5:25) = ' '
      FILP12(1:INN) = FILN(1:INN)
      FILP12(INN+1:INN+4) = TP12(1:4)
      FILP12(INN+5:25) = ' '
      FILP13(1:INN) = FILN(1:INN)
      FILP13(INN+1:INN+4) = TP13(1:4)
      FILP13(INN+5:25) = ' '
      FILP14(1:INN) = FILN(1:INN)
      FILP14(INN+1:INN+4) = TP14(1:4)
      FILP14(INN+5:25) = ' '
      FILP15(1:INN) = FILN(1:INN)
      FILP15(INN+1:INN+4) = TP15(1:4)
      FILP15(INN+5:25) = ' '
      FILP16(1:INN) = FILN(1:INN)
      FILP16(INN+1:INN+4) = TP16(1:4)
      FILP16(INN+5:25) = ' '
      FILP17(1:INN) = FILN(1:INN)
      FILP17(INN+1:INN+4) = TP17(1:4)
      FILP17(INN+5:25) = ' '
      FILP18(1:INN) = FILN(1:INN)
      FILP18(INN+1:INN+4) = TP18(1:4)
      FILP18(INN+5:25) = ' '
      FILP19(1:INN) = FILN(1:INN)
      FILP19(INN+1:INN+4) = TP19(1:4)
      FILP19(INN+5:25) = ' '
C
      OPEN(UNIT = 10,FILE = FILINP,STATUS = 'OLD',ERR=10)
C
 20    CONTINUE
C
      OPEN(UNIT = 13,FILE = FILCHK)
      OPEN(UNIT = 14,FILE = FILOUT)
      OPEN(UNIT = 15,FILE = FILGEO)
      OPEN(UNIT = 16,FILE = FILP01)
      OPEN(UNIT = 17,FILE = FILP02)
      OPEN(UNIT = 18,FILE = FILP03)
      OPEN(UNIT = 19,FILE = FILP04)
      OPEN(UNIT = 20,FILE = FILP05)
      OPEN(UNIT = 21,FILE = FILP06)
      OPEN(UNIT = 22,FILE = FILP07)
      OPEN(UNIT = 23,FILE = FILP08)
      OPEN(UNIT = 24,FILE = FILP09)
      OPEN(UNIT = 25,FILE = FILP10)
      OPEN(UNIT = 26,FILE = FILP11)
      OPEN(UNIT = 27,FILE = FILP12)
      OPEN(UNIT = 28,FILE = FILP13)
      OPEN(UNIT = 29,FILE = FILP14)
      OPEN(UNIT = 30,FILE = FILP15)
      OPEN(UNIT = 31,FILE = FILP16)
      OPEN(UNIT = 32,FILE = FILP17)
      OPEN(UNIT = 33,FILE = FILP18)
      OPEN(UNIT = 34,FILE = FILP19)
C
C
      NORI = 10
      NOWC = 13
      NOWO = 14
      NOWG = 15
      NOWP(1) = 16
      NOWP(2) = 17
      NOWP(3) = 18
      NOWP(4) = 19
      NOWP(5) = 20
      NOWP(6) = 21
      NOWP(7) = 22
      NOWP(8) = 23
      NOWP(9) = 24
      NOWP(10) =25
      NOWP(11) =26
      NOWP(12) =27
      NOWP(13) =28
      NOWP(14) =29
      NOWP(15) =30
      NOWP(16) =31
      NOWP(17) =32
      NOWP(18) =33
      NOWP(19) =34
C
      KERROR = 0
C
C     Accossing file 8
C
      NRU = 1
      LRUNIT(1,1) = 8
      LRUNIT(2,1) = 2
      CALL INITPF(NRU,LRUNIT,LOUTF)
      JUNIT = 8
      CALL DBRNU(JUNIT)
C
C     Geometry input. Connectivity matrices
C
      CALL GEOINP
C
      IF(KERROR.GT.0) GO TO 200
C
C
C    Interface geometry. Curv. coordinate and normal to the interface.
C
      CALL GEOITF
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     Shape function matrices
C
      CALL PRESFN
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     Setting up the coefficient matrix
C
      CALL MATRIX
C
      IF(KERROR.GT.0) GO TO 200
C
C
C     Input the control variables of the procedure
C
      CALL CTRINP
C
      IF(KERROR.GT.0) GO TO 200
C
      IF(KPSTOP.EQ.1) GO TO 300
C
C     Evaluate the normal force for all the step/increment required by user
C
```

```
      DO 100 NOUT = 1,NTOUT
C
C            reset to zero flags and variables
C
             CALL RESETV
C
C            Input the physical quantities of the procedure
C
             CALL VARINP(NOUT)
C
             IF(KERROR.GT.0) GO TO 200
C
C            Evaluating the variation in energy by domain integrals
C
             CALL DOMINT
C
             IF(KERROR.GT.0) GO TO 200
C
C
C            Solving the system
C
             CALL SYSOLV
C
             IF(KERROR.GT.0) GO TO 200
C
C
C            Output the force on the interface
C
             CALL OUTPUT(NOUT,FILN)
C
             IF(KERROR.GT.0) GO TO 200
C
 100  CONTINUE
C
      STOP
C
 200  CONTINUE
C
      WRITE (NOWC,3000) KERROR
      WRITE (NOWO,3000) KERROR
C
 300  CONTINUE
C
 1000 FORMAT($,A,’ : ’)
 2000 FORMAT(Q,A)
```

```
 3000 FORMAT(1H1,//,10X,’* * *  ERROR DETECTED. PROGRAM STOP : KERROR’,
     &’= ’,I5,’ * * *’)
C
      STOP
      END
C
C
C
C     NOTE : Routines follow in alphabetical order
C
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           F U N C T I O N       A L C A L 1                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      FUNCTION ALCAL1(A,B,C)
C
      INCLUDE 'domain_common'
C
C     Evaluate the integral for -1<X<0  of DSQRT(C*X**2 + B*X + A)
C
      IF ( C ) 100,200,300
C
C
C
C     Negative C
C
 100  CONTINUE
C
      TERM1 = B*(DSQRT(A))-(B-2*C)*DSQRT(A-B+C)
      DELTA = B**2 - 4*A*C
      TERM2 = (DELTA/(2*DSQRT(-C)))*(DSIN(B/(DSQRT(DELTA)))-
     &                              DSIN((B-2*C)/(DSQRT(DELTA))))
C
      ALCAL1 = (TERM1 +TERM2) /(4*C)
C
      GO TO 400
C
C
C
C     Zero  C
C
 200  CONTINUE
C
      IF(B.EQ.0) ALCAL1 = DSQRT(A)
C
      IF(B.NE.0) ALCAL1 = (2*(A**1.5-(A-B)**1.5))/(3*B)
C
      GO TO 400
C
C
C
C     Positive C
C
 300  CONTINUE
C
      TERM1 = B*(DSQRT(A))-(B-2*C)*DSQRT(A-B+C)
      TERM2 = ((4*A*C-B**2)/(2*DSQRT(C)))*DLOG((2*DSQRT(C*A)+B)/
     &                                  (2*DSQRT(C*(A-B+C))-2*C+B))
C
      ALCAL1 = (TERM1+TERM2) /(4*C)
C
C
 400  CONTINUE
C
      RETURN
      END
C
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           F U N C T I O N       A L C A L 2                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      FUNCTION ALCAL2(A,B,C)
C
      INCLUDE 'domain_common'
C
C     Evaluate the integral for 0<X<1  of DSQRT(C*X**2 + B*X t A)
C
      IF ( C ) 100,200,300
C
C
C
C     Negative C
C
 100  CONTINUE
C
      TERM1 = (B+2*C)*DSQRT(A+B+C)-B*(DSQRT(A))
      DELTA = B**2 - 4*A*C
      TERM2 = (DELTA/(2*DSQRT(-C)))*(DSIN((B+2*C)/(DSQRT(DELTA)))-
     &                               DSIN(B/(DSQRT(DELTA))))
C
      ALCAL2 = (TERM1 +TERM2) /(4*C)
C
      GO TO 400
C
C
C
C     Zero  C
C
 200  CONTINUE
C
      IF(B.EQ.0) ALCAL2 = DSQRT(A)
C
      IF(B.NE.0) ALCAL2 = (2*((A+B)**1.5-A**1.5))/(3*B)
C
      GO TO 400
C
C
C
C     Positive C
C
 300  CONTINUE
C
      TERM1 = (B+2*C)*DSQRT(A+B+C)-B*(DSQRT(A))
      TERM2 = ((4*A*C-B**2)/(2*DSQRT(C)))*
     & DLOG((2*DSQRT(C*(A+B+C))+2*C+B)/(2*DSQRT(C*A)+B))
C
      ALCAL2 = (TERM1+TERM2) /(4*C)
C
C
 400  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   B Q C A L C                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE BQCALC(IT)
C
C     Evaluates the quadratic form BQUAD(3,3,3).
C     BQUAD(i,j,k) = Integr[-1 +1] (Ni*Nj*dNk/dr *dr)
C
C     Parameters
C         I/     IT: Element type
C                                8 : 2D - 8 nodes  isop. element
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     =================================================================
C

      INCLUDE 'domain_common'
C
      IF ( IT.EQ.8) GO TO 100
         KERROR = KERROR + 1
         WRITE(NOWC,1000) IT
         GO TO 200
 100  CONTINUE
C
      BQUAD(1,1,1) =  1./3.
      BQUAD(1,1,2) = -2./5.
      BQUAD(1,1,3) =  1./15.
      BQUAD(2,2,1) =  8./15.
      BQUAD(2,2,2) = -0.
      BQUAD(2,2,3) = -8./15.
      BQUAD(3,3,1) = -1./15.
      BQUAD(3,3,2) =  2./5.
      BQUAD(3,3,3) = -1./3.
      BQUAD(1,2,1) =  1./5.
      BQUAD(2,1,1) =  1./5.
      BQUAD(1,2,2) = -4./15.
      BQUAD(2,1,2) = -4./15.
      BQUAD(1,2,3) =  1./15.
      BQUAD(2,1,3) =  1./15.
      BQUAD(1,3,1) = -1./30
      BQUAD(3,1,1) = -1./30
      BQUAD(1,3,2) =  0.
      BQUAD(3,1,2) =  0.
      BQUAD(1,3,3) =  1./30
      BQUAD(3,1,3) =  1./30
      BQUAD(2,3,1) = -1./15.
      BQUAD(3,2,1) = -1./15.
      BQUAD(2,3,2) =  4./15.
      BQUAD(3,2,2) =  4./15.
      BQUAD(2,3,3) = -1./5.
      BQUAD(3,2,3) = -1./5.
C
 200  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR.BQCALC  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           S U B R O U T I N E   C C I N C 2                   *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
C
      SUBROUTINE CCINC2(X,AL)
C
C
C     Evaluates the arclength AL(2) between three side-nodes of a second
C     order  2D isoparametric element.
C
C     Parameters
C           I/    X(2,3) : cartesian coordinates of the three nodes
C           O/    AL(2)  : arclength
C                                   AL(1) = node1-node2
C                                   AL(2) = node2-node3
C
      INCLUDE 'domain_common'
      DIMENSION X(2,3),AL(2),XX(2,3,3)
C
      DO 100 N = 1,2
      DO 100 I = 1,3
      DO 100 J = 1,3
C
 100  XX(N,I,J) = X(N,I)*X(N,J)
C
C     Evaluating the constants A,B,C
C
      A = 0.
      B = 0.
      C = 0.
C
      DO 200 I = 1,2
C
      A = A + 0.25*XX(I,1,1) + 0.25*XX(I,3,3) - 0.5*XX(I,1,3)
C
      B = B -      XX(I,1,1) +      XX(I,3,3) +   2*XX(I,1,2)
     &      -    2*XX(I,2,3)
C
```

```
      C = C +      XX(I,1,1) +    4*XX(I,2,2) +     XX(I,3,3)
     &     -    4*XX(I,1,2) +    2*XX(I,1,3) -   4*XX(I,2,3)
C
 200  CONTINUE
C
C
C     Check the values of A,B,C (MUST BE C*X**2+B*X+A > 0 FOR -1<X<1)
C
      RLARG = 0.
      RSMAL = 0.
C
      DELTA = B**2 - 4*A*C
      IF(DELTA.LE.0.OR.C.LE.0) GO TO 220
      RLARG = (B+DSQRT(DELTA))/(2*C)
      RSMAL = (B-DSQRT(DELTA))/(2*C)
C
 220  CONTINUE
C
      IF( (A.LT.0)                                          .OR.
     &    ((-C-B).GT.A)                                     .OR.
     &    (C.GT.0.AND.DELTA.GT.0.AND.B.GT.0.AND.RSMAL.LT.1) .OR.
     &    (C.GT.0.AND.DELTA.GT.0.AND.B.LT.0.AND.RLARG.GT.-1)) GO TO 250
      GO TO 300
C
 250  CONTINUE
      WRITE (NOWC,1000) A,B,C,X
      KERROR = KERROR +1
      GO TO 400
C
 300  CONTINUE
C
      AL(1) = ALCAL1(A,B,C)
      AL(2) = ALCAL2(A,B,C)
C
 400  CONTINUE
C
C
1000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CCINC2  * * *',//,
     &              20X,' BAD VALUES FOR CONSTANTS A,B,C',//,
     &              20X,' A = ',E12.5,' B = ',E12.5,' C = ',E12.5,//,
     &              20X,'NODE COORDINATES ARE :',/
     &              20X,'                      X1               X2  ',
     &             /,20X,'     NODE1',2(6X,F10.5),/,
     &              20X,'     NODE2',2(6X,F10.5),/,
     &              20X,'     NODE3',2(6X,F10.5),/)
```

```
C
      RETURN
      END
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *         S U B R O U T I N E   C H K I T F                    *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE CHKITF
C
C     Check for the double nodes on the interf. if the coords are the same
C
      INCLUDE 'domain_common'
C
      SMALL = 1.D-20
C
      DO 100 NNIS = 1,NTOTIS
         NNLC1 = NCONN(2,1,NNIS)
         NNLC2 = NCONN(2,2,NNIS)
      DO 100 ND = 1,NDIM
         ABSDIF = DABS(COORDS(ND,NNLC1) - COORDS(ND,NNLC2))
         IF(ABSDIF.LT.SMALL) GO TO 100
C           KERROR = KERROR +1
           WRITE(NOWC,1000)NNIS,ND,NCONN(1,1,NNIS),NCONN(1,2,NNIS)
 100  CONTINUE
C
 1000 FORMAT(1H1,//,20X,'* * * WARNING OF SBR. CHKITF  * * *',//,
     & 10X,'THE ', I3,'th NODES OF THE INTERFACE DO NOT HAVE THE SAME',
     & 1X,I3,' COORDINATE ',//,10X,'ABAQUS NODE # ',I5,' AND ',I5)
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *         S U B R O U T I N E   C H K S E T                    *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE CHKSET(ISET,NAB,NLC)
C
C     Find the local identity number
C
C     Parameters
C           I/    ISET : Set flag
C                             ISET = 0   : Element set
C                             ISET = 1   : Node input set
C                             ISET = 2   : Node set
C           I/    NAB  : Abaqus number
C           O/    NLC  : Local number: NLC =
C                                              IELC (ISET = 0)
C                                          +/- NNIS (ISET = 1)
C                                              NNLC (ISET = 2)
C                                      NLC = 0 if the node/elt doesn't
C                                             belong to the set
C
      INCLUDE 'domain_common'
C
      NLC = 0
C
      GO TO (100,200) ISET
C
C
C     Element set
C
      DO 50 IE = 1,NTELT
            IF (IELTOP(1,IE).NE.NAB)GO TO 50
            NLC = IE
            GO TO 300
 50   CONTINUE
      GO TO 300
C
C
```

```
C     Node input set
C
 100  CONTINUE
C
      DO 150 NNIS = 1,NTOTIS
      DO 150 ISIDE = 1,IDOUBL
             IF(NCONN(1,ISIDE,NNIS).NE.NAB) GO TO 150
             NLC = NNIS
             IF(ISIDE.EQ.1) NLC = -NNIS
             GO TO 300
 150  CONTINUE
      GO TO 300
C
C
C     Node set
C
 200  CONTINUE
C
      DO 250 IN = 1,NTNOD
             IF(NABAQ(IN).NE.NAB) GO TO 250
             NLC = IN
             GO TO 300
 250  CONTINUE
C
 300  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   C K P E R M                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE CKPERM(ID,IMOVE,INEW,B,X)
C
C     Check if in the matrix AM rows have been permuted. If yes(IMOVE=1)
C     it permutes the Right Hand Side B according to the order given by
C     vector INEW
C
      INCLUDE 'domain_common'
C
      DIMENSION INEW(ID),B(ID),X(ID)
C
      IF(IMOVE.NE.1) GO TO 500
C
        DO 100 NR = 1,ID
 100    X(NR) = B(INEW(NR))
        DO 200 NR = 1,ID
 200    B(NR) = X(NR)
C
 500  CONTINUE
C
      RETURN
      END
C
```

```
C
C    ****************************************************************
C    *                                                              *
C    *                                                              *
C    *          S U B R O U T I N E   C O N N E C                   *
C    *                                                              *
C    *                                                              *
C    ****************************************************************
C
C
      SUBROUTINE CONNEC(IFISNO,IDONE)
C
C
C     Evaluates the connectivity matrices NCONN , IELTOP & NABAQ
C     Evaluates the total number of nodes NTNOD and element NTELT
C
C     Parameters
C         I/      IFISNO(10000) :local input-set-number for abaqus nodes
C         O/      IDONE (10000) :local number for abaqus nodes
C
      INCLUDE 'domain_common'
      DOUBLE PRECISION ARRAY
      DIMENSION IFISNO(10000), IDONE(10000)
      DIMENSION ARRAY(513), JRRAY(2,513)
      EQUIVALENCE (ARRAY(1), JRRAY(1,1))
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C
C     Scanning file 8
C
C
      NTNOD = 0
      NTELT =1
C
      DO 500 K =1,99999
C
          CALL DBFILE(0,ARRAY,JRCD)
          IF(JRCD.NE.0) GO TO 600
C
          LR   = JRRAY(1,1)
          KEY  = JRRAY(1,2)
          IF(KEY.NE.1900) GO TO 500
```

```
          IEAB = JRRAY(1,3)
          IELTOP(1,NTELT) = IEAB
C
C         fill IELTOP
C
          DO 200 NN = 1,NNELT
             NNAB = JRRAY(1,NN+4)
C
C            Check if the node has been already numbered in the
C            local list. If not -> put it in the list.
C
             IF(IDONE(NNAB).NE.0) GO TO 50
               NTNOD = NTNOD+1
               IDONE(NNAB) = NTNOD
               NABAQ(NTNOD) = NNAB
C
   50        CONTINUE
C
C            Check if NNAB is a node of the input set
C
             IF(IFISNO(NNAB).EQ.0) GO TO 100
C
C               NNAB is a node of the input set -> fill NCONN
C
                NNIS = IFISNO(NNAB)
                IF(NNIS.LT.0) THEN
                  ISIDE = 1
                  NNIS = -NNIS
                ELSE
                  ISIDE = 2
                ENDIF
                NCONN(2,ISIDE,NNIS) = IDONE(NNAB)
                KP = 4 + NCONN(3,ISIDE,NNIS)*3
                NCONN(3,ISIDE,NNIS) = NCONN(3,ISIDE,NNIS) + 1
C
C               check max number of element to which NNIS belongs
C
                IF(NCONN(3,ISIDE,NNIS).LE.6) GO TO 80
                  KERROR = KERROR +1
                  WRITE(NOWC,1000) NNAB
                  GO TO 700
   80           CONTINUE
C
C               filling NCONN
C
```

```
                NCONN(KP,ISIDE,NNIS)   = IEAB
                NCONN(KP+1,ISIDE,NNIS) = NTELT
                NCONN(KP+2,ISIDE,NNIS) = NN
C
 100           CONTINUE
C
               IA = 2+(NN-1)*2
               IELTOP(IA,NTELT)   = NNAB
               IELTOP(IA+1,NTELT) = IDONE(NNAB)
 200       CONTINUE
           NTELT = NTELT +1
C
C     Check max number of elt's (1000) and nodes (3200)
C
      IF(NTELT.LT.1000.AND.NTNOD.LT.3200) GO TO 500
      WRITE(NOWC,2000) NTELT,NTNOD
      GO TO 600
C
 500  CONTINUE

 600  CONTINUE
C
      NTELT = NTELT -1
C
 700  CONTINUE
C
      IF (IOUTG.EQ.0) GO TO 900
C
      DO 790 ISIDE = 1,IDOUBL
        WRITE(NOWG,3000)ISIDE
C
        DO 750 NNIS = 1,NTOTIS
          IMAX = NCONN(3,ISIDE,NNIS)*3+3
 750    WRITE(NOWG,4000) NNIS,(NCONN(I,ISIDE,NNIS),I=1,IMAX)
 790  CONTINUE
C
      WRITE(NOWG,5000)
C
      DO 800 IELC = 1,NTELT
      IMAX = NNELT*2 + 1
 800  WRITE(NOWG,6000) IELC,(IELTOP(I,IELC),I = 1,IMAX)
C
      WRITE(NOWG,7000)
C
```

```
      NAPPO = (NTNOD-1)/8+1
      DO 850 NROW = 1,NAPPO
      NF = (NROW-1)*8
 850  WRITE(NOWG,8000) (NF+K,NABAQ(NF+K),K = 1,8)
C
 900  CONTINUE
C
1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CONNEC  * * *',//,
    &  10X,'MAX NUMBER OF ELEMENTS CONNECTED TO ONE NODE EXCEEDED',//,
    &  10X,'MORE THAN 6 ELEMENTS ARE CONNECTED TO NODE ',I4)
2000 FORMAT(1H1,///,20X,'* * *WARNING OF  SBR. CONNEC  * * *',//,
    &  10X,'MAX NUMBER OF ELEMENTS and/or NODES EXCEEDED',//,
    &  10X,'TOTAL NUMBER OF READ ELEMENTS = ',I4,' (MAX:1000)',//,
    &  10X,'TOTAL NUMBER OF READ NODES = ',I4,' (MAX: 3200)')
3000 FORMAT(1H1,///,40X,'* * * N C O N N * * *',/,
    &  40X,'* * * I S I D E  =  ',I2,'  * * *',//,
    &  1X,'NNIS NNAB NNLC NBE IAB1 LC1 P1 IAB2 LC2 P2 IAB3 LC3 P3 ',
    &      'IAB4 LC4 P4 IAB5 LC5 P5 IAB6 LC6 P6',/)
4000 FORMAT(1X,I3,2X,I4,1X,I3,1X,I3,6(1X,I4,1X,I3,1X,I2))
5000 FORMAT(1H1,///,40X,'* * * I E L T O P * * *',//,
    &  1X,'IELC IEAB NAB1 NLC1 NAB2 NLC2 NAB3 NLC3 NAB4 NLC4 NAB5 ',
    &      'NLC5 NAB6 NLC6 NAB7 NLC7 NAB8 NLC8',/)
6000 FORMAT(18(1X,I4))
7000 FORMAT(1H1,///,40X,'* * * N A B A Q * * *',//,
    &  1X,8('NNLC NNAB  '),/)
8000 FORMAT(8(2(1X,I4),1X))
C
      RETURN
      END
C
```

```
C
C
C     *******************************************************************
C     *                                                                 *
C     *                                                                 *
C     *          S U B R O U T I N E   C T R I N P                      *
C     *                                                                 *
C     *                                                                 *
C     *******************************************************************
C
C
      SUBROUTINE CTRINP
C
C     Reads the control variables of the procedure
C
      INCLUDE 'domain_common'
C
      READ(NORI,3000) NCNTOU
C
      DO 100 ISIDE = 1,2
C
 100  READ(NORI,4000) (NLAYER(ISIDE,NCONT),NCONT = 1,10)
C
C     Check the maximum number of contour and layers
C
      IF(NCNTOU.LE.10) GO TO 200
         KERROR = KERROR +1
         WRITE(NOWC,1000) NCNTOU
         GO TO 400
 200  CONTINUE
C
      DO 300 J = 1,NCNTOU
      DO 300 I = 1,2
            IF(NLAYER(I,J).LE.15) GO TO 300
            KERROR = KERROR +1
            WRITE(NOWC,2000) J,I,NLAYER(I,J)
 300  CONTINUE
C
      READ(NORI,5000) NTOUT,ICREEP
C
      DO 350 NOUT = 1,NTOUT
 350        READ(NORI,5000)NKSTEP(NOUT),NKINCR(NOUT)
C
 400  CONTINUE
C
```

```
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     &  10X,'MAXIMUM NUMBER OF CONTOUR EXCEEDED (MAX = 10)',//,
     &  10X,'REQUIRED NUMBER OF CONTOUR = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. CTRINP  * * *',//,
     &  10X,'MAXIMUM NMB. OF LAY. PER CONTOUR EXCEEDED (MAX = 15)',//,
     &  10X,'LAYER',I4,' ISIDE',I4,' REQUIRED NUMBER OF LAYERS = ',I4)
 3000 FORMAT(I5)
 4000 FORMAT(10(I5))
 5000 FORMAT(2(I5))
C
      RETURN
      END
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          F U N C T I O N    D L I S C O                        *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      FUNCTION DLISCO(L,K,J,I,IR)
C
C     Evaluates the dimensionless isoparametric coordinates of the nodes
C     and of the integration points
C
C     Parameters
C                L : identity flag: 0-> node  1-> int.pt.
C                K : node/int.pt  number
C                J : required coordinate
C                                        1 : G
C                                        2 : H
C                                        3 : R
C                I : Element type
C                                 8 : 2D - 8 nodes  isop. element
C                IR: Reduced integration flag
C                                 0 : full integration (9 i.p.)
C                                 1 : reduced integration (4 i.p.)
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     =================================================================
C
      INCLUDE 'domain_common'
C
      DLISCO = 0.
      FIV = .577350269189626
      SEV = .774596669241483
C
      IF ( I.EQ.8) GO TO 100
         KERROR = KERROR + 1
         WRITE(NOWC,1000) I
         GO TO 900
 100  CONTINUE
C
```

```
      IF(L.EQ.1) GO TO 400
C
C     node coords
C
      GO TO (200,300) J
         KERROR = KERROR + 1
         WRITE(NOWC,2000) J
         GO TO 900
C
C     First coordinate : G
C
 200  CONTINUE
C
      GO TO (210,220,230,240,250,260,270,280) K
C
         KERROR = KERROR + 1
         WRITE(NOWC,3000) K
         GO TO 900
C
 210  CONTINUE
      DLISCO = -1.
      GO TO 900
C
 220  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 230  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 240  CONTINUE
      DLISCO = -1.
      GO TO 900
C
 250  CONTINUE
      DLISCO =  0.
      GO TO 900
C
 260  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 270  CONTINUE
      DLISCO =  0.
      GO TO 900
C
 280  CONTINUE
      DLISCO = -1.
      GO TO 900
C
C     Second coordinate : H
C
 300  CONTINUE
C
      GO TO (310,320,330,340,350,360,370,380) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,3000) K
          GO TO 900
C
 310  CONTINUE
      DLISCO = -1.
      GO TO 900
C
 320  CONTINUE
      DLISCO = -1.
      GO TO 900
C
 330  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 340  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 350  CONTINUE
      DLISCO = -1.
      GO TO 900
C
 360  CONTINUE
      DLISCO =  0.
      GO TO 900
C
 370  CONTINUE
      DLISCO =  1.
      GO TO 900
C
 380  CONTINUE
```

```
      DLISCO =  0.
      GO TO 900
C
 400  CONTINUE
C
C     integration points coords
C
      IF(IR.EQ.1) GO TO 700
C
C     Full integration
C
      GO TO (500,600) J
          KERROR = KERROR + 1
          WRITE(NOWC,2000) J
          GO TO 900
C
C     First coordinate : G
C
 500  CONTINUE
C
      GO TO (510,520,530,540,550,560,570,580,590) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,4000) K
          GO TO 900
C
 510  CONTINUE
      DLISCO = -SEV
      GO TO 900
C
 520  CONTINUE
      DLISCO = 0.
      GO TO 900
C
 530  CONTINUE
      DLISCO = SEV
      GO TO 900
C
 540  CONTINUE
      DLISCO = -SEV
      GO TO 900
C
 550  CONTINUE
      DLISCO = 0.
      GO TO 900
```

```
C
  560 CONTINUE
      DLISCO = SEV
      GO TO 900
C
  570 CONTINUE
      DLISCO = -SEV
      GO TO 900
C
  580 CONTINUE
      DLISCO = 0.
      GO TO 900
C
  590 CONTINUE
      DLISCO = SEV
      GO TO 900
C
C     Second coordinate : H
C
  600 CONTINUE
C
      GO TO (610,620,630,640,650,660,670,680,690) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,4000) K
          GO TO 900
C
  610 CONTINUE
      DLISCO = -SEV
      GO TO 900
C
  620 CONTINUE
      DLISCO = -SEV
      GO TO 900
C
  630 CONTINUE
      DLISCO = -SEV
      GO TO 900
C
  640 CONTINUE
      DLISCO = 0.
      GO TO 900
C
  650 CONTINUE
      DLISCO = 0.
      GO TO 900
C
  660 CONTINUE
      DLISCO = 0.
      GO TO 900
C
  670 CONTINUE
      DLISCO = SEV
      GO TO 900
C
  680 CONTINUE
      DLISCO = SEV
      GO TO 900
C
  690 CONTINUE
      DLISCO = SEV
      GO TO 900
C
  700 CONTINUE
C
C     Reduced integration
C
      GO TO (800,850) J
          KERROR = KERROR + 1
          WRITE(NOWC,2000) J
          GO TO 900
C
C     First coordinate : G
C
  800 CONTINUE
C
      GO TO (810,820,830,840) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,5000) K
          GO TO 900
C
  810 CONTINUE
      DLISCO = -FIV
      GO TO 900
C
  820 CONTINUE
      DLISCO = FIV
      GO TO 900
C
```

```
 830  CONTINUE
      DLISCO = -FIV
      GO TO 900
C
 840  CONTINUE
      DLISCO = FIV
      GO TO 900
CC
C     Second coordinate : H
C
 850  CONTINUE
C
      GO TO (860,870,880,890) K
C
          KERROR = KERROR + 1
          WRITE(NOWC,5000) K
          GO TO 900
C
 860  CONTINUE
      DLISCO = -FIV
      GO TO 900
C
 870  CONTINUE
      DLISCO = -FIV
      GO TO 900
C
 880  CONTINUE
      DLISCO = FIV
      GO TO 900
C
 890  CONTINUE
      DLISCO = FIV
      GO TO 900
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
    &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
    &  10X,'ELEMENT-TYPE = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
    &  10X,'THE REQUIRED-COORDINATE-CODE MUST BE 1 (G) OR 2(H)',//,
    &  10X,'REQUIRED COORDINATE CODE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
    &  10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
    &  10X,'NODE NUMBER = ',I4)
```

```
 4000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
    &  10X,'FOR THIS EL-TYPE THE IT.P NUMBER MUST BE BETW. 1 AND 9',//,
    &  10X,'IT.P NUMBER = ',I4)
 5000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.DLISCO  * * *',//,
    &  10X,'FOR THIS EL-TYPE THE IT.P NUMBER MUST BE BETW. 1 AND 4',//,
    &  10X,'IT.P NUMBER = ',I4)
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   D O M I N T                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE DOMINT
C
C     Evaluates the domain integral matrix DELEN
C
      INCLUDE 'domain_common'
C
      DIMENSION LAYREQ(15),IDOTOP(9,15),PARINT(15),TOTINT(15)
C
      DO 500 ISIDE = 1,2
C
         LAYMAX = 0
         DO 50 NLAY = 1,15
 50      LAYREQ(NLAY) = 0
         DO 100 NCONT = 1,NCNTOU
            IF(NLAYER(ISIDE,NCONT).GT.LAYMAX)LAYMAX=NLAYER(ISIDE,NCONT)
 100     LAYREQ(NLAYER(ISIDE,NCONT))= 1
C
         DO 400 NNIS = 1,NTOTIS
C
            CALL IDTCAL(NNIS,ISIDE,LAYMAX,IDOTOP)
            IF (KERROR.GT.0) GO TO 900
C
            CALL INTCAL(NNIS,ISIDE,LAYMAX,LAYREQ,IDOTOP,PARINT,TOTINT)
            IF (KERROR.GT.0) GO TO 900
C
            DO 300 NLAY = 1,LAYMAX
C
               IF(LAYREQ(NLAY).EQ.0)GO TO 300
               SUM = 0.
               DO 200 NINFL = 1,NLAY-1
 200           SUM = SUM+TOTINT(NINFL)
 300        DINTG(ISIDE,NLAY,NNIS) = SUM + PARINT(NLAY)
 400     CONTINUE
 500  CONTINUE
C
      DO 700 NCONT = 1,NCNTOU
C
         NLINT = NLAYER(1,NCONT)
         NLOUT = NLAYER(2,NCONT)
         DO 600 NNIS = 1,NTOTIS
 600     DELEN(NCONT,NNIS) = DINTG(1,NLINT,NNIS) + DINTG(2,NLOUT,NNIS)
C
 700  CONTINUE
c
      WRITE(NOWO,1000)
      WRITE(NOWO,2000)(NC,NC=1,NCNTOU)

      DO 800 NNIS = 1,NTOTIS
C
              WRITE(NOWO,3000) SCURV(NNIS),(DELEN(K,NNIS),K=1,NCNTOU)
 800  CONTINUE
C
 900  CONTINUE
C
 1000 FORMAT(1H1,////,20X,'* * *  E N E R G Y   V A R I A T I O N',
     &       ' * * *',/)
 2000 FORMAT(/,1X,'CURV.COOR ',10(' CONT.#',I2,'   '))
 3000 FORMAT(1X,F8.4,2X,10(1X,E11.4))
C
      RETURN
      END
C
```

```
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   F 8 I P I N                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE F8IPIN(NOUT,KREQ,VARELT)
C
C     Read variable data from Abaqus-file 8 at element integration points
C
C     Parameters
C          I/  NOUT  : Serial number of the required step/increment
C          I/  KREQ  : Abaqus file 8-read-key for the required variable
C          O/  VARELT: Variable values at integration points
C
C                    VARELT(I,J,K) = value of the Ith component
C                                    at int.  pt. J
C                                    of element   K
C            the components of the variable are stored as :
C
C                              scalar    vector      tensor
C
C              ROW 1             a        v(1)       t(1,1)
C              ROW 2             /        v(2)       t(2,2)
C              ROW 3             /        v(3)       t(3,3)
C              ROW 4             /        /          t(1,2)
C              ROW 5             /        /          t(1,3)
C              ROW 6             /        /          t(2,3)
C              ROW 7             /        /          t(2,1)
C              ROW 8             /        /          t(3,1)
C              ROW 9             /        /          t(3,2)
C
      INCLUDE 'domain_common'
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),JRRAY(2,513)
      DIMENSION JV(9,1000),VECT(9),VARELT(9,9,1000)
      EQUIVALENCE (ARRAY(1),JRRAY(1,1))
C
      DO 10 K=1,1000
      DO 10 J= 1,9
   10 JV(J,K) = 0
```

```
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C     Scanning file 8
C
      DO 330 K = 1,999999
            CALL DBFILE(0,ARRAY,JRCD)
        IF(JRCD.NE.0) GO TO 350
            LR   = JRRAY(1,1)
            KEY  = JRRAY(1,2)
C
C             Finding the right step/increment
C
              IF (KEY.NE.2000) GO TO 330
              NST = JRRAY(1,8)
              NIN = JRRAY(1,9)
              IF(NKSTEP(NOUT).NE.NST.OR.NKINCR(NOUT).NE.NIN) GO TO 330
C
      JF = 1
      DO 300 KK = 1,999999
              CALL DBFILE(0,ARRAY,JRCD)
         IF(JRCD.NE.0) GO TO 350
              LR   = JRRAY(1,1)
              KEY  = JRRAY(1,2)
         IF(KEY.EQ.2001) GO TO 350
              IEAB = JRRAY(1,3)
              IP   = JRRAY(1,4)
              ILOC = JRRAY(1,6)
C
         IF(KEY.NE.1.OR.ILOC.NE.0) GO TO 300
C
C                The subsequent record of file 8 contains
C                values at integration points of elt IEAB
C
                 CALL DBFILE(0,ARRAY,JRCD)
                 LR = JRRAY(1,1)
                 KEY = JRRAY(1,2)
C
         IF(KEY.NE.KREQ) GO TO 300
C
C                Fill VARELT
C
                 CALL FILLIN(KREQ,LR,ARRAY,VECT)
```

```
               CALL CHKSET(0,IEAB,IELC)
               IF (IELC.GT.0) GO TO 150
                  WRITE(NOWC,2000) IEAB
                  GO TO 300
 150           CONTINUE
               DO 200 I = 1,9
 200           VARELT(I,IP,IELC) = VECT(I)
               JV(IP,IELC) = 1
C
 300  CONTINUE
 330  CONTINUE
 350  CONTINUE
C
C     Check if the required step/incr has been found
C
      IF(JF.EQ.1) GO TO 400
            KERROR = KERROR +1
            WRITE(NOWC,3000) NKINCR(NOUT),NKSTEP(NOUT)
 400  CONTINUE
C
C     Check if all int.pts. have been found
C
      DO 500 K = 1,NTELT
      DO 500 J = 1, NINTP
            IF(JV(J,K).EQ.1) GO TO 500
            KERROR = KERROR +1
            WRITE(NOWC,1000) J,IELTOP(1,K),KREQ
 500  CONTINUE
C
 600  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8IPIN  * * *',//,
     &  10X,'INT.PT. ',I4,' OF ELT',I4,'  HAS NOT BEEN FOUND',
     &   ' FOR KEY = ',I4)
 2000 FORMAT(1H1,///,20X,'* * *WARNING OF SBR. F8IPIN  * * *',//,
     &  10X,'LOCAL NUMBER OF THE ELEMENT',I4,' HAS NOT BEEN FOUND')
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8IPIN  * * *',//,
     &  10X,'INCREMENT ',I4,' OF STEP',I4,'  HAS NOT BEEN FOUND')
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   F 8 N O I N                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE F8NOIN(NOUT,KREQ,VARNGD)
C
C     Read variable data  at nodes from Abaqus-file 8
C
C     Parameters
C          I/  NOUT  : Serial number of the required step/increment
C          I/  KREQ  : Abaqus file 8-read-key for the required variable
C          O/  VARNOD: Variable values at all  nodes
C
C                    VARNOD(I,J) = value of the Ith component
C                                  at node J
C            the components of the variable are stored as :
C
C                              scalar     vector       tensor
C
C              ROW 1              a        v(1)        t(1,1)
C              ROW 2              /        v(2)        t(2,2)
C              ROW 3              /        v(3)        t(3,3)
C              ROW 4              /        /           t(1,2)
C              ROW 5              /        /           t(1,3)
C              ROW 6              /        /           t(2,3)
C              ROW 7              /        /           t(2,1)
C              ROW 8              /        /           t(3,1)
C              ROW 9              /        /           t(3,2)
C
C
      INCLUDE 'domain_common'
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),JRRAY(2,513)
      DIMENSION JV(3200),VARNOD(9,3200),VECT(9)
      EQUIVALENCE (ARRAY(1),JRRAY(1,1))
C
      DO 10 K=1,3200
 10   JV(K) = 0
```

```
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C     Scanning file 8
C
      DO 330 K = 1,999999
             CALL DBFILE(0,ARRAY,JRCD)
         IF(JRCD.NE.0) GO TO 350
             LR    = JRRAY(1,1)
             KEY   = JRRAY(1,2)
C
C            Finding the right step/increment
C
             IF (KEY.NE.2000) GO TO 330
             NST = JRRAY(1,8)
             NIN = JRRAY(1,9)
             IF(NKSTEP(NOUT).NE.NST.OR.NKINCR(NOUT).NE.NIN) GO TO 330
C
      JF = 1
      DO 300 KK = 1,999999
             CALL DBFILE(0,ARRAY,JRCD)
         IF(JRCD.NE.0) GO TO 350
             LR    = JRRAY(1,1)
             KEY   = JRRAY(1,2)
         IF(KEY.EQ.2001) GO TO 350
             NNAB = JRRAY(1,3)
C
C            Check if a node or an element variable is required
C
             IF(KREQ.GE.100) GO TO  50
C
C               Element variable is required
C
                ILOC = JRRAY(1,6)
         IF(KEY.NE.1.OR.ILOC.NE.4) GO TO 300
C
C               The subsequent record of file 8 contains nodal averaged
C               values at node NNAB
C
                CALL DBFILE(0,ARRAY,JRCD)
                LR = JRRAY(1,1)
                KEY = JRRAY(1,2)
C
```

```
C            start from here if nodal variable is required
C
  50         CONTINUE
C
         IF(KEY.NE.KREQ) GO TO 300
C
C               Fill VARNOD
C
                CALL FILLIN(KREQ,LR,ARRAY,VECT)
                CALL CHKSET(2,NNAB,NNLC)
                IF (NNLC.GT.0) GO TO 100
                  WRITE(NOWC,1000) NNAB
                  GO TO 300
 100            CONTINUE
                DO 200 I = 1,9
 200            VARNOD(I,NNLC) = VECT(I)
                JV(NNLC) = 1
C
 300  CONTINUE
 330  CONTINUE
 350  CONTINUE
C
C     Check if the required step/incr has been found
C
      IF(JF.EQ.1) GO TO 400
             KERROR = KERROR +1
             WRITE(NOWC,3000) NKINCR(NOUT),NKSTEP(NOUT)
 400  CONTINUE
C
C
C     Check if all nodes have been found
C
      DO 500 K = 1,NTNOD
             IF(JV(K).EQ.1) GO TO 500
             KERROR = KERROR + 1
             WRITE( NOWC,2000 ) K,KREQ
 500  CONTINUE
C
 600  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * WARNING OF SBR. F8NOIN  * * *',//,
     &  10X,'LOCAL NUMBER FOR NODE ',I4,' HAS NOT BEEN FOUND')
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8NOIN  * * *',//,
     &  10X,'NODE NUMBER',I4,'OF THE NODE SET HAS NOT BEEN FOUND',
     &  ' FOR KEY = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. F8NOIN  * * *',//,
     &  1CX,'INCREMENT ',I4,' OF STEP',I4,'  HAS NOT BEEN FOUND')
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *             S U B R O U T I N E   F I L L I N                 *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE FILLIN(KEY,LR,ARRAY,VECT)
C
C     Fills VECT with the values given by ARRAY according to rules
C     defined by KEY
C
C     Parameters
C         I/      KEY  : Abaqus  file-8-reading key
C         I/      LR   : Record length of ARRAY
C         I/      ARRAY: Input record read from file 8
C         O       VECT : internal variable input array
C
      INCLUDE 'domain_common'
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),VECT(9)
C
      DO 10 I = 1,9
 10   VECT(I) = 0.
C
      IF(KEY.GE.100) GO TO 400
C
C     Element variable
C
      IF(KEY.EQ.2.OR.KEY.EQ.14) GO TO 100
      IF(KEY.EQ.11) GO TO 200
      IF(KEY.GE.21.AND .KEY.LE.25) GO TO 300
C
      KERROR = KERROR + 1
      WRITE (NOWC,1000) KEY
      GO TO 700
C
C     1 component-variable
C
 100  CONTINUE
C
      VECT(1) = ARRAY(3)
      GO TO 700
C
C     Stress tensor
C
 200  CONTINUE
C
      VECT(1) = ARRAY(3)
      VECT(2) = ARRAY(4)
      VECT(3) = ARRAY(5)
      VECT(4) = ARRAY(6)
      VECT(7) = ARRAY(6)
      IF(NDIM.LT.3) GO TO 700
      VECT(5) = ARRAY(7)
      VECT(8) = ARRAY(7)
      VECT(6) = ARRAY(8)
      VECT(9) = ARRAY(8)
      GO TO 700
C
C     Strain tensor
C
 300  CONTINUE
C
      VECT(1) = ARRAY(3)
      VECT(2) = ARRAY(4)
      VECT(3) = ARRAY(5)
      VECT(4) = ARRAY(6)*0.5
      VECT(7) = ARRAY(6)*0.5
      IF(NDIM.LT.3) GO TO 700
      VECT(5) = ARRAY(7)*0.5
      VECT(8) = ARRAY(7)*0.5
      VECT(6) = ARRAY(8)*0.5
      VECT(9) = ARRAY(8)*0.5
      GO TO 700
C
C     Node variable
C
 400  CONTINUE
C
      DO 500 J = 4,LR
             IC = J-3
             IF(IC.LT.9) GO TO 500
                KERROR = KERROR + 1
                WRITE(NOWC,2000)
                GO TO 700
 500  VECT(IC) = ARRAY(J)
C
 700  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FILLIN  * * *',//,
     &  10X,'ONLY KEY = 2,11,14,21,22,23,24,25 OR KEY GT. 100',
     &  ' ARE IMPLEMENTED OPTIONS',//,10X,'KEY = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FILLIN  * * *',//,
     &  10X,'FOR NODE VARIABLES ONLY 9 COMPCNENTS CAN BE READ ')
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   F I N D I E                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE FINDIE(NNLC1,NNLC2,NNLC3,NBEL,IELOC,NSID)
C
C     Find the elements to which nodes nnlc belong and on what side
C
C     Parameters
C           I/     NNLC1 NNLC2 NNLC3 : local numbers for 3 nodes
C           O/     NBEL    : number of elements to which all the three
C                            nodes belong (max 2)
C           O/     IELOC(2): local numb. for the elts to which nnlc belong
C           O/     NSID(2) : side of IELC to which nnlc belong (must be
C                            consec. nodes on a side - otherwise nsid=0)
C                            it is NSID = +/- 1 <-> +/- 4
C
      INCLUDE 'domain_common'
      DIMENSION NSLC(3),IELOC(2),NSID(2),NP(3)
C
      NBEL = 0
      DO 10 I = 1,2
      IELOC(I) = 0
 10   NSID(I) = 0
      NSLC(1) =NNLC1
      NSLC(2) =NNLC2
      NSLC(3) =NNLC3
C
      DO 200 IELC = 1, NTELT
C
        NTN = 0
        DO 50 N = 1,3
 50     NP(N) = 0
        DO 100 IP = 1,NNELT
        DO 100 NN = 1,3
C
          KP = (IP-1) *2 +3
          IF(IELTOP(KP,IELC).EQ.NSLC(NN)) THEN
            NP(NN) = IP
            NTN = NTN +1
          ENDIF
 100    CONTINUE
C
        IF (NTN.EQ.3) THEN
          NBEL = NBEL+1
          IELOC(NBEL) =IELC
          NP1 = NP(1)
          NP2 = NP(2)
          NP3 = NP(3)
          CALL SIDNOD(1,NP1,NP2,NP3,NSIDE)
          NSID(NBEL) = NSIDE
        ENDIF
 200  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   F I N D P N                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE FINDPN(NSIDE,IPONE,IPHALF,IPZERO)
C
C     Find the nodes IPHALF,IPZERO to be perturbed by the q field
C
C     Parameters
C           I/     NSIDE : side (-1/-4 +1/+4) to which IPONE belongs
C           I/     IPONE : node with perturbation = 1
C           O/     IPHALF: node with perturbation = 1/2
C           O/     IPZERO: node with perturbation = 0
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ================================================================
C
      INCLUDE 'domain_common'
C
      GO TO (100,200,300,400,500,600,700,800)IPONE
C
          KERROR = KERROR + 1
          WRITE(NOWC,1000) IPONE
          GO TO 900
C
 100  CONTINUE
      IF(ABS(NSIDE).EQ.1) GO TO 130
      IF(ABS(NSIDE).EQ.4) GO TO 160
C
          KERROR = KERROR + 1
          WRITE(NOWC,2000) IPONE,NSIDE
          GO TO 900
C
 130  CONTINUE
      IPHALF = 8
      IPZERO = 4
      GO TO 900
C
 160  CONTINUE
      IPHALF = 5
      IPZERO = 2
      GO TO 900
C
 200  CONTINUE
      IF(ABS(NSIDE).EQ.2) GO TO 230
      IF(ABS(NSIDE).EQ.1) GO TO 260
C
          KERROR = KERROR + 1
          WRITE(NOWC,2000) IPONE,NSIDE
          GO TO 900
C
 230  CONTINUE
      IPHALF = 5
      IPZERO = 1
      GO TO 900
C
 260  CONTINUE
      IPHALF = 6
      IPZERO = 3
      GO TO 900
C
 300  CONTINUE
      IF(ABS(NSIDE).EQ.3) GO TO 330
      IF(ABS(NSIDE).EQ.2) GO TO 360
C
          KERROR = KERROR + 1
          WRITE(NOWC,2000) IPONE,NSIDE
          GO TO 900
C
 330  CONTINUE
      IPHALF = 6
      IPZERO = 2
      GO TO 900
C
 360  CONTINUE
      IPHALF = 7
      IPZERO = 4
      GO TO 900
C
 400  CONTINUE
      IF(ABS(NSIDE).EQ.4) GO TO 430
      IF(ABS(NSIDE).EQ.3) GO TO 460
C
```

```
         KERROR = KERROR + 1
         WRITE(NOWC,2000) IPONE,NSIDE
         GO TO 900
C
 430  CONTINUE
      IPHALF = 7
      IPZERO = 3
      GO TO 900
C
 460  CONTINUE
      IPHALF = 8
      IPZERO = 1
      GO TO 900
C
 500  CONTINUE
      IF(ABS(NSIDE).EQ.1) GO TO 550
C
         KERROR = KERROR + 1
         WRITE(NOWC,2000) IPONE,NSIDE
         GO TO 900
C
 550  CONTINUE
      IPHALF = 0
      IPZERO = 7
      GO TO 900
C
 600  CONTINUE
      IF(ABS(NSIDE).EQ.2) GO TO 650
C
         KERROR = KERROR + 1
         WRITE(NOWC,2000) IPONE,NSIDE
         GO TO 900
C
 650  CONTINUE
      IPHALF = 0
      IPZERO = 8
      GO TO 900
C
 700  CONTINUE
      IF(ABS(NSIDE).EQ.3) GO TO 750
C
         KERROR = KERROR + 1
         WRITE(NOWC,2000) IPONE,NSIDE
         GO TO 900
C
```

```
 750  CONTINUE
      IPHALF = 0
      IPZERO = 5
      GO TO 900
C
 800  CONTINUE
      IF(ABS(NSIDE).EQ.4) GO TO 850
C
         KERROR = KERROR + 1
         WRITE(NOWC,2000) IPONE,NSIDE
         GO TO 900
C
 850  CONTINUE
      IPHALF = 0
      IPZERO = 6
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FINDPN  * * *',//,
     &  10X,'THE NODE NUMBER MUST BE BETW.1 AND 8',//,
     &  10X,'NODE NUMBER = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. FINDPN  * * *',//,
     &  10X,' NODE ',I4,' DOES NOT BELONG TO SIDE ',I4)
C
      RETURN
      END
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   G E O I N P                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE GEOINP
C
C     Manages the geometric input
C
      INCLUDE 'domain_common'
C
      DIMENSION IFISNO(10000),IDONE(10000)
C
      READ(NORI,1000) ITYPE,IRDINT,KPSTOP,IOUTG,IDOUBL
C
      IF(ITYPE.EQ.8) GO TO 10
      KERROR = KERROR +1
      WRITE(NOWC,2000)ITYPE
      GO TO 100
 10   CONTINUE
C
      NNELT = 8
      NNLELT = 3
      NDIM = 2
      NINTP = 9
      IF (IRDINT.EQ.1) NINTP = 4
C
C     Interface definition
C
      CALL ITFDEF(IFISNO)
C
      IF (KERROR.GT.0) GO TO 100
C
C
C     Connectivity matrices
C
      CALL CONNEC(IFISNO,IDONE)
C
      IF (KERROR.GT.0) GO TO 100
C
      CALL ITFCON
C
      IF (KERROR.GT.0) GO TO 100
C
C     Node cartesian components
C
      CALL NCARCO(IDONE)
C
C     If the nodes on the iterface are double, check if coords coincide
C
      IF (IDOUBL.EQ.2) CALL CHKITF
C
 100  CONTINUE
C
 1000 FORMAT(5(I5))
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR.GEOINP  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
      RETURN
      END
C
```

```
C
C
C     **************************************************************
C     *                                                            *
C     *                                                            *
C     *          S U B R O U T I N E   G E O I T F                 *
C     *                                                            *
C     *                                                            *
C     **************************************************************
C
C
      SUBROUTINE GEOITF
C
C     Evaluates curvilinear coordinate , normal and tangent vector to
C     the interface
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ==============================================================
C
      INCLUDE 'domain_common'
C
      DIMENSION N(3), X(2,3), T(2,3), AL(2), TOLD(2)
C
      TOLD(1) = 0.
      TOLD(2) = 0.
      SCURV(1) = 0.
C
C     Following the interface
C
      DO 500 NE = 1,NTLELT
            DO 100 I =1,3
                  NNIS = (NE-1)*2 + I
                  N(I) = NCONN(2,1,NNIS)
            DO 100 J =1,NDIM
 100        X(J,I) = COORDS(J,N(I))
C
C           Evaluating the cart. comp. of the tang. at the three nodes
C
            CALL TANG2D(X,T)
C
C           Average value of the tangent vector at node 1
C
            IF(NE.EQ.1) GO TO 200
              T(1,1) = (T(1,1)+TOLD(1))/2.
              T(2,1) = (T(2,1)+TOLD(2))/2.
 200        CONTINUE
C
            TOLD(1) = T(1,3)
            TOLD(2) = T(2,3)
C
C           Cartesian component of the normal at nodes 1,2,3
C
            DO 300 I =1,3
                  NNIS = (NE-1)*2+I
                  PATANG(1,NNIS) = T(1,I)
                  PATANG(2,NNIS) = T(2,I)
                  PATANG(3,NNIS) = 0.
                  PATNOR(1,NNIS) = -T(2,I)
                  PATNOR(2,NNIS) =  T(1,I)
                  PATNOR(3,NNIS) =  0.
 300        CONTINUE
C
C           Evaluating the arclength (1-2) and (2-3) : AL(2)
C
            CALL CCINC2(X,AL)
C
            IF (KERROR.GT.0) GO TO 600
C
C           Curvilinear coordinate
C
            DO 400 I =2,3
                  NNIS = (NE-1)*2 +I
                  SCURV(NNIS) = SCURV(NNIS-1) +AL(I-1)
 400        CONTINUE
 500  CONTINUE
 600  CONTINUE
C
      IF (IOUTG.EQ.0) GO TO 800
C
      WRITE(NOWG,1000)
C
      DO 700 NNIS = 1,NTOTIS
 700  WRITE(NOWG,2000) NNIS,SCURV(NNIS),(PATANG(I,NNIS),I = 1,3),
     &                 (PATNOR(I,NNIS),I = 1,3)
C
 800  Continue
C
 1000 Format(1h1,//,40x,'* * * I N T E R F.   G E O M E T R Y * * *',//
     &1X,'NNIS  CURV. COOR   T(1)   T(2)   T(3)   N(1)   N(2)   N(3)'//)
 2000 FORMAT(1X,I4,1X,F10.5,6(1X,F6.3))
```

```
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   G R A D C R                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE GRADCR(IELC)
C
C     Evaluates the gradient of the creep strain field CRESTR at
C     the integration points of the element IELC
C     CRGRAD(NC,I,J,IELC)= dCRESTR(nc)/dXi at i.p. J of elt IELC
C
C     Parameters
C           I/    IELC : Local elt number
C
      INCLUDE 'domain_common'
C
      DIMENSION  F(3,8),OU(9,9)
C
      DO 10 N = 1,3
      DO 10 I = 1,8
 10   F(N,I) = 0.
C
      DO 350 NLEV =1,3
C
        NSTART = (NLEV-1)*3
C
        DO 200 NN = 1,NNELT
C
            NLCP = (NN-1)*2+3
            NNLC = IELTOP(NLCP,IELC)
            DO 100 ND = 1,NDIM
 100        F(ND,NN) = CRESTR(ND+NSTART,NNLC)
C
 200    CONTINUE
C
        CALL GRADIE(IELC,1,F,OU)
        IF (KERROR.NE.0) GO TO 400
C
        DO 300 J = 1,9
            CRGRAD(NSTART+1,1,J,IELC) = OU(1,J)
            CRGRAD(NSTART+2,2,J,IELC) = OU(2,J)
            CRGRAD(NSTART+3,3,J,IELC) = OU(3,J)
            CRGRAD(NSTART+1,2,J,IELC) = OU(4,J)
            CRGRAD(NSTART+1,3,J,IELC) = OU(5,J)
            CRGRAD(NSTART+2,3,J,IELC) = OU(6,J)
            CRGRAD(NSTART+2,1,J,IELC) = OU(7,J)
            CRGRAD(NSTART+3,1,J,IELC) = OU(8,J)
            CRGRAD(NSTART+3,2,J,IELC) = OU(9,J)
  300     CONTINUE
  350   CONTINUE
C
        ICGRAD(IELC) = 1
C
  400   CONTINUE
C
        RETURN
        END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *            S U B R O U T I N E   G R A D I E                 *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE GRADIE(IELC,IDIM,F,OU)
C
C     Evaluates the gradient of a scalar field or of a vector field F at
C     the integration points of the element IELC
C
C     Parameters
C           I/    IDIM : Dimension flag of the input field
C                                                       0 : scalar field
C                                                       1 : vector field
C           I/    F    : Input field at all nodes of the elt
C           O/    OU   : Output gradient field at the int.points
C                        OU(I,J) =  comp.I at int.pt. J
C
      INCLUDE 'domain_common'
C
      DIMENSION  F(3,8),OU(9,9),GR(3,3)
      DIMENSION AM(3,3),AU(3,3),AL(3,3),B(3),X(3), INEW(3)
C
C     Local array :
C            AM : coeff. matrix -> AM(I,J) = [d N(k)/d c(I)] * X(k)(J)
C
C            AL,AU : Triang. fact. matr. AM = AL*AU
C
C            B : RHS -> Bj(I) = [d N(k)/d c(I)] * Fj(k)
C
C            X : Unknown vect -> Xj(I) = d Fj/d x(I)
C
C            GR : Gradient at the int. point  ->
C                                      GR(J,I) = d Fj/ d x(I)
C                                               C
C                          N(k)    : Kth shape function
C                          c(I)    : Ith isop. coord.
C                          X(k)(J): Jth cart coord of the Kth node
C                          x(I)    : Ith cartesian coordinate
C                          Fj(k)   :Jth comp. of input field at node k
```

```
C
C
      DO 10 N = 1,9
      DO 10 I = 1,9
 10   OU(I,N) = 0.
C
      DO 20 I = 1,3
      DO 20 J = 1,3
 20   GR(I,J) = 0.
C
      NCOMP = NDIM
      IF(IDIM.EQ.0) NCOMP = 1
C
C
      DO 400 IP = 1,NINTP
C
C        Evaluate the coefficient matrix AM
C
         CALL GRAMAT(IELC,IP,AM)
C
C        LU factorization
C
         KER = KERROR
         CALL LUFACT(KER,3,IMOVE,INEW,AM,AU,AL)
         KERROR = KER
         IF(KERROR.GT.0) GO TO 500
C
C        Solving the system for all the components of F
C
         DO 100 ID = 1,NCOMP
C
C           Evaluating the RHS
C
            CALL GRARHS (IELC,IP,F,ID,B)
C
C           solving the system
C
            CALL SOLVER (3,IMOVE,INEW,AU,AL,B,X)
C
C        Storing the gradient
C
         DO 100 J = 1,NDIM
C
 100     GR(ID,J) = X(J)
C
C        Storing the result in the output array
C
         IF (IDIM.EQ.1) GO TO 300
C
C        F is a scalar field -> OU is a vector with NDIM components
C
         OU(1,IP) = GR(1,1)
         OU(2,IP) = GR(1,2)
         OU(3,IP) = GR(1,3)
C
      GO TO 400
C
C        F is a vector field -> OU is a tensor with NDIM*NDIM comp's
C
 300     CONTINUE
C
         OU(1,IP) = GR(1,1)
         OU(2,IP) = GR(2,2)
         OU(3,IP) = GR(3,3)
         OU(4,IP) = GR(1,2)
         OU(5,IP) = GR(1,3)
         OU(6,IP) = GR(2,3)
         OU(7,IP) = GR(2,1)
         OU(8,IP) = GR(3,1)
         OU(9,IP) = GR(3,2)
C
 400  CONTINUE
 500  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   G R A D I Q                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE GRADIQ(IFLAG,NNIS,IELC,IPONE,IPHALF,IPZERO,QGRAD)
C
C     Evaluates the gradient of the perturbation  field Q at
C     the integration points of the element IELC
C
C     Parameters
C       I/  IFLAG : perturbation flag
C       I/  IPONE
C           IPHALF : nodes to be perturbed
C           IPZERO
C
C                    IFLAG = 0 ->partial perturbation Q  =1   on IPONE
C                                                        =1/2 on IPHALF
C                                                        =0   on IPZERO
C                    IFLAG = 1 ->Total perturbation Q  =1   on IPONE
C                                                      =1   on IPHALF
C                                                      =1   on IPZERO
C       I/    NNIS : ref. intf. node : Q = unit normal to intf at NNIS
C       I/    IELC : Local elt number
C       O/    QGRAD: Gradient of Q at the integration points
C
C              QGRAD(1,J) :  dU1/dX1 at integration point J
C              QGRAD(2,J) :  dU2/dX2 at integration point J
C              QGRAD(3,J) :  dU3/dX3 at integration point J
C              QGRAD(4,J) :  dU1/dX2 at integration point J
C              QGRAD(5,J) :  dU1/dX3 at integration point J
C              QGRAD(6,J) :  dU2/dX3 at integration point J
C              QGRAD(7,J) :  dU2/dX1 at integration point J
C              QGRAD(8,J) :  dU3/dX1 at integration point J
C              QGRAD(9,J) :  dU3/dX2 at integration point J
C
C
      INCLUDE 'domain_common'
C
      DIMENSION  F(3,8),QGRAD(9,9)
C
      DO 10 N = 1,3
      DO 10 I = 1,8
 10   F(N,I) = 0.
C
      DO 100 ND = 1,NDIM
         F(ND,IPONE) = PATNOR(ND,NNIS)
C
         IF (IFLAG.EQ.1) F(ND,IPZERO) =PATNOR(ND,NNIS)
         IF (IPHALF.EQ.0) GO TO 100
         F(ND,IPHALF) = 0.5 * PATNOR(ND,NNIS)
         IF (IFLAG.EQ.1) F(ND,IPHALF) =PATNOR(ND,NNIS)
 100  CONTINUE
C
      CALL GRADIE(IELC,1,F,QGRAD)
C
      RETURN
      END
C
```

```
C
C
C     *******************************************************************
C     *                                                                 *
C     *                                                                 *
C     *           S U B R O U T I N E   G R A D I U                     *
C     *                                                                 *
C     *                                                                 *
C     *******************************************************************
C
      SUBROUTINE GRADIU(IELC)
C
C     Evaluates the gradient of the displacement field U at
C     the integration points of the element IELC
C
C     Parameters
C           I/    IELC : Local elt number
C
      INCLUDE 'domain_common'
C
      DIMENSION  F(3,8),OU(9,9)
C
      DO 10 N = 1,3
      DO 10 I = 1,8
 10   F(N,I) = 0.
C
      DO 200 NN = 1,NNELT
C
         NLCP = (NN-1)*2+3
         NNLC = IELTOP(NLCP,IELC)
         DO 100 ND = 1,NDIM
 100     F(ND,NN) = UNODE(ND,NNLC)
C
 200  CONTINUE
C
      CALL GRADIE(IELC,1,F,OU)
      IF (KERROR.NE.0) GO TO 400
C
      DO 300 I = 1,9
      DO 300 J = 1,9
 300  UGRAD(I,J,IELC) = OU(I,J)
C
      IUGRAD(IELC) = 1
C
 400  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   G R A M A T                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE GRAMAT(IELC,IP,AM)
C
C     Evaluate the coefficient matrix AM for integration point IP
C     of element IELC
C
C     Paramweters
C          I/      IELC : local elt id number
C          I/      IP   : position of the integr.point in the elt
C          O/      AM   : coeff. matrix ->
C                                    AM(I,J) = [d N(k)/d c(I)] * X(k)(J)
C
C                        N(k)    : Kth shape function
C                        c(I)    : Ith isop. coord.
C                        X(k)(J): Jth cart coord of the Kth integr.point
C
      INCLUDE 'domain_common'
      DIMENSION AM(3,3)
C
      DO 20 NR = 1,3
      DO 10 NC = 1,3
 10   AM(NR,NC) = 0.
 20   AM(NR,NR) = 1.
C
      DO 200 NR = 1,NDIM
      DO 200 NC = 1,NDIM
             SUM = 0.
             DO 100 K = 1, NNELT
                    INLC = 3 +(K-1)*2
                    NNLC = IELTOP(INLC,IELC)
 100         SUM = SUM+ SFDITP(IP,K,NR) * COORDS(NC,NNLC)
 200  AM(NR,NC) = SUM
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   G R A R H S                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE GRARHS(IELC,IP,F,ID,B)
C
C     Evaluates the Right Hand Side of the system to be solved in order
C     to find the gradient of the IDth component of the field F
C
C     Parameters
C          I/      IELC : local elt id number
C          I/      IP   : position of the int. point in the elt
C          I/      F    : input field at each rode
C          I/      ID   : component of F to be considered
C          O/      B    : RHS -> B(I) = [d N(k)/d c(I)] * F(ID)(k)
C
C                        N(k)     : Kth shape function
C                        c(I)     : Ith isop. coord.
C                        F(ID)(k):IDth comp. of input field at int. point k
C
      INCLUDE 'domain_common'
      DIMENSION F(3,8),B(3)
C
      DO 10 I = 1,3
 10   B(I) = 0.
C
      DO 200 J = 1,NDIM
             SUM = 0.
             DO 100 K = 1,NNELT
 100         SUM = SUM + SFDITP(IP,K,J) * F(ID,K)
 200  B(J) = SUM
C
      RETURN
      END
C
```

```
C
C
C     ***************************************************************
C     *                                                             *
C     *                                                             *
C     *          S U B R O U T I N E   I D T C A L                  *
C     *                                                             *
C     *                                                             *
C     ***************************************************************
C
      SUBROUTINE IDTCAL(NNIS,ISIDE,LAYMAX,IDOTOP)
C
C     Evaluates the topology matrix IDOTOP for the domain integral
C
C     Parameters
C          I/  LAYMAX: farthest layer from the interface for which
C                      IDOTOP must be evaluated
C          I/  NNIS  : interface number for which IDOTOP must be eval.
C          I/  ISIDE : side of the intf. for which IDOTOP must be eval.
C          O/  IDOTOP: Output topology matrix
C                       IDOTOP(I,J) =  topology of layer J
C                       the component of sach row are:
C      I =   1      2      3       4       5      6     7      8      9
C         NELLAY,IELC1,IPONE1,IPHALF1,IPZERO1,IELC2,IPONE2,IPHALF2,IPZERO2
C
C
      INCLUDE 'domain_common'
      DIMENSION IDOTOP(9,15),IELOC(2),NSID(2)
C
      DO 10 N = 1,9
      DO 10 I = 1,15
   10 IDOTOP(N,I) = 0
C
      NELT = LNOCON(1,NNIS)
C
      DO 700 LE = 1,NELT
C
         LEP = (LE-1)*2+2
         LELT = LNOCON(LEP  ,NNIS)
         IPL  = LNOCON(LEP+1,NNIS)
C
         NNLC = LELCON(IPL,ISIDE,LELT)
         IELC = LELCON( 4 ,ISIDE,LELT)
         NSIDE= LELCON( 5 ,ISIDE,LELT)
C
      DO 100 NN = 1,NNELT
         KP = (NN-1)*2+3
  100 IF(IELTOP(KP,IELC).EQ.NNLC) IPONE = NN
C
      CALL FINDPN(NSIDE,IPONE,IPHALF,IPZERO)
      IF (KERROR.GT.0) GO TO 900
C
      IDOTOP(1,1) = IDOTOP(1.1) +1
        KP = (IDOTOP(1,1)-1)*4 +2
      IDOTOP(KP  ,1) = IELC
      IDOTOP(KP+1,1) = IPONE
      IDOTOP(KP+2,1) = IPHALF
      IDOTOP(KP+3,1) = IPZERO
C
      DO 600 NLAY = 2,LAYMAX
C
         NLCZP = (IPZERO-1)*2 +3
         NNLCZ = IELTOP(NLCZP,IELC)
C
         NSOP = NSIDOP(NSIDE)
         IF (KERROR.GT.0) GO TO 900
C
         CALL SIDNOD(0,IP1,IP2,IP3,NSOP)
         IF (KERROR.GT.0) GO TO 900
C
         N1P = (IP1-1)*2+3
         NNLC1 = IELTOP(N1P,IELC)
         N2P = (IP2-1)*2+3
         NNLC2 = IELTOP(N2P,IELC)
         N3P = (IP3-1)*2+3
         NNLC3 = IELTOP(N3P,IELC)
C
         CALL FINDIE(NNLC1,NNLC2,NNLC3,NBEL,IELOC,NSID)
C
         IF (NBEL.GT.1) GO TO 200
C
            KERROR = KERROR +1
            WRITE(NOWC,1000) NNLC1,NNLC2,NNLC3,IELC
            GO TO 900
C
  200    CONTINUE
C
         DO 3C0 NB = 1, NBEL
C
           IF(IELOC(NB).EQ.IELC) GO TO 300
```

```
            IELC = IELOC(NB)
            NSIDE = NSID(NB)
            IF(NSIDE.NE.0) GO TO 400
            KERROR = KERROR +1
            WRITE(NOWC,2000) NNLC1,NNLC2,NNLC3,IELC
 300      CONTINUE
 400      CONTINUE
C
          DO 500 NN = 1,NNELT
            KP = (NN-1)*2+3
 500      IF(IELTOP(KP,IELC).EQ.NNLCZ) IPONE = NN
C
          CALL FINDPN(NSIDE,IPONE,IPHALF,IPZERO)
          IF (KERROR.GT.0) GO TO 900
C
          IDOTOP(1,NLAY) = IDOTOP(1,NLAY) +1
            KP = (IDOTOP(1,NLAY)-1)*4 +2
          IDOTOP(KP  ,NLAY) = IELC
          IDOTOP(KP+1,NLAY) = IPONE
          IDOTOP(KP+2,NLAY) = IPHALF
          IDOTOP(KP+3,NLAY) = IPZERO
C
 600        CONTINUE
 700    CONTINUE
C
 900    CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR.IDTCAL  * * *',//,
     &  10X,'CANNOT FIND ANOTHER ELEMENT TO WHICH NODES',3(2X,I4),
     &  1X,'BELONG. START FROM ELT NUMBER = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR.IDTCAL  * * *',//,
     &  10X,' NODES',3(2X,I4),'BELONG TO ELT ',I4,'BUT THEY DO NOT',
     &  10X,'BELONG. TO THE SAME SIDE ')
C
        RETURN
        END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   I N T C A L                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE INTCAL(NNIS,ISIDE,LAYMAX,LAYREQ,IDOTOP,PARINT,TOTINT)
C
C     Evaluates the domain integral PARINT & TOTINT
C
C     Parameters
C        I/  NNIS  : interface  node number
C        I/  ISIDE : side of the intface
C        I/  LAYMAX: farthest layer from the interface for which
C                       domain integral must be evaluated
C        I/  LAYREQ: layers for which PARINT must be evaluated
C        I/  IDOTOP:  topology matrix
C                        IDOTOP(I,J) =  topology of layer J
C        O/  PARINT:  domain integral [PARINT(J)=intg on elts of layer J]
C                     partial: with perturbation Q  =1   on IPONE
C                                                   =1/2 cn IPHALF
C                                                   =0   on IPZERO
C        O/  TOTINT:  domain integral [TOTINT(J)=intg on elts of layer J]
C                     total  : with perturbation Q  =1   on IPONE
C                                                   =1   on IPHALF
C                                                   =1   on IPZERO
      INCLUDE 'domain_common'
      DIMENSION IDOTOP(9,15),LAYREQ(15),PARINT(15),TOTINT(15)
      DIMENSION QGRAD(9,9),QVEC(3,9)
C
C
      DO 10 I = 1,15
      TOTINT(I) = 0.
   10 PARINT(I) = 0.
C
      DO 20 I=1,9
      DO 20 J=1,3
   20 QVEC(J,I) = 0.
C
      DO 300 NLAY = 1,LAYMAX
C
```

```
      NELLAY = IDOTOP(1,NLAY)
      SUMP = 0.
      SUMT = 0.
      DO 200 NE = 1,NELLAY
C
         KP = (NE-1)*4 +2
         IELC = IDOTOP(KP   ,NLAY)
         NONE = IDOTUP(KP+1,NLAY)
         NHALF= IDOTOP(KP+2,NLAY)
         NZERO= IDOTOP(KP+3,NLAY)
C
         IF(IUGRAD(IELC).NE.1) CALL GRADIU(IELC)
         IF(KERROR.NE.0) GO TO 900
C
         CALL GRADIQ(1,NNIS,IELC,NONE,NHALF,NZERO,QGRAD)
         IF(KERROR.NE.0) GO TO 900
C
         IF (ICREEP.EQ.1) THEN
C
            CALL QVCALC(1,NNIS,IELC,NONE,NHALF,NZERO,QVEC)
            IF(KERROR.NE.0) GO TO 900
C
            IF(ICGRAD(IELC).NE.1) CALL GRADCR(IELC)
            IF(KERROR.NE.0) GO TO 900
C
         ENDIF
C
         CALL INTEGR(IELC,QGRAD,QVEC,TINTG)
         IF(KERROR.NE.0) GO TO 900
C
         SUMT = SUMT +TINTG
C
         IF (LAYREQ(NLAY).EQ.1) THEN
C
            CALL GRADIQ(0,NNIS,IELC,NONE,NHALF,NZERO,QGRAD)
            IF(KERROR.NE.0) GO TO 900
C
            IF (ICREEP.EQ.1) THEN
C
               CALL QVCALC(0,NNIS,IELC,NONE,NHALF,NZERO,QVEC)
               IF(KERROR.NE.0) GO TO 900
C
            ENDIF
C
C
               CALL INTEGR(IELC,QGRAD,QVEC,PINTG)
               IF(KERROR.NE.0) GO TO 900
C
               SUMP = SUMP +PINTG
            ENDIF
  200    CONTINUE
C
         PARINT(NLAY) = SUMP
         TOTINT(NLAY) = SUMT
  300 CONTINUE
C
  900 CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E   I N T E G R                  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE INTEGR(IELC,QGRAD,QVEC,AINTG)
C
C     Find the integral over the element IELC of
C
C                W*Qi,i + SIGkj*Uk,i*Qi,j
C
C     by means of Gauss quadrature
C
C     Parameters
C        I/    IELC  : Local numb. of elt
C        I/    QGRAD : Gradient of the perturbation field
C        O/    AINTG : Integral over IELC of F=W*Qi,i - SIGkj*Uk,i*Qi,j
C                      = sum|ip of F[ip]*Jac[ip]*Weight[ip]
C
      INCLUDE 'domain_common'
      DIMENSION F(9), QGR(3,3), UGR(3,3), SIG(3,3),CGR(3,3,3)
      DIMENSION QGRAD(9,9),QVEC(3,9),AJACOB(9)
C
      AINTG = 0.
C
      CALL JACALC(IELC,AJACOB)
C
      DO 400 IP = 1,NINTP
C
         QGR(1,1) = QGRAD(1,IP)
         QGR(2,2) = QGRAD(2,IP)
         QGR(3,3) = QGRAD(3,IP)
         QGR(1,2) = QGRAD(4,IP)
         QGR(1,3) = QGRAD(5,IP)
         QGR(2,3) = QGRAD(6,IP)
         QGR(2,1) = QGRAD(7,IP)
         QGR(3,1) = QGRAD(8,IP)
         QGR(3,2) = QGRAD(9,IP)
C
         UGR(1,1) = UGRAD(1,IP,IELC)
         UGR(2,2) = UGRAD(2,IP,IELC)
         UGR(3,3) = UGRAD(3,IP,IELC)
         UGR(1,2) = UGRAD(4,IP,IELC)
         UGR(1,3) = UGRAD(5,IP,IELC)
         UGR(2,3) = UGRAD(6,IP,IELC)
         UGR(2,1) = UGRAD(7,IP,IELC)
         UGR(3,1) = UGRAD(8,IP,IELC)
         UGR(3,2) = UGRAD(9,IP,IELC)
C
         SIG(1,1) = SIGMIP(1,IP,IELC)
         SIG(2,2) = SIGMIP(2,IP,IELC)
         SIG(3,3) = SIGMIP(3,IP,IELC)
         SIG(1,2) = SIGMIP(4,IP,IELC)
         SIG(1,3) = SIGMIP(5,IP,IELC)
         SIG(2,3) = SIGMIP(6,IP,IELC)
         SIG(2,1) = SIGMIP(7,IP,IELC)
         SIG(3,1) = SIGMIP(8,IP,IELC)
         SIG(3,2) = SIGMIP(9,IP,IELC)
C
         DIVQ = 0.
         DO 100 ND = 1,NDIM
  100    DIVQ = DIVQ + QGR(ND,ND)
C
         SUM = 0.
         DO 300 J = 1,NDIM
         DO 300 I = 1,NDIM
            AMULT = 0.
            DO 200 K =1,NDIM
  200       AMULT = AMULT + SIG(K,J)*UGR(K,I)
  300    SUM = SUM +AMULT*QGR(I,J)
C
         F(IP) = WENIP(IP,IELC)*DIVQ - SUM
C
         IF(ICREEP.EQ.1) THEN
C
               DO 310 J=1,NDIM
C
                  CGR(1,1,J) = CRGRAD(1,J,IP,IELC)
                  CGR(2,2,J) = CRGRAD(2,J,IP,IELC)
                  CGR(3,3,J) = CRGRAD(3,J,IP,IELC)
                  CGR(1,2,J) = CRGRAD(4,J,IP,IELC)
                  CGR(1,3,J) = CRGRAD(5,J,IP,IELC)
                  CGR(2,3,J) = CRGRAD(6,J,IP,IELC)
                  CGR(2,1,J) = CRGRAD(7,J,IP,IELC)
                  CGR(3,1,J) = CRGRAD(8,J,IP,IELC)
```

```
                    CGR(3,2,J) = CRGRAD(9,J,IP,IELC)
C
 310             CONTINUE
C
                 SUM = 0.
                 DO 350 M=1,NDIM
                 DO 350 N=1,NDIM
                 DO 350 I=1,NDIM
C
 350             SUM = SUM+SIG(M,N)*CGR(M,N,I)*QVEC(I,IP)
C
                 F(IP) = F(IP)-SUM
C
            ENDIF

 400  AINTG = AINTG + WEIGHT(IP)*F(IP)*AJACOB(IP)
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   I T F C O N                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C

      SUBROUTINE ITFCON
C
C
C     Evaluates the connectivity matrices  for itf. elts LELCON & LNOCON
C     Evaluates the total number of linear  element NTLELT
C
      INCLUDE 'domain_common'
      DIMENSION IELC(2),NSIDE(2)
C
      NTLELT = (NTOTIS-1)/2
C
      DO 300 LELT = 1,NTLELT
C
      DO 100 NN = 1,NNLELT
C
         NNIS = (LELT-1)*2+NN
         LNOCON (1,NNIS)=LNOCON (1,NNIS)+1
         K =  (LNOCON (1,NNIS)-1)*2 +2
         LNOCON (K,NNIS)=LELT
         LNOCON (K+1,NNIS)=NN
C
      DO 100 IS = 1,2
         ISIDE = IS
         IF (IDOUBL.EQ.1) ISIDE = 1
 100  LELCON (NN,IS,LELT) = NCONN (2,ISIDE,NNIS)
C
      DO 200 IS = 1,2
C
      CALL FINDIE(LELCON(1,IS,LELT),LELCON(2,IS,LELT),LELCON(3,IS,LELT),
     &                NBEL,IELC,NSIDE)
      DO 200 NBE = 1,NBEL
      IF((IS.EQ.1.AND.NSIDE(NBE).LT.0).OR.(IS.EQ.2.AND.NSIDE(NBE).GT.0))
     &   THEN
         LELCON (4,IS,LELT)=IELC(NBE)
```

```
          LELCON (5,IS,LELT)=NSIDE(NBE)
        ENDIF
 200  CONTINUE
 300  CONTINUE
C
C
      IF (IOUTG.EQ.0) GO TO 600
        WRITE(NOWG,1000)
        DO 400 LELT = 1,NTLELT
 400    WRITE(NOWG,2000) ((LELCON(I,J,LELT),I=1,5),J=1,2)
C
        WRITE(NOWG,3000)
        DO 500 NNIS = 1, NTOTIS
 500    WRITE(NOWG,4000) (LNOCON(I,NNIS),I = 1,5)
C
 600  CONTINUE
C
 1000 FORMAT(1H1,///,40X,'* * * L E L C O N * * *',//,
     &  14X,'ISIDE = 1',28X,'ISIDE = 2 ',//,
     &  1X,'NNLC1  NNLC2  NNLC3   IELC  NSIDE    NNLC1  NNLC2  NNLC3',
     &     '   IELC  NSIDE',//)
 2000 FORMAT(5(1X,I5,1X),2X,5 (1X,I5,1X))
 3000 FORMAT(1H1,///,40X,'* * * L N O C O N * * *',//,
     &  1X,' NELT  LELT1   IP1   LELT2   IP2',//)
 4000 FORMAT(5(1X,I5,1X))
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           S U B R O U T I N E   I T F D E F                   *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE ITFDEF(IFISNO)
C
C     Reads input set nodes. Find total number of input set nodes
C     Fills NCONN(1,ISIDE,NNIS)
C
C     Parameters
C         O/      IFISNO(10000) :NNIS input-set-number for abaqus nodes
C                                if >0 -> node on outer interface
C                                if <0 -> node on the inner interface
C
      INCLUDE 'domain_common'
      DIMENSION IFISNO(10000),NINP(15)
C
      ISIGN = 1
      DO 300 ISIDE = 1,IDOUBL
         ISIGN = -ISIGN
         NNIS = 0
         READ(NORI,1000) NSET
         DO 100 NS = 1,NSET
            READ(NORI,2000) IN,(NINP(K),K=1,15)
            IF(IN.NE.0)THEN
              NF = NINP(1)
              NL = NINP(2)
              NT = 1+(NL-NF)/IN
              DO 30 NN = 1,NT
                    NNIS = NNIS+1
                    NNAB = NF + (NN-1)*IN
                    NCONN(1,ISIDE,NNIS) = NNAB
                    IFISNO(NNAB) = NNIS*ISIGN
 30           CONTINUE
            ELSE
              DO 60 NN=1,15
                    IF(NINP(NN).EQ.0) GO TO 60
                    NNAB = NINP(NN)
                    NNIS = NNIS +1
```

```
                NCONN(1,ISIDE,NNIS) = NNAB
                IFISNO(NNAB) = NNIS*ISIGN
 60          CONTINUE
           ENDIF
 100    CONTINUE
        NTOTIS = NNIS
C
C       Check the max number of nodes
C
        IF (NTOTIS.LE.200) GO TO 200
          KERROR = KERROR +1
          WRITE(NOWC,3000) ISIDE,NTOTIS
 200    CONTINUE
 300  CONTINUE
C
 1000 FORMAT(I5)
 2000 FORMAT(16(I5))
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. ITFDEF  * * *',//,
     &  10X,'MAX NUMBER OF INPUT SET NODES  EXCEEDED ON SIDE ',I2,//,
     &  10X,'TOTAL NUMBER OF I.S. NODES = ',I4,' (MAX: 200)')
C
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *          S U B R O U T I N E   J A C A L C                   *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE JACALC(IELC,AJACOB)
C
C     Evaluate the Jacobian at the integration points  of element IELC
C
C     Paramweters
C          I/      IELC : local elt id number
C          O/      AJACOB: Jacobian:
C                           AJACOB(I) = jacobian at int. pt. I
C
C                   AJACOB= DX/DG*DY/DH-DY/DG*DX/DH
C
      INCLUDE 'domain_common'
      DIMENSION AJACOB(9)
C
      DO 10 NC = 1,9
 10   AJACOB(NC) = 0.
C
      DO 200 IP = 1,NINTP
C
         DXDG = 0.
         DXDH = 0.
         DYDG = 0.
         DYDH = 0.
         DO 100 K = 1, NNELT
C
            INLC = 3 +(K-1)*2
            NNLC = IELTOP(INLC,IELC)
C
            DXDG = DXDG +SFDITP(IP,K,1)*COORDS(1,NNLC)
            DXDH = DXDH +SFDITP(IP,K,2)*COORDS(1,NNLC)
            DYDG = DYDG +SFDITP(IP,K,1)*COORDS(2,NNLC)
            DYDH = DYDH +SFDITP(IP,K,2)*COORDS(2,NNLC)
 100     CONTINUE
         AJACOB(IP) = DXDG*DYDH-DYDG*DXDH
 200  CONTINUE
```

```
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   L U F A C T                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE LUFACT(KER,ID,IMOVE,INEW,AM,AU,AL)
C
C     Factorize a square matrix AM(ID,ID) into a lower triang. matrix
C     AL(ID,ID) and an upper triang. matrix AU(ID,ID) . AM = AL*AU
C     If elements on the main diagonal of AM are equal to zero,
C     rows of AM are permuted and a track is kept in array INEW.
C     Then (AM)perm. is factorized.
C     Parameters
C           I/O    KER : error count.
C           I/     ID : matrix dimension
C           O/     IMOVE : row permutation code (=1 : row permuted )
C           O/     INEW : perm. record :INEW(J)= old # of new row #J
C           I/     AM : square input matrix
C           O/     AL : lower triang matrix
C           O/     AU : upper triang.matrix
C
      INCLUDE 'domain_common'
C
      DIMENSION AM(ID,ID),AU(ID,ID),AL(ID,ID),INEW(ID)
C
C     Check if matrix AM has zero elements on the main diagonal
C     and permutation of rows if it is needed.
C
      CALL RWPERM (ID,IMOVE,INEW,AM,AU,AL)
C
C     Check if AM has any zero pivot
C
      IF(IMOVE.GE.C) GO TO 5
        KER = KER+1
        GO TO 200
 5    CONTINUE
C
      PIVMIN = 1.D-10
C
      DO 20 NC = 1,ID
      DO 10 NR = 1,ID
       AU(NR,NC) = AM(NR,NC)
 10    AL(NR,NC) = 0.
 20    AL(NC,NC) = 1.
C
      DO 100 NFR = 1,ID-1
            PIVOT = AU(NFR,NFR)
            IF(DABS(PIVOT).GT.PIVMIN) GO TO 50
              KER = KER +1
              WRITE (13,1000) NFR,PIVOT
              WRITE (13,*) AM
              IF(IMOVE.GT.0) WRITE(13,2000)
              IF(IMOVE.GT.0) WRITE(13,*) INEW
              GO TO 200
 50         CONTINUE
       DO 100 NSR = NFR+1,ID
             AL(NSR,NFR) = AU(NSR,NFR)/PIVOT
       DO 100 NC = NFR,ID
 100  AU(NSR,NC) = AU(NSR,NC)-AU(NFR,NC)*AL(NSR,NFR)
C
 200  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. LUFACT  * * *',//,
     &  10X,'ROW ',I4,' PIVOT TOO SMALL!!!! PIVOT = ',E12.5,//,
     &  10X,'INPUT MATRIX :',//)
 2000 FORMAT(///,20X,'* * * N O T E ! ! ! * * *',//,
     &  10X,'ROWS HAVE BEEN PERMUTED . PERMUTATION RECORD VECTOR ',
     &  10X,'INEW FOLLOWS: (INEW(J) = OLD # OF ACTUAL ROW # J)',/)
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *            S U B R O U T I N E   M A T R I X                  *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE MATRIX
C
C     Evaluate the coefficient matrix AMAT(200,200)
C
      INCLUDE 'domain_common'
C
      DO 50 I = 1,200
      DO 50 J = 1,200
 50   AMAT(I,J) = 0.
C
      DO 70 J=NTOTIS+1,200
 70   AMAT(J,J) = 1.
C
      DO 400 NNIS = 1,NTOTIS
C
         NELT = LNOCON(1,NNIS)
         DO 300 LE = 1,NELT
C
            LEP = (LE-1)*2 +2
            LELT = LNOCON(LEP,NNIS)
            IP   = LNOCON(LEP+1,NNIS)
            DO 200 NN=1,NNLELT
C
               NCOL = (LELT-1)*2+NN
               SUM = 0.
               DO 100 NS = 1,NNLELT
C
                  NIS = (LELT-1)*2 +NS
                  SC = SCURV(NIS)
C
 100           SUM = SUM +BQUAD(IP,NN,NS)*SC
C
 200        AMAT(NNIS,NCOL) = AMAT(NNIS,NCOL)+SUM
C
 300     CONTINUE
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          S U B R O U T I N E   N C A R C O                     *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      SUBROUTINE NCARCO(IDONE)
C
C     Reads the cartesian coordinates of the nodes
C
C     Parameters
C         I/      IDONE (10000) :local number for abaqus nodes
C
      INCLUDE 'domain_common'
C
      DOUBLE PRECISION ARRAY
      DIMENSION ARRAY(513),JRRAY(2,513),IDONE (10000)
      EQUIVALENCE (ARRAY(1), JRRAY(1,1))
C
C     Rewind file 8
C
      CALL DBFILE(2,ARRAY,JRCD)
C
C     Scanning file 8
C
      NCHECK = 0
      DO 200 K = 1,99999
            CALL DBFILE(0,ARRAY,JRCD)
            IF (JRCD.NE.0) GO TO 300
            LR = JRRAY(1,1)
            KEY = JRRAY(1,2)
            IF(KEY.NE.1901) GO TO 200
            NNAB = JRRAY(1,3)
            NNLC = IDONE(NNAB)
            IF(NNLC.EQ.0) GO TO 200
            NCHECK = NCHECK + 1
            DO 100 ND = 1, NDIM
 100        COORDS(ND,NNLC) = ARRAY(3+ND)
 200  CONTINUE
 300  CONTINUE
C
 400  CONTINUE
C
      RETURN
      END
C
```

```
C
C     Check if all the nodes have been found
C
      IF(NCHECK.EQ.NTNOD) GO TO 400
        KERROR = KERROR +1
        WRITE(NOWC,1000) NCHECK,NTNOD
 400  CONTINUE
C
      IF(IOUTG.EQ.0) GO TO 600
C
      WRITE(NOWG,2000)
C
      DO 500 NNLC = 1,NTNOD
 500  WRITE(NOWG,3000) NNLC,NABAQ(NNLC),(COORDS(I,NNLC),I =1,3)
C
 600  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. NCARCO  * * *',//,
     &  10X,'TOTAL NUMBER OF READ NODES :',I4,'IS NOT CONSISTENT WITH ',
     &  'THE TOTAL NUMBER OF NODES : ',I4)
 2000 FORMAT(1H1,////,40X,'* * *   C O O R D I N A T E S * * *',//
     &  'NNLC NNAB    X1         X2         X3',/)
 3000 FORMAT (2(1X,I4),3(2X,F6.3,2X))
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *          F U N C T I O N    N S I D O P                        *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      FUNCTION NSIDOP(NSIDE)
C
C     Gives the identity of the element side opposite to nside
C
C     Parameters
C              NSIDE : input side of elt
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     =================================================================
C
      NSIDOP = 0
C
      NS = NSIDE +5
C
      GO TO (100,200,300,400,500,600,700,800,900) NS
C
        KERROR = KERROR + 1
        WRITE(NOWC,1000) NSIDE
        GO TO 990
C
 100  CONTINUE
      NSIDOP = 2
      GO TO 990
 200  CONTINUE
      NSIDOP = 1
      GO TO 990
 300  CONTINUE
      NSIDOP = 4
      GO TO 990
 400  CONTINUE
      NSIDOP = 3
      GO TO 990
 500  CONTINUE
      KERROR = KERROR + 1
      WRITE(NOWC,1000) NSIDE
      GO TO 990
  600 CONTINUE
      NSIDOP = -3
      GO TO 990
  700 CONTINUE
      NSIDOP = -4
      GO TO 990
  800 CONTINUE
      NSIDOP = -1
      GO TO 990
  900 CONTINUE
      NSIDOP = -2
C
  990 CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.NSIDOP  * * *'.//,
     &  10X,'THE SIDE NUMBER MUST BE BETW.-4 AND -1 OR 1 AND 4',//,
     &  10X,'SIDE NUMBER = ',I4)
C
      RETURN
      END
```

```
C
C
C     *************************************************************
C     *                                                           *
C     *                                                           *
C     *          S U B R O U T I N E   O U T P U T                *
C     *                                                           *
C     *                                                           *
C     *************************************************************
C
      SUBROUTINE OUTPUT(NOUT,FILN)
C
C     Parameter
C
C          I/  NOUT  : Serial number of the current step/increment
C          I/  FILN  : Job name
C
C
C     Print the energy momentum tensor
C
      INCLUDE 'domain_common'
      CHARACTER*20 FILN
C
      WRITE(NOWO,1000)
      WRITE(NOWO,2000)(NC,NC=1,NCNTOU)
C
C
      DO 100 NNIS = 1,NTOTIS
C
             WRITE(NOWO,3000) SCURV(NNIS),(FNORM(K,NNIS),K=1,NCNTOU)
  100 CONTINUE
C
C
      WRITE(NOWP(NOUT),4000) FILN,NKSTEP(NOUT),NKINCR(NOUT)
C
      DO 300 K = 1,NCNTOU
      IF (K.GT.1) WRITE(NOWP(NOUT),5000) K+1
      DO 200 NNIS = 1,NTOTIS
C
             WRITE(NOWP(NOUT),6000) SCURV(NNIS),FNORM(K,NNIS)
  200 CONTINUE
  300 CONTINUE
C
C
```

```
1000 FORMAT(1H1,////,20X,'* * *  E N E R G Y  M O M E N T U M',
    &        ' T E N S O R  * * *',/)
 2000 FORMAT(/,1X,'CURV.COOR ',10(' CONT.#',I2,'   '))
 3000 FORMAT(1X,F8.4,2X,10(1X,E11.4))
 4000 FORMAT(1X,'curvilinear coordinate',/,1X,'normal force on the',
    &        ' interface (MPa)',/,'TYPE 3',/,'LABL 2',/,'.5 .5 .1',/
    &        ,A,/,'.5 .2 .1',/,'NSTEP : ',I2,'  NINCR :',I3,/,'END')
 5000 FORMAT('1.0E32 30',I1)
 6000 FORMAT(1X,F8.4,2X,E12.5)
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           F U N C T I O N    P D S H F N                       *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      FUNCTION PDSHFN(G,H,R,K,J,I)
C
C     Evaluate partial derivative of sh. function dN(k)/dc(j) at(G,H,R)
C
C     Parameters
C               G : 1st|   isoparametric coordinate of the location at
C               H : 2nd|   which the partial derivative must be
C               R : 3rd|   evaluated
C
C               K : number of the shape function
C               J : isoparametric coordinate with respect to which
C                   the shape function must be differentiated
C               I : Element type
C                                  8 : 2D - 8 nodes  isop. element
C
      INCLUDE 'domain_common'
C
      PDSHFN = 0.
C
      IF(I.EQ.8) GO TO 10
         KERROR = KERROR+1
         WRITE (NOWC,1000) I
         GO TO 900
 10   CONTINUE
C
      IF(J.EQ.1.OR.J.EQ.2) GO TO 20
         KERROR = KERROR+1
         WRITE (NOWC,2000) J
         GO TO 900
 20   CONTINUE
C
      GO TO (100,200,300,400,500,600,700,800) K
        KERROR = KERROR +1
        WRITE (NOWI,3000) K
        GO TO 900
```

```
C
 100  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H+G)
      GO TO 900
C
 200  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H-G)
      GO TO 900
C
 300  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H+G)
      GO TO 900
C
 400  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H-G)
      GO TO 900
C
 500  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1-H)*G
      IF(J.EQ.2) PDSHFN = -0.5*(1-G)*(1+G)
      GO TO 900
C
 600  CONTINUE
      IF(J.EQ.1) PDSHFN =  0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1+G)*H
      GO TO 900
C
 700  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1+H)*G
      IF(J.EQ.2) PDSHFN =  0.5*(1-G)*(1+G)
      GO TO 900
C
 800  CONTINUE
      IF(J.EQ.1) PDSHFN = - 0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1-G)*H
C
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'THE REQUIRED-COORDINATE-CODE MUST BE 1 (G) OR 2(H)',//,
     &  10X,'REQUIRED COORDINATE CODE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
     &  10X,'NODE NUMBER = ',I4)
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           F U N C T I O N    P D S H F N                       *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      FUNCTION PDSHFN(G,H,R,K,J,I)
C
C     Evaluate partial derivative of sh. function dN(k)/dc(j) at(G,H,R)
C
C     Parameters
C                G : 1st|   isoparametric coordinate of the location at
C                H : 2nd|   which the partial derivative must be
C                R : 3rd|   evaluated
C
C                K : number of the shape function
C                J : isoparametric coordinate with respect to which
C                    the shape function must be differentiated
C                I : Element type
C                                8 : 2D - 8 nodes  isop. element
C
      INCLUDE 'domain_common'
C
      PDSHFN = 0.
C
      IF(I.EQ.8) GO TO 10
         KERROR = KERROR+1
         WRITE (NOWC,1000) I
         GO TO 900
 10   CONTINUE
C
      IF(J.EQ.1.OR.J.EQ.2) GO TO 20
         KERROR = KERROR+1
         WRITE (NOWC,2000) J
         GO TO 900
 20   CONTINUE
C
      GO TO (100,200,300,400,500,600,700,800) K
        KERROR = KERROR +1
        WRITE (NOWI,3000) K
        GO TO 900
C
 100  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H+G)
      GO TO 900
C
 200  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1-H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H-G)
      GO TO 900
C
 300  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G+H)
      IF(J.EQ.2) PDSHFN = 0.25*(1+G)*(2*H+G)
      GO TO 900
C
 400  CONTINUE
      IF(J.EQ.1) PDSHFN = 0.25*(1+H)*(2*G-H)
      IF(J.EQ.2) PDSHFN = 0.25*(1-G)*(2*H-G)
      GO TO 900
C
 500  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1-H)*G
      IF(J.EQ.2) PDSHFN = -0.5*(1-G)*(1+G)
      GO TO 900
C
 600  CONTINUE
      IF(J.EQ.1) PDSHFN =  0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1+G)*H
      GO TO 900
C
 700  CONTINUE
      IF(J.EQ.1) PDSHFN = -(1+H)*G
      IF(J.EQ.2) PDSHFN =  0.5*(1-G)*(1+G)
      GO TO 900
C
 800  CONTINUE
      IF(J.EQ.1) PDSHFN = - 0.5*(1-H)*(1+H)
      IF(J.EQ.2) PDSHFN = -(1-G)*H
C
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     & 10X,'ELEMENT-TYPE = ',I4)
 2000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     & 10X,'THE REQUIRED-COORDINATE-CODE MUST BE 1 (G) OR 2(H)',//,
     & 10X,'REQUIRED COORDINATE CODE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     & 10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
     & 10X,'NODE NUMBER = ',I4)
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   P R E S F N                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE PRESFN
C
C     Evaluates partial derivative of the shape function at the nodes
C     (SFDNOD(8,8,2)) and at the integration points (SFDITP(9,8,2)) .
C     Evaluates also the quadratic form BQUAD(3,3,3) and the gauss
C     integration weights WEIGHTS(9).
C
      INCLUDE 'domain_common'
      DIMENSION C(3)
C
      DO 50 I = 1,3
 50   C(I) = 0.
C
      IR = IRDINT
      IT = ITYPE
C
      CALL BQCALC(IT)
      CALL WEICAL(IT,IR)
C
C     Derivatives of the shape functions at the nodes
C
      DO 200 I = 1,NNELT
            DO 100 M = 1,NDIM
 100        C(M) = DLISCO(0,I,M,IT,IR)
      DO 200 K = 1,NNELT
      DO 200 J = 1,NDIM
            SFDNOD(I,K,J) = PDSHFN(C(1),C(2),C(3),K,J,IT)
 200  CONTINUE
C
C     Derivatives of the shape functions and shape functions
c     at the integration points
C
      DO 400 I = 1,NINTP
            DO 300 M = 1,NDIM
 300        C(M) = DLISCO(1,I,M,IT,IR)
```

```
      DO 400 K = 1,NNELT
      SFNITP(I,K)=SHPFNC(C(1),C(2),C(3),K,IT)
      DO 400 J = 1,NDIM
            SFDITP(I,K,J) = PDSHFN(C(1),C(2),C(3),K,J,IT)
 400  CONTINUE
C
C
      IF (IOUTG.EQ.0) GO TO 700
         WRITE(NOWG,1000)
         DO 500 I = 1,NNELT
         DO 500 K = 1,NNELT
         DO 500 J = 1,NDIM
               WRITE(NOWG,2000) I,K,J,SFDNOD(I,K,J)
 500  CONTINUE
         WRITE(NOWG,3000)
         DO 600 I = 1,NINTP
         DO 600 K = 1,NNELT
         DO 600 J = 1,NDIM
               WRITE(NOWG,2000) I,K,J,SFDITP(I,K,J)
 600  CONTINUE
 700  CONTINUE
C
 1000 FORMAT(1H1,////,40X,'* * * SHAPE FUNCTION DERIVATIVE * * *',//,
     & 1X,'NODE  SH.FN  J   D(SH.NF)/D(COORD.J)',//)
 2000 FORMAT(1X,I4,3X,I2,4X,I1,4X,F10.5)
 3000 FORMAT(1H1,////,40X,'* * * SHAPE FUNCTION DERIVATIVE * * *',//,
     & 1X,'IT.P  SH.FN  J   D(SH.NF)/D(COORD.J)',//)
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   Q V C A L C                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE QVCALC(IFLAG,NNIS,IELC,IPONE,IPHALF,IPZERO,QVEC)
C
C     Evaluates the  perturbation  field Q at
C     the integration points of the element IELC
C
C     Parameters
C       I/  IFLAG : perturbation flag
C       I/  IPONE
C           IPHALF : nodes to be perturbed
C           IPZERO
C
C                     IFLAG = 0 ->partial perturbation Q  =1   on IPONE
C                                                         =1/2 on IPHALF
C                                                         =0   on IPZERO
C                     IFLAG = 1 ->Total perturbation Q  =1   on IPONE
C                                                       =1   on IPHALF
C                                                       =1   on IPZERO
C       I/    NNIS : ref. intf. node : Q = unit normal to intf at NNIS
C       I/    IELC : Local elt number
C       O/    QVEC:  Q at the integration points
C
C             QVEC(1,J) :  Q1 at integration point J
C             QVEC(2,J) :  Q2 at integration point J
C             QVEC(3,J) :  Q3 at integration point J
C
      INCLUDE 'domain_common'
C
      DIMENSION  F(3,8),QVEC(3,9)
C
      DO 10 N = 1,3
      DO 10 I = 1,8
 10   F(N,I) = 0.
C
      DO 300 ND = 1,NDIM
         F(ND,IPONE) = PATNOR(ND,NNIS)
```

```
C
         IF (IFLAG.EQ.1) F(ND,IPZERO) =PATNOR(ND,NNIS)
         IF (IPHALF.EQ.0) GO TO 50
         F(ND,IPHALF) = 0.5 * PATNOR(ND,NNIS)
         IF (IFLAG.EQ.1) F(ND,IPHALF) =PATNOR(ND,NNIS)
C
 50      CONTINUE
C
         DO 200 IP = 1,NINTP
C
            SUM = 0.
            DO 100 NN=1,NNELT
 100        SUM = SUM+SFNITP(IP,NN)*F(ND,NN)
            QVEC(ND,IP) = SUM
 200     CONTINUE
C
 300  CONTINUE
C
      RETURN
      END
C
```

```
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   R E S E T V                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
C
      SUBROUTINE RESETV
C
C     Reset to zero the memory used in the loop over the step/increment
C
      INCLUDE 'domain_common'
C
      DO 300 N = 1,1000
            IUGRAD(N) = 0.
            ICGRAD(N) = 0.
            DO 200 J=1,9
            DO 200 K=1,9
                  UGRAD(J,K,N) = 0.
                  DO 100 I=1,3
 100              CRGRAD(J,I,K,N) = 0.
 200        CONTINUE
 300  CONTINUE
C
      DO 600 N=1,200
            DO 400 I=1,10
 400        DELEN(I,N) = 0.
            DO 500 M=1,15
            DO 500 I=1,2
 500        DINTG(I,M,N) = 0.
 600  CONTINUE
C
      RETURN
      END
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *         S U B R O U T I N E   R W P E R M                    *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      SUBROUTINE RWPERM(ID,IMOVE,INEW,AM,AP,AN)
C
C     Change the order of the rows of AM if ZERO elements are on the
C     main diagonal.
C
C     Parameters:
C           I/     ID : matrix dimension
C           O/     IMOVE : row permutation code (=1 : row permuted )
C           O/     INEW : perm. record :INEW(J)= old # of new row #J
C           I/O    AM : square input/output matrix
C            /     AP : matrix used locally to keep record of non-zero
C                       pivots
C            /     AN : matrix used locally to store the new perm. matr.
C
      DIMENSION AM(ID,ID),AP(ID,ID),AN(ID,ID),INEW(ID)
      INCLUDE 'domain_common'
C
      DO 10 NR = 1,ID
            INEW(NR) = 0
      DO 10 NC = 1,ID
            AP(NR,NC) = 0.
            AN(NR,NC) = 0.
 10   CONTINUE
C
      IMOVE = 0
      PIVMIN = 0.00001
C
C     Check if matrix AM has zero elements on the main diagonal
C
      DO 50 NR = 1,ID
            IF(DABS(AM(NR,NR)).GE.PIVMIN) GO TO 50
            IMOVE = 1
 50   CONTINUE
C
      IF(IMOVE.EQ.0) GO TO 900
```

```
C
C     Permutation required
C
C     Prepare matrix AP with the position of non-zero pivots and
C     vector INEW with the # of rows which have non-zero pivots
C
      DO 100 NPIV = 1,ID
      DO 100 NR   = 1,ID
             IF(DABS(AM(NR,NPIV)).LT.PIVMIN) GO TO 100
               IPOS  = INEW(NPIV)+1
               INEW(NPIV) = IPOS
               AP(NPIV,IPOS) = DFLOAT(NR)
 100  CONTINUE
C
C     Permutation
C
      NP = 0
C
 200  CONTINUE
C
      DO 700 NPIV = 1,ID
C
         IF(INEW(NPIV).NE.NP) GO TO 700
C
         IF(NP.EQ.0) THEN
           IMOVE = -1
           WRITE(13,1000) NPIV
C
           DO 210 JJ = 1,ID
 210       WRITE(13,*) (AM(JJ,KK),KK=1,ID)
           GO TO 900
         ENDIF
C
         PIVMAX = PIVMIN
C
         DO 300 NCP = 1,NP
            NR = INT(AP(NPIV,NCP))
            PIVOT = DABS(AM(NR,NPIV))
            IF(PIVOT.GE.PIVMAX) THEN
               PIVMAX = PIVOT
               NRCH   = NR
            ENDIF
 300     CONTINUE
C
         INEW(NPIV) = -NRCH
      DO 350 NC = 1,ID
 350  AN(NPIV,NC) = AM(NRCH,NC)
C
C     Packing of AP
C
      DO 600 NR = 1,ID
         IF(INEW(NR).LT.0) GO TO 600
         DO 500 NC = 1,ID
            IF(INT(AP(NR,NC)).EQ.NRCE) THEN
              INEW(NR) = INEW(NR) -1
              DO 400 NLC = NC,ID-1
 400          AP(NR,NLC) = AP(NR,NLC+1)
              AP(NR,ID)  = 0.
              GO TO 600
            ENDIF
 500     CONTINUE
 600  CONTINUE
C
      NP = NP-1
      GO TO 200
 700  CONTINUE
C
      NP = NP+1
C
C     Check if all the pivot have been found
C
      INEG = 1
      DO 750 NR = 1, ID
 750  IF(INEW(NR).GT.0) INEG = 0
C
      IF(INEG.EQ.1) GO TO 800
C
      IF(NP.LE.ID) GO TO 200
C
      IMOVE = -1
      WRITE(13,2000)
      WRITE(13,*) INEW
      WRITE(13,3000)
      WRITE(13,*) AM
      WRITE(13,4000)
      WRITE(13,*) AP
      WRITE(13,5000)
      WRITE(13,*) AN
C
 800  CONTINUE
C
      DO 850 NR = 1,ID
         INEW(NR) = -INEW(NR)
      DO 850 NC = 1,ID
         AM(NR,NC) = AN(NR,NC)
 850  CONTINUE
C
 900  CONTINUE
C
1000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. RWPERM  * * *',//,
     &  20X,'NO PIVOT AVAILABLE FOR ROW N.',I4,//,
     &  20X,'INPUT MATRIX AM FOLLOWS ',//)
2000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. RWPERM  * * *',//,
     &  20X,'NP GREATER THAN ID.  NP = ',I4,' ID = ',I4,//,
     &  20X,'PERMUTATION RECORD VECTOR INEW  FOLLOWS ',//)
3000  FORMAT(1X,//,20X,'INPUT MATRIX AM FOLLOWS',//)
4000  FORMAT(1X,//,20X,'PIVOT MATRIX AP FOLLOWS',//)
5000  FORMAT(1X,//,20X,'OUTPUT MATRIX AN FOLLOWS',//)
      RETURN
      END
C
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *         F U N C T I O N    S H P F N C                       *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
      FUNCTION SHPFNC(G,H,R,K,I)
C
C     Evaluate  shape function N(k) at(G,H,R)
C
C     Parameters
C               G : 1st|   isoparametric coordinate of the location at
C               H : 2nd|   which the shape function must be
C               R : 3rd|   evaluated
C
C               K : number of the shape function
C               I : Element type
C                             8 : 2D - 8 nodes  isop. element
C
      INCLUDE 'domain_common'
C
      SHPFNC= 0.
C
      IF(I.EQ.8) GO TO 10
         KERROR = KERROR+1
         WRITE (NOWC,1000) I
         GO TO 900
 10   CONTINUE
C
      GO TO (100,200,300,400,500,600,700,800) K
        KERROR = KERROR +1
        WRITE (NOWI,3C00) K
        GO TO 900
C
 100  CONTINUE
      SHPFNC = -0.25*(1-G)*(1-H)*(1+G+H)
      GO TO 900
C
 200  CONTINUE
      SHPFNC = -0.25*(1+G)*(1-H)*(1-G+H)
      GO TO 900
C
 300  CONTINUE
      SHPFNC = -0.25*(1+G)*(1+H)*(1-G-H)
      GO TO 900
C
 400  CONTINUE
      SHPFNC = -0.25*(1-G)*(1+H)*(1+G-H)
      CO TO 900
C
 500  CONTINUE
      SHPFNC = 0.5*(1-G)*(1+G)*(1-H)
      GO TO 900
C
 600  CONTINUE
      SHPFNC = 0.5*(1+G)*(1-H)*(1+H)
      GO TO 900
C
 700  CONTINUE
      SHPFNC = 0.5*(1-G)*(1+G)*(1+H)
      GO TO 900
C
 800  CONTINUE
      SHPFNC = 0.5*(1-G)*(1+H)*(1-H)
C
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
 3000 FORMAT(1H1,///,20X,'* * * ERROR IN FNC.PDSHFN  * * *',//,
     &  10X,'FOR THIS EL-TYPE THE NODE NUMBER MUST BE BETW. 1 AND 8',//,
     &  10X,'NODE NUMBER = ',I4)
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *          S U B R O U T I N E   S I D N O D                    *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
C
      SUBROUTINE SIDNOD(IFLAG,N1,N2,N3,NSIDE)
C
C     Find the side to which N1,N2,N3 belong (if IFLAG = 1) or
C     the nodes N1,N2,N3 that belong to NSIDE (if IFLAG = 0)
C
C     Parameters
C           I/     L : identity flag: 0-> find node  1-> find side
C           I/O    N1,N2,N3 : node position (1-8)
C           I/O    NSIDE : side (-1/-4 +1/+4)(nodes must be consecutive)
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ================================================================
C
      INCLUDE 'domain_common'
C
      IF(IFLAG.EQ.0) GO TO 500
C
C     NSIDE has to be found
C
      NSIDE = 0
      GO TO (100,200,300,400)N1
      GO TO 600
C
 100  CONTINUE
      IF(N2.EQ.5.AND.N3.EQ.2) NSIDE = 1
      IF(N2.EQ.8.AND.N3.EQ.4) NSIDE =-4
      GO TO 600
C
 200  CONTINUE
      IF(N2.EQ.6.AND.N3.EQ.3) NSIDE = 2
      IF(N2.EQ.5.AND.N3.EQ.1) NSIDE =-1
      GO TO 600
C
 300  CONTINUE
      IF(N2.EQ.7.AND.N3.EQ.4) NSIDE = 3
      IF(N2.EQ.6.AND.N3.EQ.2) NSIDE =-2
      GO TO 600
C
 400  CONTINUE
      IF(N2.EQ.8.AND.N3.EQ.1) NSIDE = 4
      IF(N2.EQ.7.AND.N3.EQ.3) NSIDE =-3
      GO TO 600
C
 500  CONTINUE
C
C     N1,N2,N3 have to be found
C
      NS = NSIDE +5
C
      GO TO (510,520,530,540,550,560,570,580,590) NS
C
         KERROR = KERROR + 1
         WRITE(NOWC,1000) NSIDE
         GO TO 600
C
 510  CONTINUE
      N1 = 1
      N2 = 8
      N3 = 4
      GO TO 600
 520  CONTINUE
      N1 = 4
      N2 = 7
      N3 = 3
      GO TO 600
 530  CONTINUE
      N1 = 3
      N2 = 6
      N3 = 2
      GO TO 600
 540  CONTINUE
      N1 = 2
      N2 = 5
      N3 = 1
      GO TO 600
 550  CONTINUE
      KERROR = KERROR+1
      WRITE(NOWC,1000) NSIDE
      GO TO 600
```

```
  560  CONTINUE
       N1 = 1
       N2 = 5
       N3 = 2
       GO TO 600
  570  CONTINUE
       N1 = 2
       N2 = 6
       N3 = 3
       GO TO 600
  580  CONTINUE
       N1 = 3
       N2 = 7
       N3 = 4
       GO TO 600
  590  CONTINUE
       N1 = 4
       N2 = 8
       N3 = 1
C
  600  CONTINUE
C
 1000  FORMAT(1H1,///,20X,'* * * ERROR IN SBR. SIDNOD  * * *',//,
     &  10X,'THE SIDE NUMBER MUST BE BETW.-4 AND -1 OR 1 AND 4',//,
     &  10X,'SIDE NUMBER = ',I4)
C
       RETURN
       END
```

```
C
C
C     **************************************************************
C     *                                                            *
C     *                                                            *
C     *          S U B R O U T I N E   S O L V E R                 *
C     *                                                            *
C     *                                                            *
C     **************************************************************
C
      SUBROUTINE SOLVER(ID,IMOVE,INEW,AU,AL,B,X)
C
C     Solve the systems AL*C = B and AU*X = C
C     If matrix AM has been permuted, the RHS is permuted too
C
C     Parameters
C          I/     ID : matrix/vector dimension
C          I/     IMOVE : row permutation code (=1 : row permuted )
C          I/     INEW : perm. record :INEW(J)= old # of new row #J
C          I/     AL : lower triang matrix AL(ID,ID)
C          I/     AU : upper triang.matrix AU(ID,ID)
C          I/     B  : RHS of the system AL*AU * X = B
C          O/     X  : unknown vector
C
      INCLUDE 'domain_common'
      DIMENSION AU(ID,ID), AL(ID,ID), B(ID), X(ID),INEW(ID)
C
C     Check if rows of AM have been permuted. If YES, permute B
C
      CALL CKPERM(ID,IMOVE,INEW,B,X)
C
      DO 10 NC = 1,ID
   10 X(NC) = 0.
C
C     Evaluate X  -> AL*X = B
C
      DO 200 NR = 1,ID
            SUM = 0.
         IF(NR.EQ.1) GO TO 200
            DO 100 NC = 1,NR-1
  100       SUM = SUM + X(NC) * AL(NR,NC)
  200 X(NR) = B(NR) -SUM
C
C     Copy X in B
C
```

```
      DO 250 I = 1,ID
      B(I) = X(I)
  250 X(I) = 0.

C     Evaluate X  -> AU*X = B
C
      DO 400 NR = ID,1,-1
            SUM = 0.
         IF (NR.EQ.ID) GO TO 400
            DO 300 NC = NR+1,ID
  300       SUM = SUM + X(NC) * AU(NR,NC)
  400 X(NR) = (B(NR)-SUM)/AU(NR,NR)
C
C
      RETURN
      END
C
```

```
C
C
C     *****************************************************************
C     *                                                               *
C     *                                                               *
C     *           S U B R O U T I N E   S Y S O L V                   *
C     *                                                               *
C     *                                                               *
C     *****************************************************************
C
      SUBROUTINE SYSOLV
C
C     Solve the system AMAT*F=DELEN and evaluates FNORM=force
C     on the interface
C
      INCLUDE 'domain_common'
C
       DIMENSION AM(200,200),AU(200,200),AL(200,200)
       DIMENSION B(200),X(200), INEW(200)
C
C     Local array :
C            AM : coeff. matrix =AMAT
C            AL,AU : Triang. fact. matr. AM = AL*AU
C            B : RHS -> Bj(I) = DELEN(j,I)
C            X : Unknown vect -> Xj(I) = FNORM(j,I)
C
      DO 10 I = 1,200
      DO 10 J = 1,200
   10 AM(I,J) = AMAT(I,J)
C
C     LU factorization
C
      KER = KERROR
      CALL LUFACT(KER,200,IMOVE,INEW,AM,AU,AL)
      KERROR = KER
      IF(KERROR.GT.0) GO TO 500
C
C     Solving the system for all the components of DELEN
C
      DO 300 NCONT = 1,NCNTOU
C
C        Evaluating the RHS
C
         DO 100 I = 1,200
  100    B(I) = DELEN(NCONT,I)
```

```
C
C        solving the system
C
         CALL SOLVER (200,IMOVE,INEW,AU,AL,B,X)
C
C        Storing the solution
C
         DO 200 I=1,200
 200     FNORM(NCONT,I) = X(I)
C
 300  CONTINUE
C
 500  CONTINUE
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *            S U B R O U T I N E   T A N G 2 D                   *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
      SUBROUTINE TANG2D(X,T)
C
C     Evaluates the components of the unit tangent vector at the three
C     subsequent nodes on a side af a 2nd ord. isop 2D elt
C
C     Parameters
C          I/    X : Cartesian coordinate of the nodes
C          O/    T : Cartesian components of the unit tangent vector
C                    at the nodes
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ==============================================================
C
      INCLUDE 'domain_common'
C
      DIMENSION X(2,3),T(2,3),DXDS(2,3),SQMOD(3)
C
C     First node
C
      SQMOD(1) = 0.
      DO 100 I = 1,2
            DXDS(I,1) = -1.5 * X(I,1) + 2 * X(I,2) - 0.5 * X(I,3)
 100  SQMOD(1) = SQMOD(1) + DXDS(I,1)**2
C
C     second node
C
      SQMOD(2) = 0.
      DO 200 I = 1,2
            DXDS(I,2) = -0.5 * X(I,1) + 0.5 * X(I,3)
 200  SQMOD(2) = SQMOD(2) + DXDS(I,2)**2
c
c     third node
C
      SQMOD(3) = 0.
      DO 300 I = 1,2
      DXDS(I,3) = 0.5 * X(I,1) - 2 * X(I,2) + 1.5 * X(I,3)
  300 SQMOD(3) = SQMOD(3) + DXDS(I,3)**2
C
      DO 400 N = 1,3
      DO 400 I = 1,2
  400 T(I,N)  = DXDS(I,N)/DSQRT(SQMOD(N))
C
      RETURN
      END
C
```

```
C
C
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           S U B R O U T I N E   V A R I N P                    *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
      SUBROUTINE VARINP(NOUT)
C
C     Parameter
C          I/  NOUT  : Serial number of the required step/increment
C
C
C     Reads Elastic energy and Stres tensor at the integration points
C     Reads also the displacement field and the creep strains at the nodes
C
      INCLUDE 'domain_common'
      DIMENSION VARELT(9,9,1000),VARNOD(9,3200)
C
C     Input the elastic energy
C
      CALL F8IPIN(NOUT,14,VARELT)
C
      DO 100 IE = 1, NTELT
      DO 100 IP = 1,NINTP
  100 WENIP(IP,IE) = VARELT(1,IP,IE)
C
C     Input the stress tensor
C
      CALL F8IPIN(NOUT,11,VARELT)
C
      DO 200 IE = 1, NTELT
      DO 200 IP = 1,NINTP
      DO 200 NC = 1,9
  200 SIGMIP(NC,IP,IE) = VARELT(NC,IP,IE)
C
C     Input the displacement at the nodes
C
      CALL F8NOIN(NOUT,101,VARNOD)
C
      DO 300 NNLC = 1,NTNOD
      DO 300 ND = 1,NDIM
  300 UNODE(ND,NNLC) = VARNOD(ND,NNLC)
C
      IF(ICREEP.EQ.1) THEN
C
C         Input the creep strain at the nodes
C
          CALL F8NOIN(NOUT,23,VARNOD)
C
          DO 400 NNLC = 1,NTNOD
          DO 400 NC = 1,9
  400     CRESTR(NC,NNLC) = VARNOD(NC,NNLC)
C
      ENDIF
C
      RETURN
      END
```

```
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     *           S U B R O U T I N E    W E I C A L                 *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
      SUBROUTINE WEICAL(IT,IR)
C
C     Evaluates the gauss weights at integration points WEIGHT(9)
C
C     Parameters
C         I/      I : Element type
C                                8 : 2D - 8 nodes  isop. element
C         I/      IR: Reduced integration flag
C                                0 : full integration (9 i.p.)
C                                1 : reduced integration (4 i.p.)
C
C     NOTE!!!! ONLY FOR 2nd ORDER - 8 NODES - ISOPARAMETRIC 2D ELEMENTS
C     ================================================================
C
      INCLUDE 'domain_common'
C
      FIV = .555555555555556
      EIG = .888888888888889
C
      IF ( IT.EQ.8) GO TO 100
         KERROR = KERROR + 1
         WRITE(NOWC,1000) IT
         GO TO 900
  100 CONTINUE
C
      IF(IR.EQ.1) GO TO 200
C
C     Full integration
C
      WEIGHT(1) = FIV*FIV
      WEIGHT(2) = EIG*FIV
      WEIGHT(3) = FIV*FIV
      WEIGHT(4) = EIG*FIV
      WEIGHT(5) = EIG*EIG
      WEIGHT(6) = EIG*FIV
      WEIGHT(7) = FIV*FIV
      WEIGHT(8) = EIG*FIV
      WEIGHT(9) = FIV*FIV
C
      GO TO 900
C
 200  CONTINUE
C
C     Reduced integration
C
      WEIGHT(1) = 1.
      WEIGHT(2) = 1.
      WEIGHT(3) = 1.
      WEIGHT(4) = 1.
      WEIGHT(5) = 0.
      WEIGHT(6) = 0.
      WEIGHT(7) = 0.
      WEIGHT(8) = 0.
      WEIGHT(9) = 0.
C
 900  CONTINUE
C
 1000 FORMAT(1H1,///,20X,'* * * ERROR IN SBR. WEICAL  * * *',//,
     &  10X,'ONLY ELEMENT-TYPE 8 IS IMPLEMENTED',//,
     &  10X,'ELEMENT-TYPE = ',I4)
C
      RETURN
      END
C
```

```
C
C
C
C     ****************************************************************
C     *                                                              *
C     *                                                              *
C     * C O M M O N   B L O K S   O F   P R O G R A M   D O M A I N  *
C     *                                                              *
C     *                                                              *
C     ****************************************************************
C
C
C     Common variables, program DOMAIN
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON / CONN / NCONN(21,2,200),IELTOP(17,1000),NABAQ(3200)
      COMMON / CONL / LELCON(5,2,100),LNOCON(5,200)
      COMMON / COOR / COORDS(3,3200)
      COMMON / CTRL / ITYPE,IRDINT,IOUTG,KPSTOP,IDOUBL,ICREEP
      COMMON / CTRV / NCNTOU,NLAYER(2,10)
      COMMON / ELHS / DINTG(2,15,200),DELEN(10,200)
      COMMON / ERRO / KERROR
      COMMON / EVR1 / WENIP(9,1000),SIGMIP(9,9,1000)
      COMMON / EVR2 / UGRAD(9,9,1000),IUGRAD(1000)
      COMMON / EVR3 / CRGRAD(9,3,9,1000),ICGRAD(1000)
      COMMON / INTF / PATNOR(3,200),PATANG(3,200),SCURV(200)
      COMMON / MATR / AMAT(200,200)
      COMMON / NUMB / NNELT,NNLELT,NDIM,NINTP,NTOTIS,NTNOD,NTELT,NTLELT
      COMMON / NVAR / UNODE(3,3200),FNORM(10,200),CRESTR(9,3200)
      COMMON / SHFN / SFDITP(9,8,2),SFDNOD(8,8,2),SFNITP(9,8)
      COMMON / STEP / NKSTEP(19),NKINCR(19),NTOUT
      COMMON / UNIT / NORI,NOWC,NOWO,NOWG,NOWP(19)
      COMMON / WEIG / BQUAD(3,3,3),WEIGHT(9)
C
```

APPENDIX III: THE COMPUTER PROGRAM GOODIER

```
C     ******************************************************************
C     *                                                                *
C     *                                                                *
C     *           P R O G R A M    G O O D I E R                       *
C     *                                                                *
C     *                                                                *
C     ******************************************************************
C
C
C     Evaluates the normal component of the force on the interface
C     of a cylindrical misfitting inclusion in an infinite matrix
C     (both isotropic media with the same poisson ratio).
C
C     Values are given at N (number of nodes) locations along the
C     interface.
C
C     Only the upper-right quadrant of the inclusion is considered (for
C     symmetry) the first node is at angle=0. with the direction of the
C     applied stress .
C
C     File Handling
C
      DIMENSION TAUN(100)
      CHARACTER*80 FILOUT
C
C
      PRINT *,'* * *           PROGRAM GOODIER              * * *'
      PRINT *,' '
      PRINT *,''
C
      PRINT 1000,' Please enter the FILOUT name'
      READ(*,*) FILOUT
C
      OPEN(UNIT = 13,FILE = FILOUT,STATUS = 'NEW')
C
C
      NOW = 13
      SMALL = 1.E-32
      PI2 = ASIN(1.)
      RAD = 1./PI2
c
      PRINT 1000,' Please enter E matrix'
      READ(*,*) EMATRI
      WRITE(*,*) EMATRI
      PRINT 1000,' Please enter E inclusion'
```

```
      READ(*,*) EINCLU
      WRITE(*,*) EINCLU
      PRINT 1000,' Please enter Poisson ratio'
      READ(*,*) PNU
      WRITE(*,*) PNU
C
      AMUM = EMATRI/(2.*(1+PNU))
      AMUI = EINCLU/(2.*(1+PNU))
C
      WRITE(*,*)'AMUM ',AMUM,'  AMUI ',AMUI
C
      PRINT 1000,' Please enter Sigma inf.'
      READ(*,*) SINF
      WRITE(*,*) SINF
      PRINT 1000,' Please enter misfit'
      READ(*,*) DELTA
      WRITE(*,*) DELTA
C
C     evaluate factor K (AK) , L (AL) , M (AM)
c
      AK =(AMUM-AMUI)/(AMUM+AMUI/(1.-2.*PNU))
      AL =(AMUM-AMUI)/(AMUM+(3.-4.*PNU)*AMUI)
      AM = 1. + (1.-2.*PNU)*AMUM/AMUI
C
      WRITE(*,*) 'AK ',AK,'  AL ',AL,'  AM ',AM
C
      PRINT 1000,' Please enter number of nodes'
      READ(*,*) NNOD
      WRITE(*,*) NNOD
C
      ASTEP = PI2/(NNOD-1)
C
            SUMLEF = 0.
            SUMRIG = 0.

C
      DO 200 NN = 1,NNOD
C
C
      ANG = (NN-1)*ASTEP
      SCURV = RAD*ANG
C
C     Inside the inclusion.Jump if hole (AMUI = 0)
C
      IF(AMUI.LT.SMALL) GO TO 100
```

```
C
      SRR = (1.-AK)*SINF/2.+ (1.-AL)*COS(2.*ANG)*SINF/2.+
     &      -2.*AMUM*DELTA/AM
      STT = (1.-AK)*SINF/2.- (1.-AL)*COS(2.*ANG)*SINF/2.+
     &      -2.*AMUM*DELTA/AM
      SRT =                  - (1.-AL)*SIN(2.*ANG)*SINF/2.
C
      DURDR= (SINF/(4.*AMUI))*((1.-2.*PNU)*(1.-AK)+
     &                         (1.-AL)*COS(2.*ANG))+
     &        +DELTA*(1.-(1.-2.*PNU)*AMUM/(AM*AMUI))
      DUTDR=-(SINF/(4.*AMUI))*(1.-AL)*SIN(2.*ANG)
C
      W = (1./(4.*AMUI))*((1.-PNU)*(SRR**2+STT**2)-
     &    2.*PNU*SRR*STT + 2.*SRT**2)
      TDUDN = SRR*DURDR+SRT*DUTDR
C
      EMTI = W-TDUDN
C
 100  CONTINUE
C
C
C     In the matrix
C
      SRR = (1.-AK)*SINF/2.+ (1.-AL)*COS(2.*ANG)*SINF/2.+
     &      -2.*AMUM*DELTA/AM
      STT = (1.+AK)*SINF/2.- (1.+3.*AL)*COS(2.*ANG)*SINF/2.+
     &      +2.*AMUM*DELTA/AM
      SRT =                  - (1.-AL)*SIN(2.*ANG)*SINF/2.
C
      DURDR= (SINF/(4.*AMUM))*((1.-2.*PNU)-AK+
     &                         (1.-AL*(1.-4.*PNU))*COS(2.*ANG))+
     &        -DELTA/AM
      DUTDR= (SINF/(4.*AMUM))*(AL*(5.-4.*PNU)-1)*SIN(2.*ANG)
C
      W = (1./(4.*AMUM))*((1.-PNU)*(SRR**2+STT**2)-
     &    2.*PNU*SRR*STT + 2.*SRT**2)
      TDUDN = SRR*DURDR+SRT*DUTDR
C
      EMTM = W-TDUDN
      TAUN(NN) = EMTM-EMTI
C
      WRITE(NOW,*) SCURV,TAUN(NN)
C
             IF(NN.LE.NNOD/2) SUMLEF = SUMLEF +TAUN(NN)
     IF(NN.GT.NNOD/2) SUMRIG = SUMRIG +TAUN(NN)
C
C
 200  CONTINUE
            YMLEF = SUMLEF*2/(NNGD)
            YMRIG = SUMRIG*2/(NNOD)
C
      WRITE(NOW,*)YMLEF,YMRIG
C
 1000 FORMAT($,A,' : ')
C
      STOP
      END
```

APPENDIX IV: ABAQUS INPUT FILE FOR THE TEST CASE: A CYLINDRICAL INCLUSION IN AN INFINITE MATRIX

```
**
**    Abaqus 4.7  input file for cylindrical inclusion
**                in an infinite matrix.
*HEADING
INCLUSION 200/100
** NEUTRAL FILE GENERATED ON: 04-JUN-87
*NODE, INPUT=16,NSET=ALL
*ELEMENT, TYPE=CPE8, ELSET=FIBER
   367   59   60   62   61  187  189  191  188
   368   61   62   64   63  191  193  195  192
   388   63   64   66   65  195  197  199  196
   389   48   63   65   49  194  196  198  166
   390   47   61   63   48  190  192  194  164
   391   46   59   61   47  186  188  190  162
   392   65   66   68   67  199  201  203  200
   393   67   68   70   69  203  205  207  204
   394   69   70   72   71  207  209  211  208
   395   71   72   74   73  211  213  215  212
   396   81   82   84   83  231  233  235  232
   397   73   74   76   75  215  217  219  216
   398   79   80   82   81  227  229  231  228
   399   75   76   78   77  219  221  223  220
   400   77   78   80   79  223  225  227  224
   401   49   65   67   50  198  200  202  168
   402   42   49   50   41  167  168  169  157
   403   43   48   49   42  165  166  167  158
   404   44   47   48   43  163  164  165  159
   405   45   46   47   44  161  162  163  160
   406   50   67   69   51  202  204  206  170
   407   51   69   71   52  206  208  210  172
   408   52   71   73   53  210  212  214  174
   409   53   73   75   54  214  216  218  176
   410   57   81   83   58  230  232  234  184
   411   56   79   81   57  226  228  230  182
   412   54   75   77   55  218  220  222  178
   413   55   77   79   56  222  224  226  180
   414   41   50   51   36  169  170  171  149
   415   36   37   42   41  148  151  157  149
   416   37   38   43   42  150  153  158  151
   417   38   39   44   43  152  155  159  153
   418   39   40   45   44  154  156  160  155
   419   36   51   52   31  171  172  173  140
   420   31   52   53   26  173  174  175  131
   421   26   53   54   21  175  176  177  122
   422    6   57   58    1  183  184  185   86
   423   21   54   55   16  177  178  179  113
   424   11   56   57    6  181  182  183   95
   425   16   55   56   11  179  180  181  104
   426   31   32   37   36  139  142  148  140
   427   32   33   38   37  141  144  150  142
   428   33   34   39   38  143  146  152  144
   429   34   35   40   39  145  147  154  146
   430   26   27   32   31  130  133  139  131
   431   21   22   27   26  121  124  130  122
   432    1    2    7    6   85   88   94   86
   433   16   17   22   21  112  115  121  113
   434    6    7   12   11   94   97  103   95
   435   11   12   17   16  103  106  112  104
   436   27   28   33   32  132  135  141  133
   437   28   29   34   33  134  137  143  135
   438   29   30   35   34  136  138  145  137
   439   22   23   28   27  123  126  132  124
   440    2    3    8    7   87   90   96   88
   441   17   18   23   22  114  117  123  115
   442    7    8   13   12   96   99  105   97
   443   12   13   18   17  105  108  114  106
   444   23   24   29   28  125  128  134  126
   445   24   25   30   29  127  129  136  128
   446    3    4    9    8   89   92   98   90
   447    4    5   10    9   91   93  100   92
   448   18   19   24   23  116  119  125  117
   449    8    9   14   13   98  101  107   99
   450   13   14   19   18  107  110  116  108
   451   19   20   25   24  118  120  127  119
   452    9   10   15   14  100  102  109  101
   453   14   15   20   19  109  111  118  110
*ELEMENT, TYPE=CPE8, ELSET=NEAR
    73  523  524  527  526 1179 1181 1185 1180
    74  526  527  530  529 1185 1187 1191 1186
    75  529  530  533  532 1191 1193 1196 1192
    76  532  533  536  535 1196 1199 1203 1198
    77  535  536  539  538 1203 1205 1209 1204
    78  538  539  542  541 1209 1211 1214 1210
    79  541  542  545  544 1214 1217 1221 1216
    80  556  557  560  559 1245 1247 1250 1246
    81  544  545  548  547 1221 1223 1227 1222
    82  553  554  557  556 1239 1241 1245 1240
    83  547  548  551  550 1227 1229 1233 1228
    84  550  551  554  553 1232 1235 1239 1234
    85  522  523  526  525 1177 1180 1183 1178
```

86	525	526	529	528	1183	1186	1189	1184
87	528	529	532	531	1189	1192	1195	1190
88	531	532	535	534	1195	1198	1201	1197
89	534	535	538	537	1201	1204	1207	1202
90	537	538	541	540	1207	1210	1213	1208
91	540	541	544	543	1213	1216	1219	1215
92	555	556	559	558	1243	1246	1249	1244
93	543	544	547	546	1219	1222	1225	1220
94	552	553	556	555	1237	1240	1243	1238
95	546	547	550	549	1225	1228	1231	1226
96	549	550	553	552	1231	1234	1237	1233
97	473	522	525	477	1176	1178	1182	1083
98	477	525	528	481	1182	1184	1188	1091
99	481	528	531	485	1188	1190	1194	1099
100	485	531	534	489	1194	1197	1200	1107
101	489	534	537	493	1200	1202	1206	1115
102	493	537	540	497	1206	1208	1212	1123
103	497	540	543	501	1212	1215	1218	1131
104	517	555	558	521	1242	1244	1248	1171
105	501	543	546	505	1218	1220	1224	1139
106	513	552	555	517	1236	1238	1242	1163
107	505	546	549	509	1224	1226	1230	1147
108	509	549	552	513	1230	1233	1236	1155
109	472	473	477	476	1081	1083	1089	1082
110	476	477	481	480	1089	1091	1097	1090
111	480	481	485	484	1097	1099	1105	1098
112	484	485	489	488	1105	1107	1111	1106
113	488	489	493	492	1111	1115	1121	1114
114	492	493	497	496	1121	1123	1129	1122
115	496	497	501	500	1129	1131	1137	1130
116	516	517	521	520	1169	1171	1175	1170
117	500	501	505	504	1137	1139	1143	1138
118	512	513	517	516	1161	1163	1169	1162
119	504	505	509	508	1143	1147	1153	1146
120	508	509	513	512	1153	1155	1161	1154
121	471	472	476	475	1079	1082	1087	1080
122	475	476	480	479	1087	1090	1095	1088
123	479	480	484	483	1095	1098	1103	1096
124	483	484	488	487	1103	1106	1110	1104
125	487	488	492	491	1110	1114	1119	1113
126	491	492	496	495	1119	1122	1127	1120
127	495	496	500	499	1127	1130	1135	1128
128	515	516	520	519	1167	1170	1174	1168
129	499	500	504	503	1135	1138	1142	1136
130	511	512	516	515	1159	1162	1167	1160
131	503	504	508	507	1142	1146	1151	1145
132	507	508	512	511	1151	1154	1159	1152
133	470	471	475	474	1077	1080	1085	1078
134	474	475	479	478	1085	1088	1093	1086
135	478	479	483	482	1093	1096	1101	1094
136	482	483	487	486	1101	1104	1109	1102
137	486	487	491	490	1109	1113	1117	1112
138	490	491	495	494	1117	1120	1125	1118
139	494	495	499	498	1125	1128	1133	1126
140	514	515	519	518	1165	1168	1173	1166
141	498	499	503	502	1133	1136	1141	1134
142	510	511	515	514	1157	1160	1165	1158
143	502	503	507	506	1141	1145	1149	1144
144	506	507	511	510	1149	1152	1157	1150
145	409	470	474	410	1076	1078	1084	960
146	410	474	478	411	1084	1086	1092	961
147	411	478	482	412	1092	1094	1100	962
148	412	482	486	413	1100	1102	1108	963
149	413	486	490	438	1108	1112	1116	1016
150	438	490	494	439	1116	1118	1124	1017
151	439	494	498	440	1124	1126	1132	1018
152	468	514	518	469	1164	1166	1172	1075
153	440	498	502	441	1132	1134	1140	1019
154	467	510	514	468	1156	1158	1164	1074
155	441	502	506	466	1140	1144	1148	1072
156	466	506	510	467	1148	1150	1156	1073
157	404	409	410	405	952	960	954	951
158	405	410	411	406	954	961	956	953
159	406	411	412	407	956	962	958	955
160	407	412	413	408	958	963	959	957
161	408	413	438	434	959	1016	1010	1008
162	434	438	439	435	1010	1017	1012	1009
163	435	439	440	436	1012	1018	1014	1011
164	464	468	469	465	1070	1075	1071	1069
165	436	440	441	437	1014	1019	1015	1013
166	463	467	468	464	1068	1074	1070	1067
167	437	441	466	462	1015	1072	1066	1064
168	462	466	467	463	1066	1073	1068	1065
169	399	404	405	400	943	951	945	942
170	400	405	406	401	945	953	947	944
171	401	406	407	402	947	955	949	946
172	402	407	408	403	949	957	950	948
173	403	408	434	430	950	1008	1002	1000
174	430	434	435	431	1002	1009	1004	1001
175	431	435	436	432	1004	1011	1006	1003

176	460	464	465	461	1062	1069	1063	1061
177	432	436	437	433	1006	1013	1007	1005
178	459	463	464	460	1060	1067	1062	1059
179	433	437	462	458	1007	1064	1058	1056
180	458	462	463	459	1058	1065	1060	1057
181	394	399	400	395	934	942	936	933
182	395	400	401	396	936	944	938	935
183	396	401	402	397	938	946	940	937
184	397	402	403	398	940	948	941	939
185	398	403	430	426	941	1000	994	992
186	426	430	431	427	994	1001	996	993
187	427	431	432	428	996	1003	998	995
188	456	460	461	457	1054	1061	1055	1053
189	428	432	433	429	998	1005	999	997
190	455	459	460	456	1052	1059	1054	1051
191	429	433	458	454	999	1056	1050	1048
192	454	458	459	455	1050	1057	1052	1049
193	389	394	395	390	925	933	927	924
194	390	395	396	391	927	935	929	926
195	391	396	397	392	929	937	931	928
196	392	397	398	393	931	939	932	930
197	393	398	426	422	932	992	986	984
198	422	426	427	423	986	993	988	985
199	423	427	428	424	988	995	990	987
200	452	456	457	453	1046	1053	1047	1045
201	424	428	429	425	990	997	991	989
202	451	455	456	452	1044	1051	1046	1043
203	425	429	454	450	991	1048	1042	1040
204	450	454	455	451	1042	1049	1044	1041
205	384	389	390	385	916	924	918	915
206	385	390	391	386	918	926	920	917
207	386	391	392	387	920	928	922	919
208	387	392	393	388	922	930	923	921
209	388	393	422	418	923	984	978	976
210	418	422	423	419	978	985	980	977
211	419	423	424	420	980	987	982	979
212	448	452	453	449	1038	1045	1039	1037
213	420	424	425	421	982	989	983	981
214	447	451	452	448	1036	1043	1038	1035
215	421	425	450	446	983	1040	1034	1032
216	446	450	451	447	1034	1041	1036	1033
217	379	384	385	380	907	915	909	906
218	380	385	386	381	909	917	911	908
219	381	386	387	382	911	919	913	910
220	382	387	388	383	913	921	914	912

221	383	388	418	414	914	976	970	968
222	414	418	419	415	970	977	972	969
223	415	419	420	416	972	979	974	971
224	444	448	449	445	1030	1037	1031	1029
225	416	420	421	417	974	981	975	973
226	443	447	448	444	1028	1035	1030	1027
227	417	421	446	442	975	1032	1026	1024
228	442	446	447	443	1026	1033	1028	1025
229	286	379	380	287	901	906	902	729
230	287	380	381	288	902	908	903	730
231	288	381	382	289	903	910	904	731
232	289	382	383	290	904	912	905	732
233	290	383	414	331	905	968	964	813
234	331	414	415	332	964	969	965	814
235	332	415	416	333	965	971	966	815
236	377	444	445	378	1022	1029	1023	900
237	333	416	417	334	966	973	967	816
238	376	443	444	377	1021	1027	1022	899
239	334	417	442	375	967	1024	1020	897
240	375	442	443	376	1020	1025	1021	898
241	281	286	287	282	721	729	723	720
242	282	287	288	283	723	730	725	722
243	283	288	289	284	725	731	727	724
244	284	289	290	285	727	732	728	726
245	285	290	331	327	728	813	807	805
246	327	331	332	328	807	814	809	806
247	328	332	333	329	809	815	811	808
248	373	377	378	374	895	900	896	894
249	329	333	334	330	811	816	812	810
250	372	376	377	373	893	899	895	892
251	330	334	375	371	812	897	891	889
252	371	375	376	372	891	898	893	890
253	276	281	282	277	712	720	714	711
254	277	282	283	278	714	722	716	713
255	278	283	284	279	716	724	718	715
256	279	284	285	280	718	726	719	717
257	280	285	327	323	719	805	799	797
258	323	327	328	324	799	806	801	798
259	324	328	329	325	801	808	803	800
260	369	373	374	370	887	894	888	886
261	325	329	330	326	803	810	804	802
262	368	372	373	369	885	892	887	884
263	326	330	371	367	804	889	883	881
264	367	371	372	368	883	890	885	882
265	271	276	277	272	703	711	705	702

266	272	277	278	273	705	713	707	704
267	273	278	279	274	707	715	709	706
268	274	279	280	275	709	717	710	708
269	275	280	323	319	710	797	791	789
270	319	323	324	320	791	798	793	790
271	320	324	325	321	793	800	795	792
272	365	369	370	366	879	886	880	878
273	321	325	326	322	795	802	796	794
274	364	368	369	365	877	884	879	876
275	322	326	367	363	796	881	875	873
276	363	367	368	364	875	882	877	874
277	266	271	272	267	694	702	696	693
278	267	272	273	268	696	704	698	695
279	268	273	274	269	698	706	700	697
280	269	274	275	270	700	708	701	699
281	270	275	319	315	701	789	783	781
282	315	319	320	316	783	790	785	782
283	316	320	321	317	785	792	787	784
284	361	365	366	362	871	878	872	870
285	317	321	322	318	787	794	788	786
286	360	364	365	361	869	876	871	868
287	318	322	363	359	788	873	867	865
288	359	363	364	360	867	874	869	866
289	261	266	267	262	685	693	687	684
290	262	267	268	263	687	695	689	686
291	263	268	269	264	689	697	691	688
292	264	269	270	265	691	699	692	690
293	265	270	315	311	692	781	775	773
294	311	315	316	312	775	782	777	774
295	312	316	317	313	777	784	779	776
296	357	361	362	358	863	870	864	862
297	313	317	318	314	779	786	780	778
298	356	360	361	357	861	868	863	860
299	314	318	359	355	780	865	859	857
300	355	359	360	356	859	866	861	858
301	256	261	262	257	676	684	678	675
302	257	262	263	258	678	686	680	677
303	258	263	264	259	680	688	682	679
304	259	264	265	260	682	690	683	681
305	260	265	311	307	683	773	767	765
306	307	311	312	308	767	774	769	766
307	308	312	313	309	769	776	771	768
308	353	357	358	354	855	862	856	854
309	309	313	314	310	771	778	772	770
310	352	356	357	353	853	860	855	852

311	310	314	355	351	772	857	851	849
312	351	355	356	352	851	858	853	850
313	251	256	257	252	667	675	669	666
314	252	257	258	253	669	677	671	668
315	253	258	259	254	671	679	673	670
316	254	259	260	255	673	681	674	672
317	255	260	307	303	674	765	759	757
318	303	307	308	304	759	766	761	758
319	304	308	309	305	761	768	763	760
320	349	353	354	350	847	854	848	846
321	305	309	310	306	763	770	764	762
322	348	352	353	349	845	852	847	844
323	306	310	351	347	764	849	843	841
324	347	351	352	348	843	850	845	842
325	246	251	252	247	658	666	660	657
326	247	252	253	248	660	668	662	659
327	248	253	254	249	662	670	664	661
328	249	254	255	250	664	672	665	663
329	250	255	303	299	665	757	751	749
330	299	303	304	300	751	758	753	750
331	300	304	305	301	753	760	755	752
332	345	349	350	346	839	846	840	838
333	301	305	306	302	755	762	756	754
334	344	348	349	345	837	844	839	836
335	302	306	347	343	756	841	835	833
336	343	347	348	344	835	842	837	834
337	241	246	247	242	649	657	651	648
338	236	241	242	237	640	648	642	639
339	242	247	248	243	651	659	653	650
340	243	248	249	244	653	661	655	652
341	244	249	250	245	655	663	656	654
342	245	250	299	295	656	749	743	741
343	295	299	300	296	743	750	745	742
344	296	300	301	297	745	752	747	744
345	341	345	346	342	831	838	832	830
346	297	301	302	298	747	754	748	746
347	340	344	345	341	829	836	831	828
348	298	302	343	339	748	833	827	825
349	339	343	344	340	827	834	829	826
350	237	242	243	238	642	650	644	641
355	238	243	244	239	644	652	646	643
356	239	244	245	240	646	654	647	645
357	240	245	295	291	647	741	735	733
358	291	295	296	292	735	742	737	734
359	292	296	297	293	737	744	739	736

```
 360  337  341  342  338  823  830  824  822
 361  293  297  298  294  739  746  740  738
 362  336  340  341  337  821  828  823  820
 363  294  298  339  335  740  825  819  817
 364  335  339  340  336  819  826  821  818
*ELEMENT, TYPE=CPE8, ELSET=FAR
   1  577  581  582  578 1284 1290 1286 1283
   2  578  582  583  579 1286 1291 1288 1285
   3  579  583  584  580 1288 1292 1289 1287
   4  580  584  600  597 1289 1326 1322 1320
   5  597  600  601  598 1322 1327 1324 1321
   6  598  601  602  599 1324 1328 1325 1323
   7  599  602  618  615 1325 1362 1358 1356
   8  634  637  638  635 1396 1400 1397 1395
   9  615  618  619  616 1358 1363 1360 1357
  10  633  636  637  634 1394 1399 1396 1393
  11  616  619  620  617 1360 1364 1361 1359
  12  617  620  636  633 1361 1398 1394 1392
  13  573  577  578  574 1277 1283 1279 1276
  14  574  578  579  575 1279 1285 1281 1278
  15  575  579  580  576 1281 1287 1282 1280
  16  576  580  597  594 1282 1320 1316 1314
  17  594  597  598  595 1316 1321 1318 1315
  18  595  598  599  596 1318 1323 1319 1317
  19  596  599  615  612 1319 1356 1352 1350
  20  631  634  635  632 1390 1395 1391 1389
  21  612  615  616  613 1352 1357 1354 1351
  22  630  633  634  631 1388 1393 1390 1387
  23  613  616  617  614 1354 1359 1355 1353
  24  614  617  633  630 1355 1392 1388 1386
  25  569  573  574  570 1270 1276 1272 1269
  26  570  574  575  571 1272 1278 1274 1271
  27  571  575  576  572 1274 1280 1275 1273
  28  572  576  594  591 1275 1314 1310 1308
  29  591  594  595  592 1310 1315 1312 1309
  30  592  595  596  593 1312 1317 1313 1311
  31  593  596  612  609 1313 1350 1346 1344
  32  628  631  632  629 1384 1389 1385 1383
  33  609  612  613  610 1346 1351 1348 1345
  34  627  630  631  628 1382 1387 1384 1381
  35  610  613  614  611 1348 1353 1349 1347
  36  611  614  630  627 1349 1386 1382 1380
  37  565  569  570  566 1263 1269 1265 1262
  38  566  570  571  567 1265 1271 1267 1264
  39  567  571  572  568 1267 1273 1268 1266
  40  568  572  591  588 1268 1308 1304 1302
  41  588  591  592  589 1304 1309 1306 1303
  42  589  592  593  590 1306 1311 1307 1305
  43  590  593  609  606 1307 1344 1340 1338
  44  625  628  629  626 1378 1383 1379 1377
  45  606  609  610  607 1340 1345 1342 1339
  46  624  627  628  625 1376 1381 1378 1375
  47  607  610  611  608 1342 1347 1343 1341
  48  608  611  627  624 1343 1380 1376 1374
  49  561  565  566  562 1256 1262 1258 1255
  50  562  566  567  563 1258 1264 1260 1257
  51  563  567  568  564 1260 1266 1261 1259
  52  564  568  588  585 1261 1302 1298 1296
  53  585  588  589  586 1298 1303 1300 1297
  54  586  589  590  587 1300 1305 1301 1299
  55  587  590  606  603 1301 1338 1334 1332
  56  622  625  626  623 1372 1377 1373 1371
  57  603  606  607  604 1334 1339 1336 1333
  58  621  624  625  622 1370 1375 1372 1369
  59  604  607  608  605 1336 1341 1337 1335
  60  605  608  624  621 1337 1374 1370 1368
  61  524  561  562  527 1251 1255 1252 1181
  62  527  562  563  530 1252 1257 1253 1187
  63  530  563  564  533 1253 1259 1254 1193
  64  533  564  585  536 1254 1296 1293 1199
  65  536  585  586  539 1293 1297 1294 1205
  66  539  586  587  542 1294 1299 1295 1211
  67  542  587  603  545 1295 1332 1329 1217
  68  557  622  623  560 1366 1371 1367 1247
  69  545  603  604  548 1329 1333 1330 1223
  70  554  621  622  557 1365 1369 1366 1241
  71  548  604  605  551 1330 1335 1331 1229
  72  551  605  621  554 1331 1368 1365 1235
*MPC
   9  193  641
   9   62  237
   9  189  639
   9   60  236
   9   64  238
   9  197  643
   9   66  239
   9  201  645
   9   68  240
   9  205  733
   9   70  291
   9  209  734
   9   72  292
   9  213  736
   9  233  822
   9   84  338
   9   74  293
   9  217  738
   9   82  337
   9  229  820
   9   76  294
   9  221  817
   9   80  336
   9   78  335
   9  225  818
*NSET,NSET=INTBOUN
193,62,189,60,64,197,66,201,68,205,70,209,72,213,
233,84, 74,217, 82,229, 76,221, 80, 78,225
*NSET,NSET=EXTBOUN
641,237,639,236,238,643,239,645,240,733,291,734,292,
736,822,338,293,738,337,820,294,817,336,335,818
*ELSET,ELSET=CLOSE,GENERATE
241,364,1
*NSET,NSET=LEFT
    1    2    3    4    5   58   83   84   85   87   89
   91  185  234  235  338
  342  346  350  354  358  362  366  370  374  378  445
  449  453  457  461  465
  469  518  519  520  521  558  559  560  623  626  629
  632  635  638  824  832
  840  848  856  864  872  880  888  896 1023 1031 1039
 1047 1055 1063 1071 1172
 1173 1174 1175 1248 1249 1250 1367 1373 1379 1385 1391
 1397
*NSET,NSET=BOTTOM
    5   10   15   20   25   30   35   40   45   46   59
   60   93  102  111  120
  129  138  147  156  161  186  187  236  241  246  251
  256  261  266  271  276
  281  286  379  384  389  394  399  404  409  470  471
  472  473  522  523  524
  561  565  569  573  577  581  640  649  658  667  676
  685  694  703  712  721
  901  907  916  925  934  943  952 1076 1077 1079 1081
 1176 1177 1179 1251 1256
 1263 1270 1277 1284
```

```
*NSET,NSET=TOP
638,637,636,620,619,618,602,1400,1399,1398,1364,1363,1362
*NSET,NSET=RIGHT
561,582,583,584,600,601,602,1290,1291,1292,1326,1327,1328
*ELSET,ELSET=INNER
NEAR,FIBER
*ELSET,ELSET=MATRIX
NEAR,FAR
*EQUATION
2
1400,2,1,638,2,-1
2
637,2,1,638,2,-1
2
1399,2,1,638,2,-1
2
636,2,1,638,2,-1
2
1398,2,1,638,2,-1
2
620,2,1,638,2,-1
2
1364,2,1,638,2,-1
2
619,2,1,638,2,-1
2
1363,2,1,638,2,-1
2
618,2,1,638,2,-1
2
1362,2,1,638,2,-1
2
602,2,1,638,2,-1
2
1328,1,1,602,1,-1
2
601,1,1,602,1,-1
2
1327,1,1,602,1,-1
2
600,1,1,602,1,-1
2
1326,1,1,602,1,-1
2
584,1,1,602,1,-1
```

```
2
1292,1,1,602,1,-1
2
583,1,1,602,1,-1
2
1291,1,1,602,1,-1
2
582,1,1,602,1,-1
2
1290,1,1,602,1,-1
2
581,1,1,602,1,-1
*BOUNDARY
LEFT,1,,0.
BOTTOM,2,,0.
*SOLID SECTION,ELSET=MATRIX,MATERIAL=MMATRIX
*SOLID SECTION,ELSET=FIBER,MATERIAL=MFIBER
*MATERIAL,NAME=MMATRIX
*ELASTIC,TYPE=ISO
200.0E3,0.3
*EXPANSION,ZERO=0.
0.0 ,0.
*MATERIAL,NAME=MFIBER
*ELASTIC,TYPE=ISO
100.0E3,0.3
*EXPANSION,TYPE=ORTHO,ZERO=0.
1.E-6, 1.E-6, 0.,0.
*INITIAL CONDITIONS,TYPE=TEMPERATURE
ALL,0.
*STEP,LINEAR
*STATIC
*TEMPERATURE
ALL,-1000.
*CLOAD
638,2,100.
*EL PRINT,ELSET=FIBER,POSITION=AVERAGED AT NODES
S
ENER
EE
*EL PRINT,ELSET=CLOSE,POSITION=AVERAGED AT NODES
S
ENER
EE
*NODE PRINT,NSET=ALL
U
```

```
*PLOT
def.conf. Em/Ef = 2 Misfit -1.e-3 Load = 100MPa
*DETAIL,ELSET=INNER
*DISPLACED
U
*PLOT
def.conf. Em/Ef = 2 Misfit -1.e-3 Load = 100MPa
*DETAIL,ELSET=FAR
*DISPLACED
U
*PLOT
stress Em/Ef = 2 Misfit -1.e-3 Load = 100MPa
*DETAIL,ELSET=INNER
*CONTOUR
S11
S22
S12
S33
*DETAIL,ELSET=FAR
*CONTOUR
S11
S22
S12
S33
*NODE FILE,NSET=ALL
U
*EL FILE
S
ENER
EE
*END STEP
```

APPENDIX V: TYPICAL ABAQUS INPUT FILE FOR THE ELASTIC ANALYSIS OF A $\gamma - \gamma'$ UNIT CELL

```
**
**       Abaqus 4.7  Input file for elastic analysis
**                   of gamma-gamma' unit cell.
**
*HEADING
Miyazaki,Nakamura,Mori alloy /tension.
*NODE,INPUT=16
*NODE,INPUT=17
*NSET,NSET=ALL,GENERATE
1,5002
*NSET,NSET=BOTTOM1,GENERATE
1,41,1
*NSET,NSET=BOTTOM2,GENERATE
1843,4597,81
*NSET,NSET=BOTTOM
BOTTOM1,BOTTOM2
*NSET,NSET=LEFT1,GENERATE
1,1641,41
*NSET,NSET=LEFT2,GENERATE
1763,4517,81
*NSET,NSET=LEFT
LEFT1,LEFT2
*NSET,NSET=INTBOUN,GENERATE
2978,3058
*NSET,NSET=EXTBOUN,GENERATE
3059,3139
*ELEMENT,TYPE=CGPE10R
1,1,3,85,83,2,44,84,42,5001,5002
*ELGEN,ELSET=CARTES
1,20,2,1,20,82,20
*ELEMENT,TYPE=CGPE10R
401,1641,1643,1846,1844,1642,1765,1845,1763,5001,5002
*ELGEN,ELSET=FLAYL
401,20,2
*ELEMENT,TYPE=CGPE10R,ELSET=FLAYR
421,1681,1599,1886,1884,1640,1805,1885,1803,5001,5002
422,1599,1517,1888,1886,1558,1807,1887,1805,5001,5002
423,1517,1435,1890,1888,1476,1809,1889,1807,5001,5002
424,1435,1353,1892,1890,1394,1811,1891,1809,5001,5002
425,1353,1271,1894,1892,1312,1813,1893,1811,5001,5002
426,1271,1189,1896,1894,1230,1815,1895,1813,5001,5002
427,1189,1107,1898,1896,1148,1817,1897,1815,5001,5002
428,1107,1025,1900,1898,1066,1819,1899,1817,5001,5002
429,1025, 943,1902,1900, 984,1821,1901,1819,5001,5002
430, 943, 861,1904,1902, 902,1823,1903,1821,5001,5002
```

```
431, 861, 779,1906,1904, 820,1825,1905,1823,5001,5002
432, 779, 697,1908,1906, 738,1827,1907,1825,5001,5002
433, 697, 615,1910,1908, 656,1829,1909,1827,5001,5002
434, 615, 533,1912,1910, 574,1831,1911,1829,5001,5002
435, 533, 451,1914,1912, 492,1833,1913,1831,5001,5002
436, 451, 369,1916,1914, 410,1835,1915,1833,5001,5002
437, 369, 287,1918,1916, 328,1837,1917,1835,5001,5002
438, 287, 205,1920,1918, 246,1839,1919,1837,5001,5002
439, 205, 123,1922,1920, 164,1841,1921,1839,5001,5002
440, 123,  41,1924,1922,  82,1843,1923,1841,5001,5002
*ELSET,ELSET=FLAYER
FLAYL,FLAYR
*ELEMENT,TYPE=CGPE10R
441,1844,1846,2008,2006,1845,1927,2007,1925,5001,5002
*ELGEN,ELSET=SLAYER
441,40,2,1,7,162,40
*ELEMENT,TYPE=CGPE10R
721,3059,3061,3223,3221,3060,3142,3222,3140,5001,5002
*ELGEN,ELSET=TLAYER
721,40,2,1,9,162,40
*ELSET,ELSET = ELEXBD,GENERATE
721,761
*ELSET,ELSET = ELINBD,GENERATE
681,720
*ELSET,ELSET = GAMMA,GENERATE
721,1080
*ELSET,ELSET = GAMMAP,GENERATE
1,720
*SOLID SECTION,ELSET=GAMMA,MATERIAL=MGAMMA
*SOLID SECTION,ELSET=GAMMAP,MATERIAL=MGAMMAP
*MATERIAL,NAME=MGAMMA
*ELASTIC, TYPE=ORTHO
112400.,62700.,112400.,62700.,62700.,112400.,56900.,56900.,
56900.,25.
*EXPANSION,ZERO=25.
0.0E-5,850.
*MATERIAL,NAME=MGAMMAP
*ELASTIC, TYPE=ORTHO
166600.,106500.,166600.,106500.,106500.,166600.,99200.,99200.,
99200.,25.
*EXPANSION,ZERO=25.
1.E-6,850.
*BOUNDARY
BOTTOM,2,,0.0
LEFT,1,,0.0
```

```
5002,1,2,0.0
*EQUATION
2
4518,2,1,4517,2,-1
2
4519,2,1,4517,2,-1
2
4520,2,1,4517,2,-1
2
4521,2,1,4517,2,-1
2
4522,2,1,4517,2,-1
2
4523,2,1,4517,2,-1
2
4524,2,1,4517,2,-1
2
4525,2,1,4517,2,-1
2
4526,2,1,4517,2,-1
2
4527,2,1,4517,2,-1
2
4528,2,1,4517,2,-1
2
4529,2,1,4517,2,-1
2
4530,2,1,4517,2,-1
2
4531,2,1,4517,2,-1
2
4532,2,1,4517,2,-1
2
4533,2,1,4517,2,-1
2
4534,2,1,4517,2,-1
2
4535,2,1,4517,2,-1
2
4536,2,1,4517,2,-1
2
4537,2,1,4517,2,-1
2
4538,2,1,4517,2,-1
2
4539,2,1,4517,2,-1
2
4540,2,1,4517,2,-1
2
4541,2,1,4517,2,-1
2
4542,2,1,4517,2,-1
2
4543,2,1,4517,2,-1
2
4544,2,1,4517,2,-1
2
4545,2,1,4517,2,-1
2
4546,2,1,4517,2,-1
2
4547,2,1,4517,2,-1
2
4548,2,1,4517,2,-1
2
4549,2,1,4517,2,-1
2
4550,2,1,4517,2,-1
2
4551,2,1,4517,2,-1
2
4552,2,1,4517,2,-1
2
4553,2,1,4517,2,-1
2
4554,2,1,4517,2,-1
2
4555,2,1,4517,2,-1
2
4556,2,1,4517,2,-1
2
4557,2,1,4517,2,-1
2
4557,1,1,4597,1,-1
2
4558,1,1,4597,1,-1
2
4559,1,1,4597,1,-1
2
4560,1,1,4597,1,-1
```

```
2
4561,1,1,4597,1,-1
2
4562,1,1,4597,1,-1
2
4563,1,1,4597,1,-1
2
4564,1,1,4597,1,-1
2
4565,1,1,4597,1,-1
2
4566,1,1,4597,1,-1
2
4567,1,1,4597,1,-1
2
4568,1,1,4597,1,-1
2
4569,1,1,4597,1,-1
2
4570,1,1,4597,1,-1
2
4571,1,1,4597,1,-1
2
4572,1,1,4597,1,-1
2
4573,1,1,4597,1,-1
2
4574,1,1,4597,1,-1
2
4575,1,1,4597,1,-1
2
4576,1,1,4597,1,-1
2
4577,1,1,4597,1,-1
2
4578,1,1,4597,1,-1
2
4579,1,1,4597,1,-1
2
4580,1,1,4597,1,-1
2
4581,1,1,4597,1,-1
2
4582,1,1,4597,1,-1
2
4583,1,1,4597,1,-1
2
4584,1,1,4597,1,-1
2
4585,1,1,4597,1,-1
2
4586,1,1,4597,1,-1
2
4587,1,1,4597,1,-1
2
4588,1,1,4597,1,-1
2
4589,1,1,4597,1,-1
2
4590,1,1,4597,1,-1
2
4591,1,1,4597,1,-1
2
4592,1,1,4597,1,-1
2
4593,1,1,4597,1,-1
2
4594,1,1,4597,1,-1
2
4595,1,1,4597,1,-1
2
4596,1,1,4597,1,-1
*EQUATION
2
INTBOUN,2,1,EXTBOUN,2,-1
2
INTBOUN,1,1,EXTBOUN,1,-1
*INITIAL CONDITIONS,TYPE=TEMPERATURE
ALL,0.0
*STEP,LINEAR
*STATIC
*TEMPERATURE
ALL,5000.
*CLOAD
4517,2,147.
*EL PRINT,ELSET=FLAYR
S
*NODE PRINT,NSET=INTBOUN
U
*PLOT
```

```
MNM TENSION
*DISPLACED
U
*PLOT
M/N/M  misfit and load
*CONTOUR
S11
S22
S33
S12
MISES
PRESS
*NODE FILE
U
*EL FILE
S
ENER
*END STEP
```

APPENDIX VI: TYPICAL ABAQUS INPUT FILE FOR THE ANALYSIS OF A STRESS-ANNEALING TRANSIENT FOR A $\gamma - \gamma'$ UNIT CELL

```
**
**     Abaqus 4.7  input file for analysis of stress annealing
**                 transient for a gamma-gamma' unit cell.
*HEADING
Miyazaki,Nakamura,Mori
*NODE,INPUT=16
*NODE,INPUT=17
*NSET,NSET=ALL,GENERATE
1,5002
*NSET,NSET=BOTTOM1,GENERATE
1,41,1
*NSET,NSET=BOTTOM2,GENERATE
1843,4597,81
*NSET,NSET=BOTTOM
BOTTOM1,BOTTOM2
*NSET,NSET=LEFT1,GENERATE
1,1641,41
*NSET,NSET=LEFT2,GENERATE
1763,4517,81
*NSET,NSET=LEFT
LEFT1,LEFT2
*NSET,NSET=INTBOUN,GENERATE
2978,3058
*NSET,NSET=EXTBOUN,GENERATE
3059,3139
*ELEMENT,TYPE=CGPE10R
1,1,3,85,83,2,44,84,42,5001,5002
*ELGEN,ELSET=CARTES
1,20,2,1,20,82,20
*ELEMENT,TYPE=CGPE10R
401,1641,1643,1846,1844,1642,1765,1845,1763,5001,5002
*ELGEN,ELSET=FLAYL
401,20,2
*ELEMENT,TYPE=CGPE10R,ELSET=FLAYR
421,1681,1599,1886,1884,1640,1805,1885,1803,5001,5002
422,1599,1517,1888,1886,1558,1807,1887,1805,5001,5002
423,1517,1435,1890,1888,1476,1809,1889,1807,5001,5002
424,1435,1353,1892,1890,1394,1811,1891,1809,5001,5002
425,1353,1271,1894,1892,1312,1813,1893,1811,5001,5002
426,1271,1189,1896,1894,1230,1815,1895,1813,5001,5002
427,1189,1107,1898,1896,1148,1817,1897,1815,5001,5002
428,1107,1025,1900,1898,1066,1819,1899,1817,5001,5002
429,1025, 943,1902,1900, 984,1821,1901,1819,5001,5002
430, 943, 861,1904,1902, 902,1823,1903,1821,5001,5002
431, 861, 779,1906,1904, 820,1825,1905,1823,5001,5002
432, 779, 697,1908,1906, 738,1827,1907,1825,5001,5002
433, 697, 615,1910,1908, 656,1829,1909,1827,5001,5002
434, 615, 533,1912,1910, 574,1831,1911,1829,5001,5002
435, 533, 451,1914,1912, 492,1833,1913,1831,5001,5002
436, 451, 369,1916,1914, 410,1835,1915,1833,5001,5002
437, 369, 287,1918,1916, 328,1837,1917,1835,5001,5002
438, 287, 205,1920,1918, 246,1839,1919,1837,5001,5002
439, 205, 123,1922,1920, 164,1841,1921,1839,5001,5002
440, 123,  41,1924,1922,  82,1843,1923,1841,5001,5002
*ELSET,ELSET=FLAYER
FLAYL,FLAYR
*ELEMENT,TYPE=CGPE10R
441,1844,1846,2008,2006,1845,1927,2007,1925,5001,5002
*ELGEN,ELSET=SLAYER
441,40,2,1,7,162,40
*ELEMENT,TYPE=CGPE10R
721,3059,3061,3223,3221,3060,3142,3222,3140,5001,5002
*ELGEN,ELSET=TLAYER
721,40,2,1,9,162,40
*ELSET,ELSET = ELEXBD,GENERATE
721,761
*ELSET,ELSET = ELINBD,GENERATE
681,720
*ELSET.ELSET = GAMMA,GENERATE
721,1080
*ELSET,ELSET = GAMMAP,GENERATE
1,720
*SOLID SECTION,ELSET=GAMMA,MATERIAL=MGAMMA
*SOLID SECTION,ELSET=GAMMAP,MATERIAL=MGAMMAP
*MATERIAL,NAME=MGAMMA
*ELASTIC, TYPE=ORTHO
112400.,62700.,112400.,62700.,62700.,112400.,56900.,56900.,
56900.,25.
*EXPANSION,ZERO=25.
0.0E-5,850.
*CREEP,LAW=STRAIN
5.4E-15,4.8,0.,850.
*MATERIAL,NAME=MGAMMAP
*ELASTIC, TYPE=ORTHO
166600.,106500.,166600.,106500.,106500.,166600.,99200.,99200.,
99200.,25.
*EXPANSION,ZERO=25.
1.E-6,850.
*BOUNDARY
BOTTOM,2,,0.0
```

```
LEFT,1,,0.0
5002,1,2,0.0
*EQUATION
2
4518,2,1,4517,2,-1
2
4519,2,1,4517,2,-1
2
4520,2,1,4517,2,-1
2
4521,2,1,4517,2,-1
2
4522,2,1,4517,2,-1
2
4523,2,1,4517,2,-1
2
4524,2,1,4517,2,-1
2
4525,2,1,4517,2,-1
2
4526,2,1,4517,2,-1
2
4527,2,1,4517,2,-1
2
4528,2,1,4517,2,-1
2
4529,2,1,4517,2,-1
2
4530,2,1,4517,2,-1
2
4531,2,1,4517,2,-1
2
4532,2,1,4517,2,-1
2
4533,2,1,4517,2,-1
2
4534,2,1,4517,2,-1
2
4535,2,1,4517,2,-1
2
4536,2,1,4517,2,-1
2
4537,2,1,4517,2,-1
2
4538,2,1,4517,2,-1
```

```
2
4539,2,1,4517,2,-1
2
4540,2,1,4517,2,-1
2
4541,2,1,4517,2,-1
2
4542,2,1,4517,2,-1
2
4543,2,1,4517,2,-1
2
4544,2,1,4517,2,-1
2
4545,2,1,4517,2,-1
2
4546,2,1,4517,2,-1
2
4547,2,1,4517,2,-1
2
4548,2,1,4517,2,-1
2
4549,2,1,4517,2,-1
2
4550,2,1,4517,2,-1
2
4551,2,1,4517,2,-1
2
4552,2,1,4517,2,-1
2
4553,2,1,4517,2,-1
2
4554,2,1,4517,2,-1
2
4555,2,1,4517,2,-1
2
4556,2,1,4517,2,-1
2
4557,2,1,4517,2,-1
2
4557,1,1,4597,1,-1
2
4558,1,1,4597,1,-1
2
4559,1,1,4597,1,-1
2
```

```
4560,1,1,4597,1,-1
2
4561,1,1,4597,1,-1
2
4562,1,1,4597,1,-1
2
4563,1,1,4597,1,-1
2
4564,1,1,4597,1,-1
2
4565,1,1,4597,1,-1
2
4566,1,1,4597,1,-1
2
4567,1,1,4597,1,-1
2
4568,1,1,4597,1,-1
2
4569,1,1,4597,1,-1
2
4570,1,1,4597,1,-1
2
4571,1,1,4597,1,-1
2
4572,1,1,4597,1,-1
2
4573,1,1,4597,1,-1
2
4574,1,1,4597,1,-1
2
4575,1,1,4597,1,-1
2
4576,1,1,4597,1,-1
2
4577,1,1,4597,1,-1
2
4578,1,1,4597,1,-1
2
4579,1,1,4597,1,-1
2
4580,1,1,4597,1,-1
2
4581,1,1,4597,1,-1
2
4582,1,1,4597,1,-1
2
4583,1,1,4597,1,-1
2
4584,1,1,4597,1,-1
2
4585,1,1,4597,1,-1
2
4586,1,1,4597,1,-1
2
4587,1,1,4597,1,-1
2
4588,1,1,4597,1,-1
2
4589,1,1,4597,1,-1
2
4590,1,1,4597,1,-1
2
4591,1,1,4597,1,-1
2
4592,1,1,4597,1,-1
2
4593,1,1,4597,1,-1
2
4594,1,1,4597,1,-1
2
4595,1,1,4597,1,-1
2
4596,1,1,4597,1,-1
*EQUATION
2
INTBOUN,2,1,EXTBOUN,2,-1
2
INTBOUN,1,1,EXTBOUN,1,-1
*AMPLITUDE,TIME=V,NAME=LOWER
0.0,0.0,1.0,1.0
*INITIAL CONDITIONS,TYPE=TEMPERATURE
ALL,0.0
*STEP
*STATIC,PTOL=1.
*TEMPERATURE
ALL,5000.
*EL PRINT,ELSET=FLAYR
S
*NODE PRINT,NSET=INTBOUN
U
```

```
*PLOT
deforemed conf. temperature
*DISPLACED
U
*PLOT
M/N/M  misfit only
*CONTOUR
S11
S22
S33
S12
MISES
PRESS
*NODE FILE
U
*EL FILE
S
ENER
*END STEP
*STEP,INC=60
*VISCO,PTOL=0.005,CETOL=7.0E-4
0.0001,5.0E4
*CLOAD,AMP=LOWER
4517,2,147.
*EL PRINT,ELSET=FLAYR,FREQ=60
S
*NODE PRINT,NSET=INTBOUN,FREQ=60
U
*PLOT,FREQ=5
M/N/M tension
*CONTOUR
MISES
S11
S22
CEEQ
*NODE FILE,FREQUENCY = 5
U
*EL FILE,FREQUENCY=5
S
ENER
*EL FILE,FREQUENCY=5,POSITION=AVERAGED AT NODES
CE
*END STEP
```