Title
A STUDY OF TRANSPORT PHENOMENA AND INTERFACE STABILITY DURING SOLIDIFICATION OF BINARY SOLUTIONS USING FRONT TRACKING FINITE ELEMENTS

Author
Tsai, H.-L.

Publication Date
1984-05-01

LBL-19680

A STUDY OF TRANSPORT PHENOMENA AND INTERFACE STABILITY DURING SOLIDIFICATION OF BINARY SOLUTIONS USING FRONT TRACKING FINITE ELEMENTS

H.-L. Tsai
(Ph.D. Thesis)

May 1984

cam

Center for Advanced Materials

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

LBL-19680

A STUDY OF TRANSPORT AND INTERFACE STABILITY DURING SOLIDIFICATION OF BINARY SOLUTIONS USING FRONT TRACKING FINITE ELEMENTS

Hai-Lung Tsai

Ph.D. Thesis

Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

May 1985

A STUDY OF TRANSPORT PHENOMENA AND INTERFACE STABILITY DURING SOLIDIFICATION OF BINARY SOLUTIONS USING FRONT TRACKING FINITE ELEMENTS

by

HAI-LUNG TSAI

ABSTRACT

A new numerical method using "front tracking" finite elements was developed to solve the multi-dimensional transient heat and mass transfer equations associated with the solidification of bindary solutions. The energy balance equation at the interface was not treated as a boundary condition, but rather as an independent equation whose solution gave the new position of interface. A special new method was developed by which the interface was tracked in time by two steps: first the magnitude of displacement and the normal direction were independently obtained for each node on the interface, then they were superimposed to determine the new interface position. The numerical method can incorporate realistic thermodynamic conditions on the interface (including the effects of interfacial tension) and can accommodate the non-isothermal as well as the irregular but smooth interface. A novel meshing system based on a systematic exponential gridding concept was developed to yield accurate temperatures in the solid and liquid phases and concentration distribution in the liquid.

This numerical method was then employed to study the transport phenomena as well as the morphological stability of a planar interface

during a solidification process in a saline solution. The transient temperature and concentration distributions in the solid and liquid regions for both planar and curved interfaces were calculated.

To study the stability of a planar interface, several numerical perturbations in space and/or time were performed on either the outer boundary or the interface to simulate various physical effects. The results indicate that for the analyzed conditions, a temperature perturbation on the outer surface of the domain cannot generate the instability of the moving interface, even in a situation in which the solute in front of the interface is constitutionally supercooled. Furthermore, it appears that an increased solute concentration on the interface has a stabilizing effect. These results, obtained through rigorous analysis, contradict the existing interface morphology stability criteria based on concepts of equilibrium thermodynamics prevalent in the technical literature. The new results indicate the importance of the transient dynamic effects in the study of solid-liquid interface stability criteria. These effects have been neglected in previous work.

The numerical study was also used to investigate the effects of concentration perturbations on the solid-liquid interface on the stability of the interface. It was shown that a continuous concentration perturbation can lead to an unstable interface. These results demonstrate the importance of the new numerical method in the study of solidification processes and indicate the need for future studies to promote a fundamental understanding of the physical phenomena associated with the perturbed growth of a solid-liquid interface during solidification.

Boris Rubinsky

Chairman, Thesis Committee

Dedicated, with love, to my parents.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Professor Boris Rubinsky for his guidance throughout this research. As one of his first Ph.D. students, I have had the privilege of obtaining his advice both in this study and in many non-academic matters. To him, I am always in debt.

I would like to thank Professor Eugene E. Haller for his encouragement and support during this study. I also thank Professors Ralph Greif and Donald R. Olander for carefully reviewing the manuscript and making valuable comments.

Support for this research was given by the National Science Foundation under Grant No. DEM-8105916 and by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Sciences Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098, and is gratefully acknowledged.

I wish to thank my wife, Chiu-Liang, for her love, help and understanding during this study.

NOMENCLATURE

$[A_i]$	constant matrices, i = 1,2, and 3; Eq. (3.60)
A	amplitude of temperature perturbation, Eq. (4.1)
A'	amplitude of temperature perturbation, Eq. (4.2)
$[B]$	temperature gradient interpolation matrix, Eq. (3.4)
B	amplitude of concentration perturbation, Eq. (4.3)
$[C]$	conductance matrix, Eq. (3.10)
C	solute concentration
C_S, C_1	solute concentration of the solid phase
C_L, C_2	solute concentration of the liquid phase
C_i, C_∞	initial solute concentration in the liquid phase
C_0	solute concentration on the interface
c	thermal capacity
c_S	thermal capacity of the solid phase
c_L	thermal capacity of the liquid phase
D	mass diffusivity of the liquid phase
$f[\ldots]$	Newton's divided difference, Eq. (3.62)
$F(\omega)$	defined in Eq. (1.21)
$\{g\}$	defined in Eq. (3.40)
$G(\omega)$	defined in Eq. (1.20)
G_C	concentration gradient of liquid phase at the interface
G_L	temperature gradient of liquid phase at the interface
G_S	temperature gradient of solid phase at the interface
h	convection heat transfer coefficient

[I]	identity matrix
[J]	Jacobian transformation matrix
$\|J\|$	Jacobian of transformation
[K]	defined in Eq. (3.10)
$[K_i]$	i = c, h, and r; defined in Eq. (3.10)
$k = C_S/C_L$	partition coefficient
k_S, k_1	thermal conductivity of the solid phase
k_L, k_2	thermal conductivity of the liquid phase
L	latent heat of fusion
m	slope of liquidus line
[N]	displacement interpolation vector
{dn}	displacement vector
$\vec{n}$	normal direction of the interface
$\vec{n}_1$	normal direction of the solid boundary
$\vec{n}_2$	normal direction of the liquid boundary
P	defined as P = 1 - k
$P_n(x)$	interpolation polynomial, Eq. (3.61)
q	heat flux, Eq. (3.8)
q_r	radiation heat flux, Eq. (3.8)
{R}	defined in Eq. (3.10)
$\{R_i\}$	i = t, q, h, r; defined in Eq. (3.10)
r-z	Cartesian coordinates
$S(\omega)$	defined in Eq. (1.18)
S(t)	moving phase interface
S_i	i = 1, 2, 3, and 4; part of the moving interface, Eq. (3.8)
{T}	temperature matrix, defined in Eq. (1.18)

T_S, T_1	temperature of the solid phase
T_L, T_2	temperature of the liquid phase
T_i, T_∞	initial temperature of the liquid phase
T_m	melting point of pure liquid in plane interface
T_0	temperature on the interface
t	time
Δt_{cr}	critical time step in numerical stability analysis
$[U]$	matrix of directional cosine of elements on the interface
$\{v\}$	defined in Eq. (3.37)
V	constant interfacial velocity
V_n	interfacial normal velocity
W_i, W_j	Gauss weights
$W(\tau)$	weighting function
Δx	mesh size in the x direction
x-z	Cartesian coordinates
α_S, α_1	thermal diffusivity of the solid phase
α_L, α_2	thermal diffusivity of the liquid phase
α_i	i = 1, 2, 3; coefficients in the boundary condition, Eq. (2.8)
ξ-η	natural or local coordinates
ω	wave frequency, Eq. (1.16)
ω_C	defined in Eq. (A.13)
ω_L	defined in Eq. (A.17)
ω_S	defined in Eq. (A.18)
$[\phi]$	interpolation matrix
ϕ_i	interpolation functions

$\phi(x,t)$	perturbed interface, Eq. (1.16)
$[\Phi]$	defined in Eq. (3.36)
λ_i	eigenvalues
δ	amplitude of perturbation, Eq. (1.16)
ξ_S	averaging thermal conductivity of the solid phase, Eq. (A.31)
ξ_L	averaging thermal conductivity of the liquid phase, Eq. (A.31)
Ω	sum of the solid and liquid domains
$\Omega_1(t)$	solid domain
$\Omega_2(t)$	liquid domain
$\partial\Omega_1(t)$	total boundary of the solid domain
$\partial\Omega_2(t)$	total boundary of the liquid domain
$\Gamma_1(t)$	boundary of the solid domain, excluding the interfacial boundary S(t)
$\Gamma_2(t)$	boundary of the liquid domain, excluding the interfacial boundary S(t)
γ	mean interfacial curvature
$[\Lambda]$	defined in Eq. (3.39)
σ	Stefan-Boltzmann constant
ε	radiation emissivity
θ	defined in Eq. (3.21)
ρ	density
ρ_S	density of the solid phase
ρ_L	density of the liquid phase
R	radius of interfacial curvature

LIST OF FIGURES

LIST OF FIGURES (cont'd)

LIST OF FIGURES (cont'd)

LIST OF TABLES

CHAPTER 1:

INTRODUCTION

1.1 SOLIDIFICATION OF SOLUTIONS

1.1.1 Applications and Difficulties

Many practical problems in applied science and engineering involve the solidification of solutions, e.g., ice-making, food processing, medicine, crystallography, metallurgy, welding, and many others. Because the properties of the solid phase are not only inherited from its liquid phase, but significantly influenced by the details of the solidification process, a fundamental understanding of the physical phenomena occurring during solidification processes is essential to obtain the desired structure of the solid phase [1-4]. For example, properties of alloys such as strength, toughness, corrosion resistance, etc. are dependent in part on the degree of local segregation resulting from the spacing of dendrite arms in the alloys. By properly controlling the solidification procedures, properties of the alloy can be improved. Of course, the alloy properties can be modified by subsequent heat treatment, but the existing defects cannot in general be completely removed by these additional costly procedures. An important application of current solidification technology may be found in the crystal growth of electronic materials, which requires an understanding of solidification processes [5,6].

From a theoretical point of view, the analysis of solidification of solutions involves the simultaneous solution of transient heat and mass transfer equations in both time-dependent solid and liquid domains. The difficulties of studying solidification stem in part from the non-linearities associated with the transient position of the interface that separates the two phases with quite different properties. The interface is neither fixed in space nor is its motion known *a priori*; nevertheless, it is part of the solution. Furthermore, the governing heat and mass transfer equations in both phases are nonlinearly coupled at the unknown moving interface.

The most difficult aspect in the study of solidification processes is the establishment of a valid mathematical model that fully describes the real physical phenomena. An incomplete knowledge of, for example, anisotropic interfacial free energy, interfacial structure and kinetics (molecular attachment), etc. and their roles in determining the interface stability leads to difficulties in obtaining the correct governing equations. The phenomena on a molecular level occurring near the interface are intimately related to the fundamental physics of the materials. Although the basic principle of interface instability is partially answered by the supercooling effect, the dendritic shape, dendritic arm spacing, and sidebranching can hardly be predicted. Hence, a complete theoretical simulation of interface time evolution, from a plane transition to instability, to a complete dendrite is still beyond the scope of present technology.

Experimental studies on solidification of solutions are limited to phenomena observations and the empirical correlations among such parameters as the degree of supercooling, dendrite growing velocity, dendrite arm spacing, etc. [7-17]. These results may have practical importance and value, but do not make significant contributions to the fundamental understanding of solidification processes. It is noted that the comparison between experimental results and those predicted by the over-simplified theories are not adequate. Direct measurements of interfacial free energy and temperatures and solute distributions. especially around the dendrite, are complicated by the moving, microscopic size, and complicated geometry of dendrites.

1.1.2 Thermal and Solute Redistributions

A simple example will be given to illustrate the physical phenomena occurring during solidification in solutions. Extensions to the more general cases will be discussed thereafter. The example is for the semi-infinite domain shown in Fig. 1.1. The media is a bindary saline solution, at a uniform initial temperature T_i and concentration C_i. It is assumed that T_i is higher than the equilibrium temperature corresponding to C_i, i.e., the solution is not supercooled. Adiabatic boundary conditions are assumed on both upper and lower sides of the domain. At time $t = 0$ the cooling process is initiated by suddenly imposing a constant temperature T_∞, which is less than T_i, on the left outer surface of the domain. If T_∞ is below the freezing point of the solution, the solidification process will start from the left and the solid-liquid interface will move to the right of the domain. This process is called "unidirectional solidification." It is noted that the solution freezing temperature is determined by the amount of solute contained in the solution. In general, the liquid phase and its solidified phase possess

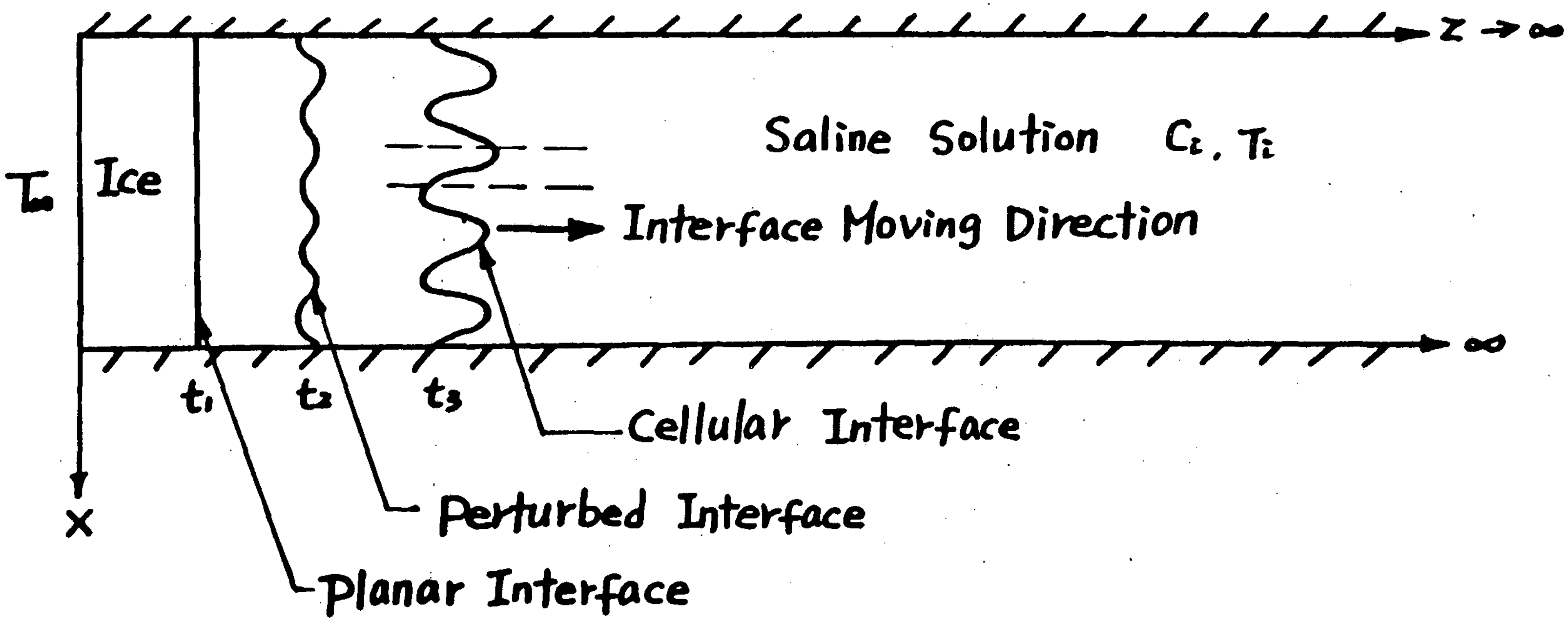


FIGURE 1.1. A unidirectional solidification process and interface time evolution, $t_3 > t_2 > t_1$.

quite different thermodynamic, chemical, and physical properties. Problems involving solidification or melting usually are referred to as "phase change," "moving boundary," "free boundary," or "Stefan" problems.

The temperature difference between the outer surface temperature and the interfacial temperature serves as the driving thermal force for advancing the interface. The sensible heat of the solid and the latent heat released by the freezing process are transported through the frozen solid layer by conduction, then rejected into the environment. The sensible heat from the liquid solution will also be transported by conduction and/or convection to the interface and then conducted through the solid region. Hence, the rate of solidification is governed in part by the rate at which the latent heat of fusion generated at the interface can be removed.

According to the phase diagram for solution, the liquid phase and its solid phase will not contain the same amount of solute. Hence, during the solidification of solutions, solute will be partially rejected or incorporated by the solid according to the phase diagram. A partition coefficient takes a value varying from zero (corresponding to the complete rejection of solute by the solid phase) to one (corresponding to the complete incorporation of solute into the solid phase). Usually the solid phase will contain less solute than the liquid phase, as indicated by the negative slopes of the solidus and liquidus lines in the phase diagram (see Fig. 1.2). The accumulation of rejected solute in the change of phase interface will lower the change of phase temperature. Thus the rate of solidification for solution is determined by the heat transfer

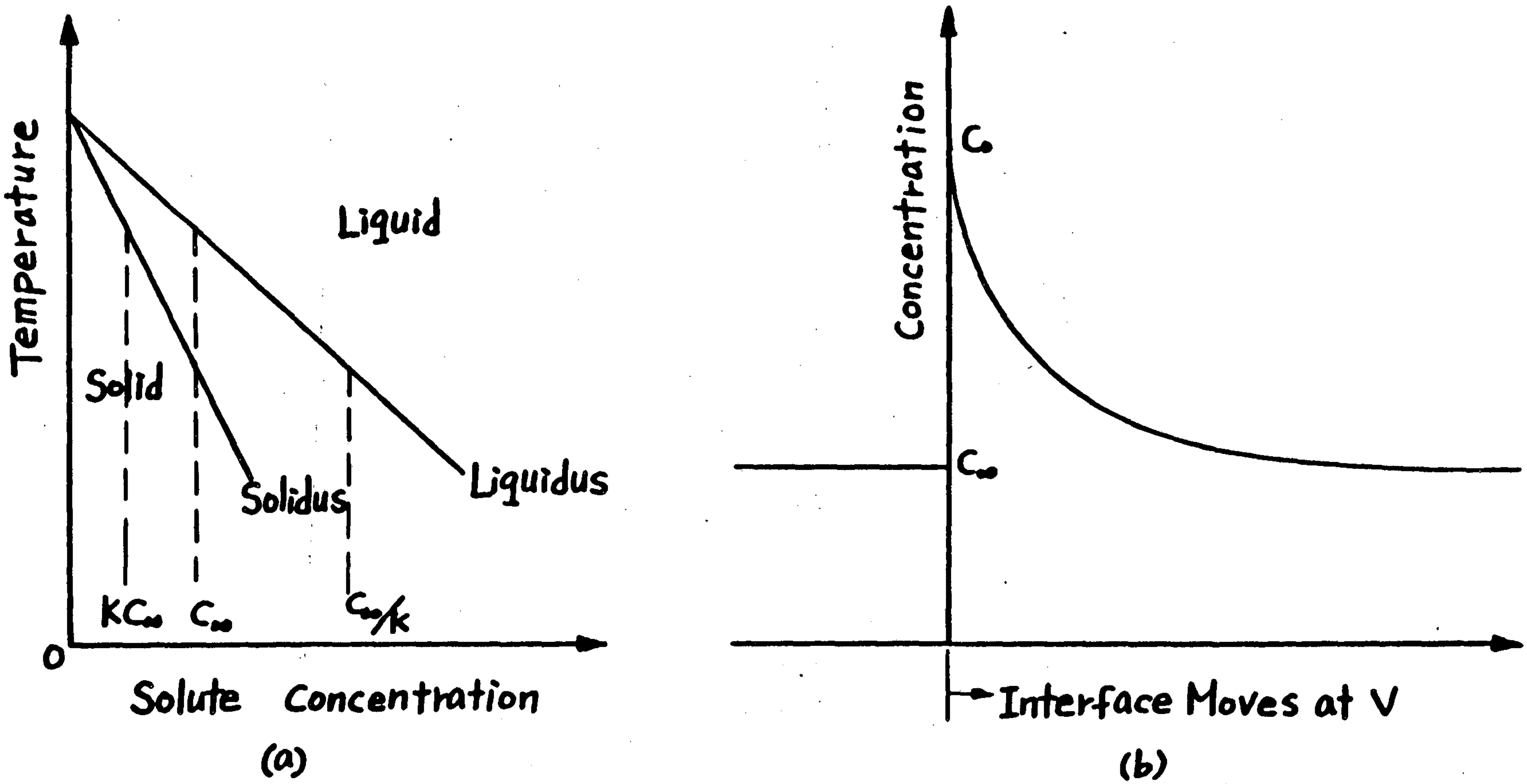


FIGURE 1.2. (a) Part of phase diagram, $k < 1$. (b) Concentration distribution in a steady-state solidification process.

process as well as the mass transfer process [18-30]. Due to the low mass diffusion coefficient in the liquid phase, the rejected solute will form a thin solute-rich layer in front of the phase interface. This is called the "concentration boundary layer." This phenomenon of solute segregation will cause the change of phase interface to become unstable according to the various interface stability theories, discussed in Section 1.2.

Many experiments have found that during the solidification of solutions, the solid-liquid interface is initially planar, but after some time the planar interface will become unstable [31-33,38]. The time evolution of the interface is also shown in Fig. 1.1. It is seen that at the onset of the process a small sinusoidal protuberance appears on the interface. This perturbed interface will then grow outward and become "finger-like" dendrites. This process is called cellular or dendritic growth. Although the name dendrite originates from the Greek word for "tree," we will use it in a more general sense. The shape of the dendrites depends greatly on the properties of the solution and the freezing conditions. In fact, finger-like, tree-like, and many other extremely complex geometries of dendrites have been observed. The dendrite is a microstructure with sizes ranging from a few to approximately 100 μm in diameter. The tips of dendrites are growing faster than the body and eventually neighboring dendrites will merge. Most of the rejected solute will be trapped between dendrites. The dendrite is a more stable form of interface. It is noted that the phenomena of interface instability and dendritic growth are only found when the liquid to be frozen contains solutes or is initially supercooled. For pure liquids that are not supercooled, the interface is always stable and planar.

A general solidification process of multi-component solutions involves the same basic phenomena as described above. Having many kinds of solutes present with the solution will further complicate the solute redistribution process and may result in the inter-reaction among these solutes. Hence, analyzing a uni-directional solidification process in binary solutions provides us with an optimal opportunity for the study of the fundamental characteristics of solidification processes in solutions and solid-liquid interface morphology [28-30, 32-36].

1.2 SOLID-LIQUID INTERFACE STABILITY

1.2.1 Constitutional Supercooling

The interface stability theories will be reviewed in detail here because later work will make extensive reference to this section. Chapter 5 will include commentary on the assumptions from which the theories were derived.

Consider the uni-directional solidification of a binary alloy similar to that in Fig. 1.1, having a partition coefficient k which is assumed constant and less than unity. The relevant part of the phase diagram is shown in Fig. 1.2. The melt is initially at uniform solute concentration C_∞. The temperature boundary conditions are varied in such a way that the solid-liquid interface advances at a constant velocity V. Steady-state temperature and concentration profiles are assumed in a frame of reference moving with the interface. This signifies that the rate of solute rejection by the solid during solidification will exactly equal the diffusion rate of solute away from the interface into the liquid. The composition profiles remain constant relative to the interface and are shown in Fig. 1.2. The solute distribution can be determined by solving the diffusion equation in a moving coordinate frame of reference attached to the interface:

$$D \frac{\partial^2 C_L}{\partial z^2} + V \frac{\partial C_L}{\partial z} = 0 \quad , \tag{1.1}$$

where z is the distance from the interface, D is the solute diffusivity, and V is the constant velocity of the interface. The boundary conditions are:

$$C_L = C_\infty/k = C_0 = C_S/k \qquad \text{at} \quad z = 0$$

$$C_L \rightarrow C_\infty \qquad \text{at} \quad z \rightarrow \infty \ .$$

The solution can be obtained easily as:

$$C_L = C_\infty \left[1 + \left(\frac{1-k}{k}\right) \exp\left(-Vz/D\right)\right] . \tag{1.2}$$

Since

$$C_0 = C_\infty \left(1 + \frac{1-k}{k}\right) \tag{1.3}$$

$$G_C = C_\infty \left(\frac{1-k}{k}\right)\left(-\frac{V}{D}\right) = \left(-\frac{V}{D}\right)(C_L - C_S) \quad , \tag{1.4}$$

hence

$$C_L = C_0 + \frac{DG_C}{V} \left[1 - \exp\left(-Vz/D\right)\right] , \tag{1.5}$$

where G_C is the concentration gradient in the liquid at the interface, C_0 is the solution concentration at the interface, and k is the partition coefficient, $k = C_S/C_L$.

In addition to the solute distribution derived in Eq. (1.2), it also is necessary to know the thermal diffusion fields of both liquid and solid. The governing equations are similar to that of Eq. (1.1) for temperatures in the liquid and solid states.

$$\alpha_L \frac{\partial^2 T_L}{\partial z^2} + V \frac{\partial T_L}{\partial z} = 0 \tag{1.6}$$

$$\alpha_S \frac{\partial^2 T_S}{\partial z^2} + V \frac{\partial T_S}{\partial z} = 0 \ , \tag{1.7}$$

where α_L and α_S are the thermal diffusivities of the liquid and solid, respectively. The boundary conditions are $T_S = T_L = T_0$ at the interface $z = 0$; the temperature gradients in the interface are given by G_L for liquid and G_S for solid at $z = 0$. With these boundary conditions, the solutions of Eqs. (1.6) and (1.7) are:

$$T_L = T_0 + \frac{\alpha_L G_L}{V} \left[1 - \exp\,(-Vz/\alpha_L)\right] \tag{1.8}$$

$$T_S = T_0 + \frac{\alpha_S G_S}{V} \left[1 - \exp\,(-Vz/\alpha_S)\right] \ , \tag{1.9}$$

where T_0 is the equilibrium temperature common to solid and liquid at the interface. Notice that Eqs. (1.5), (1.8), and (1.9) have exactly the same form.

Close to the moving interface, the temperature distribution can be expanded in a Maclaurin Series to yield

$$T_L = T_0 + G_L z \tag{1.10}$$

$$T_S = T_0 + G_S z \quad . \tag{1.11}$$

The energy balance on the interface is given by:

$$k_S G_S - k_L G_L = LV \quad , \tag{1.12}$$

where L is the latent heat of fusion per unit volume, and k_S and k_L are the thermal conductivities of solid and liquid, respectively.

To study interface stability, the actual temperature distributions and the equilibrium liquid temperature corresponding to the actual concentration are plotted in Fig. 1.3. The equilibrium temperature is directly related to the solute distribtuion in Eq. (1.5) by means of the phase diagram. From Fig. 1.3, it is seen that there is a region in the liquid near the interface that is supercooled, i.e., the equilibrium phase transformation temperature for the specific concentration at a given location is above the actual temperature. It is noted that in the supercooled region near the interface, the degree of supercooling increases in a direction away from the interface. Thus, if any part of the interface is perturbed and the "tip" of the protuberance enters the increasing supercooled region, it will grow faster and the interface will become unstable.

The above phenomenon is known as "constitutional supercooling" instability. The word constitutional indicates that the supercooling arises from a change in composition. Constitutional supercooling as a cause for interface instability was first proposed by Chalmers and co-workers in 1953 [37], in conjunction with experiments in directional

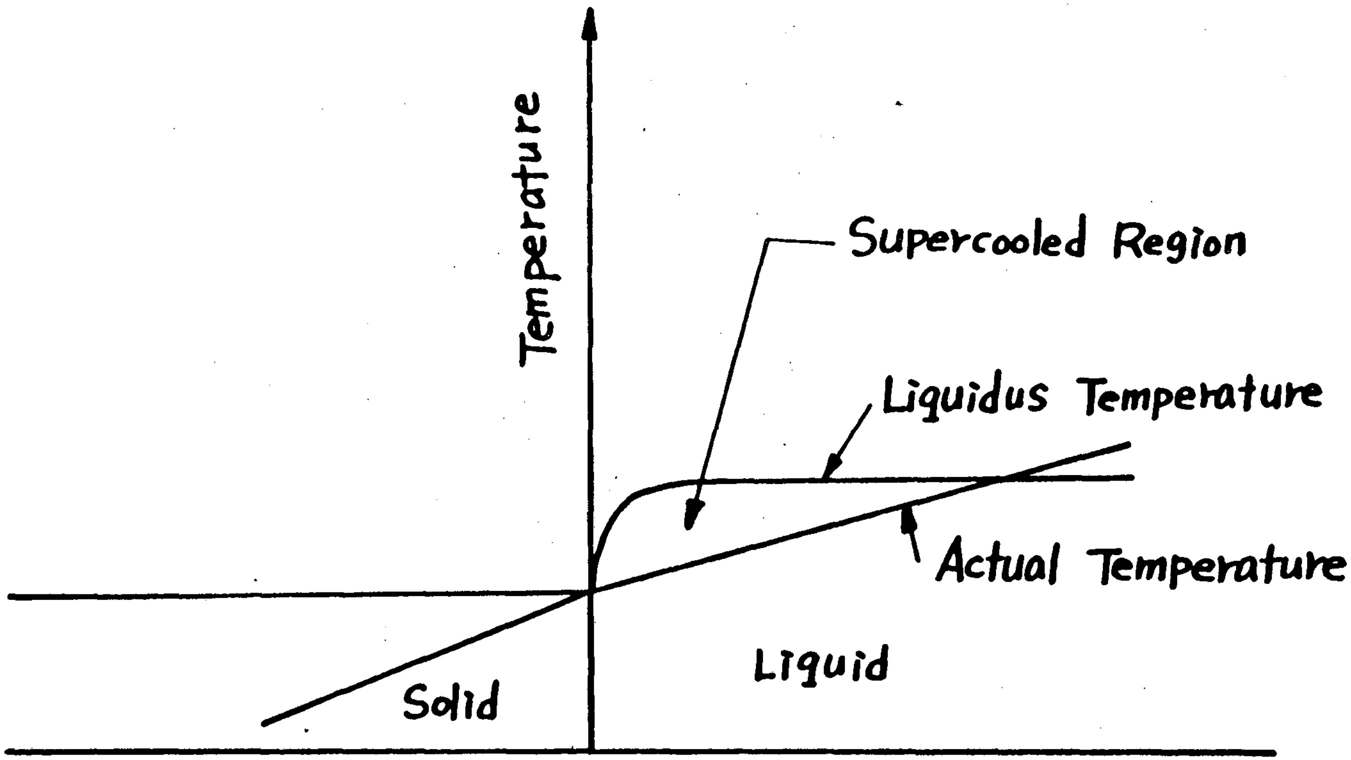


FIGURE 1.3. Constitutional supercooling near a steady-state interface.

crystallization of dilute tin alloys. In order for constitutional supercooling instability to occur, the gradient of the equivalent liquidus temperature curve on the change of phase interface (see Fig. 1.3) must be larger than the actual temperature gradient in the liquid on the interface. Thus, if the equilibrium temperature is given in terms of the concentration by

$$T_e = T_m + mC_L \quad , \tag{1.13}$$

where T_m is the pure liquid melting point and m is the liquidus slope (assumed to be constant), then constitutional supercooling implies that for an unstable interface,

$$\frac{G_L}{G_e} = \frac{G_L}{mG_C} = \frac{G_L}{mC_\infty \left(\frac{1-k}{k}\right)\left(-\frac{V}{D}\right)} < 1 \tag{1.14}$$

or

$$\frac{G_L}{V} < \frac{1-k}{k}\left(\frac{mC_\infty}{D}\right) , \tag{1.15}$$

where G_C is the concept of concentration gradient at the interface given in Eq. (1.4).

In summary, the constitutional supercooling was derived under the following assumptions:

(1) The planar interface is moving at a constant velocity in the z-direction.

(2) The domain is one-dimensional and semi-infinite in the z positive direction.

(3) The interface is sharp and infinitesimally thin.

(4) The interface is in thermodynamic equilibrium.

(5) The interfacial kinetics are negligible.

(6) A moving coordinate is used and attached on the interface such that $z = 0$ marks the plane of interface.

(7) The temperature and concentration distributions of both solid and liquid are steady-state in the moving coordinate system.

(8) There is no convection in the liquid, and no change of density in the liquid during freezing.

(9) The solute diffusion in the solid is negligible.

(10) All the material properties of solid and liquid are constant.

(11) The partition coefficient $k = C_S/C_L$ is constant over the range of interest.

Despite the initial assumptions made to develop the constitutional supercooling criterion, Eq. (1.14), this criterion can also be obtained in the absence of assumptions (1), (2), (6), and (7). Thus, the concept of constitutional supercooling as it was originally proposed is applicable to the multi-dimensional transient solidification processes.

1.2.2 Mullins-Sekerka Criterion

The constitutional supercooling criterion was based on a static analysis that showed that constitutional supercooling could be a source of interface instability. The same problem was considered in a dynamic and more rigorous analysis by Mullins and Sekerka (M-S). The same assumptions were made as those in the constitutional supercooling theory. The M-S criterion also considered the diffusion of heat and solute around the perturbed interface, and the interfacial free energy.

The linear perturbation analysis was first employed to solve this problem by Mullins and Sekerka in 1964; however, thereafter numerous extensions have been added [38-47]. In the M-S analysis a steady-state system was initially assumed, similar to that described in the previous section. At time $t = 0$, a small perturbation was imposed on the system, and all equations and boundary conditions were linearized with respect to this perturbation. The time dependence of the amplitude of perturbation was then investigated. In general, if the perturbation is growing in time, the interface is unstable. On the other hand, if the perturbation decays to zero, the interface is stable. It is noted that for any kind of initial disturbances, the final stability criterion of the interface should be the same according to the general theory of stability. The disturbance can be in the temperature, concentration, or interfacial shape.

Because most functions can be represented by a Fourier series of sinusoidal functions, sinusoidal perturbations of all possible wavelengths can be considered. Only if the rate of growth of the perturbation is negative for all wavelengths is the interface considered stable.

A sinusoidal perturbation of very small amplitude δ to the planar interface in the constant moving coordinate can be described by:

$$z = \phi(x,t) = \delta(t) \sin \omega x \quad , \tag{1.16}$$

where $\omega = 2\pi/\lambda$ is the wave frequency. Notice that a two-dimensional model is assumed.

Following the work of Mullins and Sekerka (the detailed procedures are outlined in Appendix 1), the interface stability criterion is:

$$\frac{\dot{\delta}}{\delta} = \frac{V\left[-2T_m\Gamma\omega^2\left(\omega_C - \frac{V}{D}P\right) - (\xi_S + \xi_L)\left(\omega_C - \frac{V}{D}P\right) + 2mG_C\left(\omega_C - \frac{V}{D}\right)\right]}{(\xi_S - \xi_L)\left(\omega_L - \frac{V}{D}P\right) + 2\omega mG_C} \tag{1.17}$$

Upon examining the physical significance of this interface stability criterion, the time evolution of a perturbation of wavelength $(2\pi/\omega)$ within the region of applicability of the model used becomes evident. It is apparent that a positive value of $\dot{\delta}/\delta$ for any ω means the flat interface is unstable, whereas a negative sign for all ω indicates a decaying perturbation and a stable interface. It is noted that the sign of Eq. (1.17) depends only on the sign of the numerator, since both terms in the denominator are always positive, The term $(\xi_S - \xi_L)$ is proportional to V as shown by Eq. (A.34), and is positive. Notice that $\omega_C > V/D > VP/D$, which results from the definition of ω_C [Eq. (A.13)] and from the fact that $p = 1-k < 1$. Therefore, the value $\omega_C - (V/D)P$ is positive. The second term of the denominator, $2\omega mG_C$, also is positive because both m and G_C have the same sign. Therefore, we expect the instability or stability of a planar interface to depend only on the sign of the numerator. Dividing the numerator through by a positive value $2[\omega_C - (V/D)P]V$ gives

$$S(\omega) = -T_m\Gamma\omega^2 - \tfrac{1}{2}(\xi_S + \xi_L) + \frac{mG_C[\omega_C - (V/D)]}{\omega_C - (V/D)P} \quad . \tag{1.18}$$

The frequency dependence function $S(\omega)$ must be negative for stability. The first term in Eq. (1.18) arises from the capillarity, which is always negative and has a stabilizing influence for all wavelengths. Furthermore, a shorter wavelength (large ω) favors stability. This is exactly the sort of stabilizing effect that would be expected of the surface tension. The second term, representing the temperature gradients, also is stabilizing for positive values. The third term represents the effect of solute accumulation, favors instability, and always is positive. Instability occurs when there is any frequency for which the magnitude of the third term is larger than that of the sum of the first two terms.

It is possible to simplify the M-S criterion and recover the result of the constitutional supercooling theory derived earlier by removing the capillary effects and the dependence of the stability criterion on the wavelength. Under the simplification mentioned above, the M-S criterion for instability can be expressed by:

$$mG_C > \tfrac{1}{2}(\xi_S + \xi_L) \quad . \tag{1.19}$$

The above criterion is essentially the same as the constitutional supercooling criterion, Eq. (1.14), except that a mean value of G_L and G_S, weighted by the thermal conductivities of these two phases, is substituted for G_L in Eq. (1.14). It is noted that the stability criterion Eq. (1.17) has a greater region of stability than Eq. (1.14) or (1.19).

The M-S stability criterion can be written in a slightly different form by separating out the wavelength-dependent and -independent part of $S(\omega)$. Define

$$G(\omega) = \frac{-T_m\Gamma\omega^2}{mG_C} + F(\omega) \quad , \tag{1.20}$$

where

$$F(\omega) = \frac{\omega_C - (V/D)}{\omega_C - (V/D)P} \quad . \tag{1.21}$$

Then the condition for stability becomes

$$\tfrac{1}{2}(\xi_S + \xi_L) > mG_C G(\omega) \quad . \tag{1.22}$$

$G(\omega)$ is composed of two parts. The first one is proportional to ω^2. The second, $F(\omega)$, is proportional to ω^2 for small ω and tends to unity for large ω, since Eq. (1.22) must be valid for all ω for the interface to be stable. In other words, the general condition for stability must satisfy

$$\tfrac{1}{2}(\xi_C + \xi_L) > mG_C G(\omega)_{max} \quad . \tag{1.23}$$

Notice the constitutional supercooling criterion corresponds to $G(\omega)_{max} = 1$. In fact, $G(\omega)_{max}$ has a maximum possible value of unit and will be lower in general. Thus the constitutional supercooling is a necessary but not sufficient condition for instability; the degree of constitutional supercooling must exceed some specific value.

1.3 PURPOSE OF PRESENT STUDY

The major purpose of this work are listed below:

(1) A major difficulty in analyzing solidification processes in solutions is the lack of analytical tools to study the process. The major purpose of this work is to develop a new numerical method to solve the multi-dimensional transient heat and mass transfer equations in the solid and liquid phases during solidification of binary solutions. The finite element computer program developed is capable of handling solidification processes with a non-isothermal phase interface and with geometrically irregular but smooth shapes of the

interface. Currently, there are no other numerical or analytical methods with these capabilities.

(2) The numerical method was used in studying a transient solidification process in a binary solution. Temperature and concentration distributions in both the solid and liquid phases were obtained during the arbitrary transient solidification process. The solute accumulation ahead of a planar interface and around a curved interface were observed. The results of this part of the study, although anticipated, constituted the first rigorous proof of the phenomena.

(3) The new computer code offers a unique method of studying interface stability phenomena during solidification in solutions. A mathematical model describing a typical solidification process in solutions was established. Then the planar interface stability was studied for several types of perturbation. Surprising and unique results conflicting with previous stability criteria were obtained for the effects of various parameters on interface stability. These results illustrate the importance of the new numerical method and indicate the need for new fundamental studies on the phenomena associated with the interface stability. A discussion of these results is included herein.

Following the introduction in Chapter 1, the mathematical model for typical solidification processes in solution is formulated in Chapter 2. A new general numerical method using front-tracking finite elements that was developed to solve the mathematical model is discussed in Chapter 3. In Chapter 4 this numerical method is employed to study the planar interface stability problems in solutions. The results are presented and discussed in Chapter 5, followed by conclusions in Chapter 6.

CHAPTER 2:

MATHEMATICAL MODEL

2.1 A GENERAL SOLIDIFICATION PROBLEM

Consider a general solidification process in binary solutions at some specific time t as shown in Fig. 2.1. There is a constant domain Ω that contains two time-dependent subdomains $\Omega_1(t)$ and $\Omega_2(t)$ such that $\Omega = \Omega_1(t) \cup \Omega_2(t)$. $\Omega_1(t)$ and $\Omega_2(t)$ represent the solid region and the liquid region, respectively. The boundaries of the domains are $\partial\Omega_1(t) = \Gamma_1(t) \cup S(t)$ of $\Omega_1(t)$ and $\partial\Omega_2(t) = \Gamma_1(t) \cup S(t)$ of $\Omega_2(t)$, where $S(t)$ is the moving phase interface common to $\Omega_1(t)$ and $\Omega_2(t)$. It is noted that all the domains and their boundaries are time dependent. The outward unit vector normal to boundaries $\partial\Omega_1(t)$ and $\partial\Omega_2(t)$ are $\vec{n}_1$ and $\vec{n}_2$, respectively. Any type of boundary conditions, including essential, natural, mixed, and radiation can be applied on parts of the boundaries $\partial\Omega_1(t)$ and $\partial\Omega_2(t)$. These boundary conditions are allowed to be time-dependent. We are interested in determining the position of phase interface $S(t)$, the temperature distributions $T_1(t)$ and $T_2(t)$, and the solute concentration distributions $C_1(t)$ and $C_2(t)$ at any instant in time.

2.2 GOVERNING EQUATIONS

A complete description of solidification processes involves the kinetics of atomic rearrangement near the phase interface, the transport of heat and mass in solid and liquid regions, the convection due to

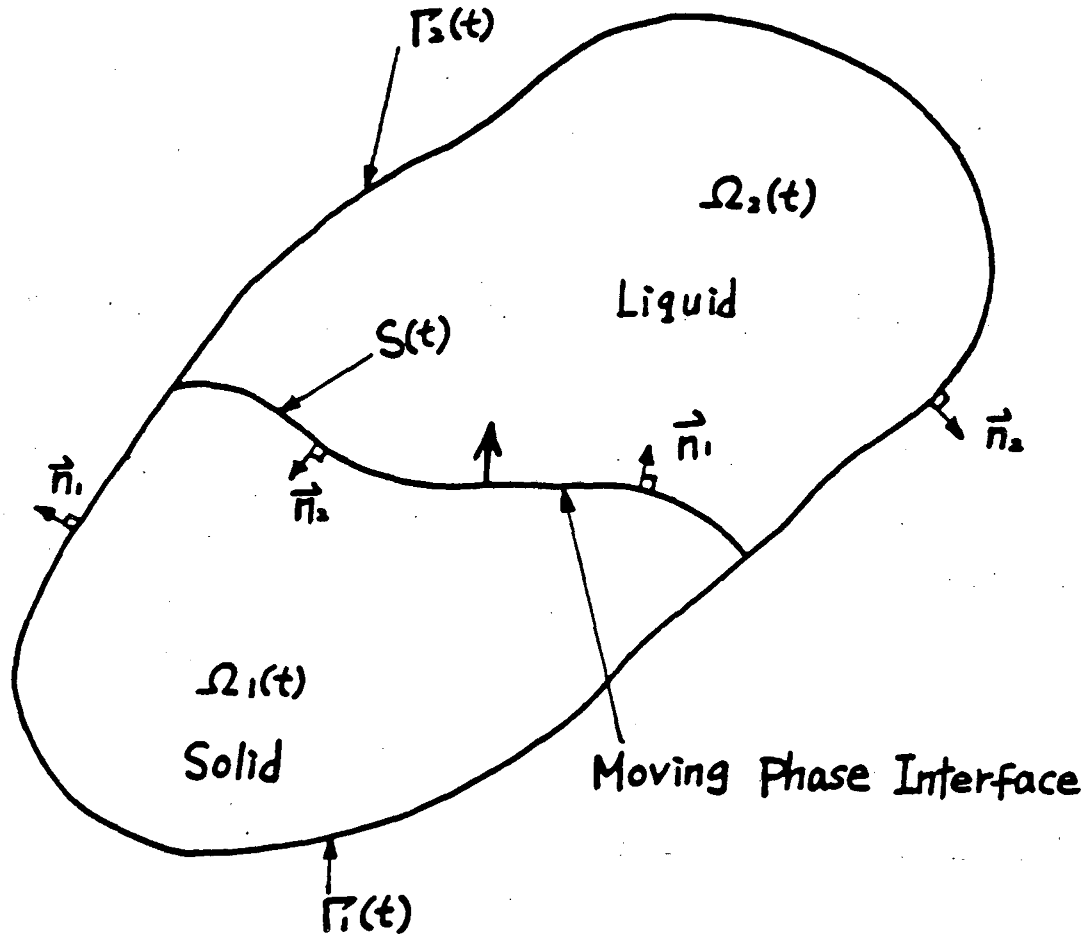


FIGURE 2.1. A general solidification problem.

difference in density of solid and liquid, the natural convection arising from density variation in the liquid, and so forth [48-63]. We will not deal directly with atomic effects or theories of nucleation kinetics. These theories are based on statistical mechanics and concern the fundamental processes of solidification on atomic scale. To make this problem tractable, we propose the following approximations:

(1) The solid-liquid interface is a definite surface in space.

(2) The effects of interface kinetics are negligible.

(3) There is thermodynamic equilibrium on the phase interface.

(4) There is no convection in the liquid.

(5) The mass diffusion in the solid region is negligible.

The governing equations describing solidification processes under the above assumptions are:

A. Heat Transfer Equations:

$$\nabla\cdot(k_1 \nabla T_1) = \rho_1 C_1 \frac{\partial T_1}{\partial t} \quad \text{in} \quad \Omega_1(t) \tag{2.1}$$

$$\nabla\cdot(k_2 \nabla T_2) = \rho_2 C_2 \frac{\partial T_2}{\partial t} \quad \text{in} \quad \Omega_2(t) \tag{2.2}$$

B. Mass Transfer Equations:

$$\nabla\cdot(D \nabla C) = \frac{\partial C}{\partial t} \quad \text{in } \Omega_2(t) \tag{2.3}$$

C. Interface Conditions:

$$(k_1 \nabla T_1 - k_2 \nabla T_2)\cdot\vec{n} = \rho_1 L(\vec{v}\cdot\vec{n}) \quad \text{on } S(t) \tag{2.4}$$

$$-D \nabla C\cdot\vec{n} = C(1-k)(\vec{v}\cdot\vec{n}) \quad \text{on } S(t) \tag{2.5}$$

$$T_1 = T_2 = T_m - mC - T_m\Gamma\gamma \quad \text{on } S(t)\ . \tag{2.6}$$

Here L is the latent heat of fusion per unit volume, k is the partition coefficient, T_m is the melting point of the pure substance with a planar interface, m is the slope of the liquidus line, which may vary as a function of solute concentration, Γ is the Gibbs-Thompson coefficient, γ is the mean interface curvature, $\vec{v}$ is the local solidification velocity, and $\vec{n}$ is the unit vector normal to the interface and coinciding with $\vec{n}_1$ on the interface.

All the material properties in Eqs. (2.1)-(2.6) may vary. Cases with non-isotropic and nonlinear (e.g., temperature and concentration dependent) properties are permitted. These governing equations are written in general vectorial form and are independent of the coordinate systems used.

It can be seen that all the governing equations are coupled at the interface through Eqs. (2.4)-(2.6). Equation (2.4) is obtained from the energy balance at the interface. Equation (2.5) represents the mass balance at the interface. Equation (2.6) indicates that under the thermodynamic equilibrium assumption, the temperature on the interface is determined by the interfacial concentration as well as the interfacial curvature. Hence the interface temperature is not only a function of time, bur varies along the interface. It should also be noted that γ, $\vec{n}$, and $\vec{v}$ all are functions of time and space.

Before proceeding to the next section, it is worth pausing to compare the assumptions and governing equations of this model with those of interface stability theories discussed previously. The most distinct aspect of this model is the incorporation of transient effects, which makes it close to representing realistic situations. The temperature and concentration distributions in both solid and liquid are changing with time, but are not fixed with respect to the interface. Every node on the interface may advance at an arbitrary velocity, depending on the applied boundary conditions. The present model provides us with the ability to study more complex situations than the previous quasi-steady models found in the literature [64,65].

2.3 INITIAL AND BOUNDARY CONDITIONS

2.3.1 Initial Conditions

Arbitrary initial conditions can be employed in the numerical analysis. However, in this specific study uniform conditions were chosen:

$$\begin{aligned} T &= T_1^0 \quad \text{in } \Omega_2 \\ T &= T_2^0 \quad \text{in } \Omega_2 \\ C &= C^0 \quad \text{in } \Omega_2 \end{aligned} \tag{2.7}$$

2.3.2 Boundary Conditions

In addition to the interface conditions [Eqs. (2.4)-(2.6)], which will serve as part of the boundary conditions for domains $\Omega_1(t)$ and $\Omega_2(t)$, the following general boundary conditions may be imposed on any parts of the boundaries:

$$\begin{aligned} \alpha_1 \nabla T_1 \cdot \vec{n}_1 + \beta_1 T_1 &= \gamma_1 \qquad \text{on } \Gamma_1(t) \\ \alpha_2 \nabla T_2 \cdot \vec{n}_2 + \beta_2 T_2 &= \gamma_2 \qquad \text{on } \Gamma_2(t) \\ \alpha_3 \nabla C \cdot \vec{n}_3 + \beta_3 C &= \gamma_3 \qquad \text{on } \Gamma_3(t) \;. \end{aligned} \tag{2.8}$$

It should be noted that any type of boundary conditions, including Dirichlet, Neumann, and mixed, can be obtained from the general boundary conditions, Eqs. (2.8), by properly choosing the coefficients α_i, β_i, and γ_i, where i = 1, 2, and 3. In particular, essential boundary conditions are derived from Eqs. (2.8) by the so-called penalty method. For example, if one chooses β_1 and γ_1 to be some very large values compared to α_1, then the first equation in Eqs. (2.8) is a good approximation to $T = \gamma_1/\beta_1 = T_C$, an essential boundary condition. By adopting Eqs. (2.8), the implementation also is simplified. The reader is reminded that these boundary conditions can be functions of time and space.

2.4 NONLINEARITY ON THE INTERFACE

Equations (2.1) to (2.6) indicate that diffusion of heat and mass are coupled only at the interface and are considered independent in the rest of the domain. The difficulties in obtaining the solutions stem in part from the fact that Eq. (2.4) is nonlinear. To demonstrate this, consider the simpler situation where $T_1 = T_2 = T_C$, a constant value and one-dimensional problem.

Take the total derivatives of T_1 and T_2 on the interface:

$$\left(\frac{\partial T_1}{\partial x}\,dx + \frac{\partial T_1}{\partial t}\,dt\right)_{x=S(t)} = \left(\frac{\partial T_2}{\partial x}\,dx + \frac{\partial T_2}{\partial t}\,dt\right)_{x=S(t)} = 0$$

$$\frac{\partial T_1}{\partial x}\frac{dS(t)}{dt} + \frac{\partial T_1}{\partial t} = \frac{\partial T_2}{\partial x}\frac{dS(t)}{dt} + \frac{\partial T_2}{\partial t} = 0 \quad \text{at } x = S(t) \; . \tag{2.9}$$

Rearrange to obtain

$$\frac{dS(t)}{dt} = \frac{-\partial T_1/\partial t}{\partial T_1/\partial x} = \frac{-\partial T_2/\partial t}{\partial T_2/\partial x} \,. \qquad (2.10)$$

Equation (2.4) becomes

$$k_1 \frac{\partial T_1}{\partial x} - k_2 \frac{\partial T_2}{\partial x} = \rho_1 L \frac{\partial T_1/\partial t}{\partial T_1/\partial x} = \rho_1 L \frac{\partial T_2/\partial t}{\partial T_1/\partial x} \,. \qquad (2.11)$$

The nonlinearity of Eq. (2.4) is therefore evident.

2.5 PREVIOUS NUMERICAL WORK

To date, a large number of analytical and numerical methods have been presented in the technical literature for the solution of problems of heat transfer with phase transformation in a pure substance. Many of these methods are summarized in Refs. 66-69. Most of the analytical methods are restricted to one-dimensional situations [70-72]. Multi-dimensional situations usually are solved using numerical methods [73-77]. Several solutions using finite elements method also have been reported [78-87].

The numerical techniques using finite elements can be separated into two groups based on the formulation of the problem. In the first group, enthalpy is the dependent variable (see Refs. 77,81-82,86-87). The second group of methods deals with the energy equation written in terms of temperature as the dependent variable (see Refs. 83-85). Solutions using the finite element for the enthalpy formulation include the work by Comini *et al.* [86], Ronel and Baliga [87], and Miller and Miller [81,82].

Bonnerot and Jamet [78] were the first to develop a finite element that discretizes the domain by means of isoparametric elements corresponding to a six-noded triangular prism in a space defined by the x-y Cartesian coordinates and t, the time variable. The free boundary was approximated by a polygon whose vertices coincided with triangulation nodes. In their method, the elements deformed continuously in time to accommodate the displacement of change of phase interface. The method discussed above is restricted to the solidification processes in pure substances. To study the physical phenomena that occur during solidification processes in solutions and alloys, we have developed a new multi-dimensional finite element method using "front tracking" finite elements.

The front tracking finite element method uses moving or deforming elements to track continuously in time the position of the change of phase interface. A general front tracking procedure for the study of solidification processes in a pure substance was first established and reported by Rubinsky *et al.* [83,85]. A different front tracking method was also developed by Rubinsky for the study of heat and mass transfer during one-dimensional transient solidification processes in a solution in the presence of forced convection [84].

In this study, a new general multi-dimensional numerical method of solution using front tracking finite elements for the study of heat and mass transfer problems during transient solidification processes in binary solutions was developed. Specific to the front tracking finite element method is the fact that the energy balance on the change of phase interface is not treated as a boundary condition, but rather as an independent equation whose solution gives the position of the interface in time. Because the front tracking method tracks the change of phase interface continuously in time, the method can deal with irregular interface morphologies and can consider the local thermodynamics on the interface, including capillary effects and nucleation kinetics.

CHAPTER 3:
FRONT TRACKING FINITE ELEMENTS

3.1 INTRODUCTION

There are no numerical methods that can be used to solve the mathematical model of a typical transient solidification process in binary solutions described by Eqs. (2.1) to (2.8). In this chapter, such a new general numerical method will be developed. Because of the characteristics of this problem, which is specified by an irregular transient change of phase interface, the finite element method, which is able to accommodate irregular geometries, was chosen as the most appropriate method of solution. In general, the finite element method can handle problems with complex geometries, anisotropic materials, and arbitrary boundary conditions. Finite element methods also permit refinement of the domain when necessary.

3.2 SPACE DISCRETIZATION

Since the governing equations (2.1) to (2.3) have the same form, we will develop the finite element formulation only for Eq. (2.1). For convenience, Eq. (2.1) is rewritten here and the subscript disregarded:

$$\nabla \cdot (k \nabla T) = \rho_C \frac{\partial T}{\partial t} \quad . \tag{3.1}$$

The finite element formulations will be derived in a general form, so that a general purpose program can be developed (limited to two-or three-dimensional axisymmetry problems). The governing equation is the diffusion equation, but the method can easily be extended to incorporate convection terms if necessary. Following standard procedures [88-96], the solution domain is first divided into M elements of arbitrary shape. In general, these elements can be rectangular, triangular, or mixed, and the number of nodes of each element can be different from each other. We approximate the unknown exact temperature T and its gradients of each element by

$$
\begin{aligned}
T(r,z,t) &= \sum_{i=1}^{N} \phi_i(r,z)\, T_i(t) \\
\frac{\partial T}{\partial r}(r,z,t) &= \sum_{i=1}^{N} \frac{\partial \phi_i}{\partial r}(r,z)\, T_i(t) \\
\frac{\partial T}{\partial z}(r,z,t) &= \sum_{i=1}^{N} \frac{\partial \phi_i}{\partial z}(r,z)\, T_i(t) \\
\frac{\partial T}{\partial t}(r,z,t) &= \sum_{i=1}^{N} \phi_i(r,z)\, \frac{dT_i}{dt}(t) \quad ,
\end{aligned}
\tag{3.2}
$$

where N is a finite value representing the number of nodes. $T_i(t)$ are unknown nodal values to be found, $\phi_i(r,z)$ are the shape or interpolation functions over element i.

In matrix form,

$$
\begin{aligned}
T(r,z,t) &= [\phi(r,z)]\{T(t)\} \\
\left.\begin{matrix} \dfrac{\partial T}{\partial r}(r,z,t) \\[2ex] \dfrac{\partial T}{\partial z}(r,z,t) \end{matrix}\right\} &= [B(r,z)]\{T(t)\} \\
\frac{\partial T}{\partial t}(r,z,t) &= [\phi(r,z)]\left\{\frac{dT}{dt}(t)\right\} \; .
\end{aligned}
\tag{3.3}
$$

where $[\phi(r,z)] = [\phi_1, \phi_2, \ldots, \phi_N]$ is the temperature interpolation matrix. $\{T(t)\} = [T_1, T_2, \ldots, T_N]^T$, and $dT/dt = \left[\frac{dT_1}{dt}, \frac{dT_2}{dt}, \ldots, \frac{dT_N}{dt}\right]^T$:

$$[B(r,z)] = \begin{bmatrix} \frac{\partial\phi_1}{\partial r} & \frac{\partial\phi_2}{\partial r} & \cdots & \frac{\partial\phi_N}{\partial r} \\ \frac{\partial\phi_1}{\partial z} & \frac{\partial\phi_2}{\partial z} & \cdots & \frac{\partial\phi_N}{\partial z} \end{bmatrix} \tag{3.4}$$

[B] is the temperature gradient interpolation matrix.

By the method of weighted residues, Eq. (3.1) becomes

$$\int_{\Omega_i} \left[\nabla\cdot(k\,\nabla T) - \rho_C \frac{\partial T}{\partial t}\right] \phi_i \, d\forall = 0 \ , \tag{3.5}$$

where Ω_i is the volume of element i and ϕ_i are weighting functions. The exact equation has been modified so that it will be satisfied only in a weighted average sense as Eq. (3.5). Integration by parts, followed by the use of Gauss' theorem, yields

$$\int_{\Omega_i} (k\,\nabla T \cdot \nabla\phi_i - \rho_C \frac{\partial T}{\partial t} \phi_i) \, d\forall - \int_{\Gamma_i} \phi_i k\,\nabla T \cdot \vec{n} \, dA = 0 \ , \tag{3.6}$$

where Γ_i is the boundary surface of element i. The integration by parts formula and Gauss' theorem are, respectively,

$$\begin{aligned} &\nabla\cdot(vk\ \nabla\ u) = k\,\nabla u \cdot v + v\nabla\cdot(k\,\nabla u) \\ &\int_{\Omega} \nabla \cdot \vec{\sigma} \, dA = \int_{\partial\Omega} \vec{\sigma} \cdot \vec{n} \, dS \end{aligned} \tag{3.7}$$

Any type of boundary condition can now be incorporated through the surface integral term in Eq. (3.6). The general boundary conditions are allowed as follows:

$$
\begin{aligned}
T &= T\,(r,z,t) && \text{on } S_1 \\
\nabla T \cdot \vec{n} &= -q(r,z,t) && \text{on } S_2 \\
\nabla T \cdot \vec{n} &= h(t - T_\infty) && \text{on } S_3 \\
\nabla T \cdot \vec{n} &= \sigma\varepsilon T^4 - \alpha q_r && \text{on } S_4
\end{aligned}
\quad , \tag{3.8}
$$

where $\Gamma_i = S_1 \cup S_2 \cup S_3 \cup S_4$. These boundary conditions represent the specific surface temperature, specified surface heat flux, convective heat transfer, and radiation heat transfer, respectively.

For nonisotropic materials, in general

$$
k \nabla T = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} \frac{\partial T}{\partial r} \\ \frac{\partial T}{\partial z} \end{Bmatrix} = [K][B(r,z)]\{T(t)\} \tag{3.9}
$$

Substituting Eqs. (3.8) and (3.9) into Eq. (3.6), we obtain the general form of the governing equation,

$$
[C]\left\{\frac{dT}{dt}(t)\right\} + [K]\{T(t)\} = \{R\} \quad , \tag{3.10}
$$

where

$$
[C] = \int_{\Omega_i} \rho_C\{\phi\}[\phi]\, d\forall \quad , \qquad \{R\} = \{R_T\} + \{R_q\} + \{R_h\} + \{R_r\}
$$

$$
[K] = [K_C] + [K_h] + [K_r] \quad , \qquad \{R_T\} = \int_{S_1} k \nabla T\{\phi\}\, dA
$$

$$
[K_C] = \int_{\Omega_i} [B]^T[K][B]\, d\forall \quad , \qquad \{R_q\} = \int_{S_2} q\{\phi\}\, dA
$$

$$
[K_h] = \int_{\Omega_i} h\{\phi\}[\phi]\, d\forall \;, \qquad \{R_h\} = \int_{S_3} hT_\infty\{\phi\}\, dA
$$

$$
[K_r]\{T\} = \int_{S_4} \sigma\varepsilon T^4\{\phi\}\, dA \;, \qquad \{R_r\} = \int_{S_4} \alpha q_r\{\phi\}\, dA \quad ,
$$

and where $d\forall$ (= r dr dz) is the volume of an element. It should be noted that the original governing equation, (3.1), has been reduced to a set of ordinary differential equations, (3.10).

The idea and formulations outlined above are quite simple and straightforward. However, it can be imagined that the calculations of any matrix in Eq. (3.10) for a curvilinear element would present great difficulty if performed directly in terms of the r-z coordinates. Furthermore, the character of such calculations (e.g., the limit of integration) would change from element to element in the domain. Thus we introduce an invertible transformation between the original arbitrary element and a master element of simple shape. Figure 3.1 shows this domain transformation, where r-z are the global coordinates and ξ-η are the local or natural coordinates. They are related through

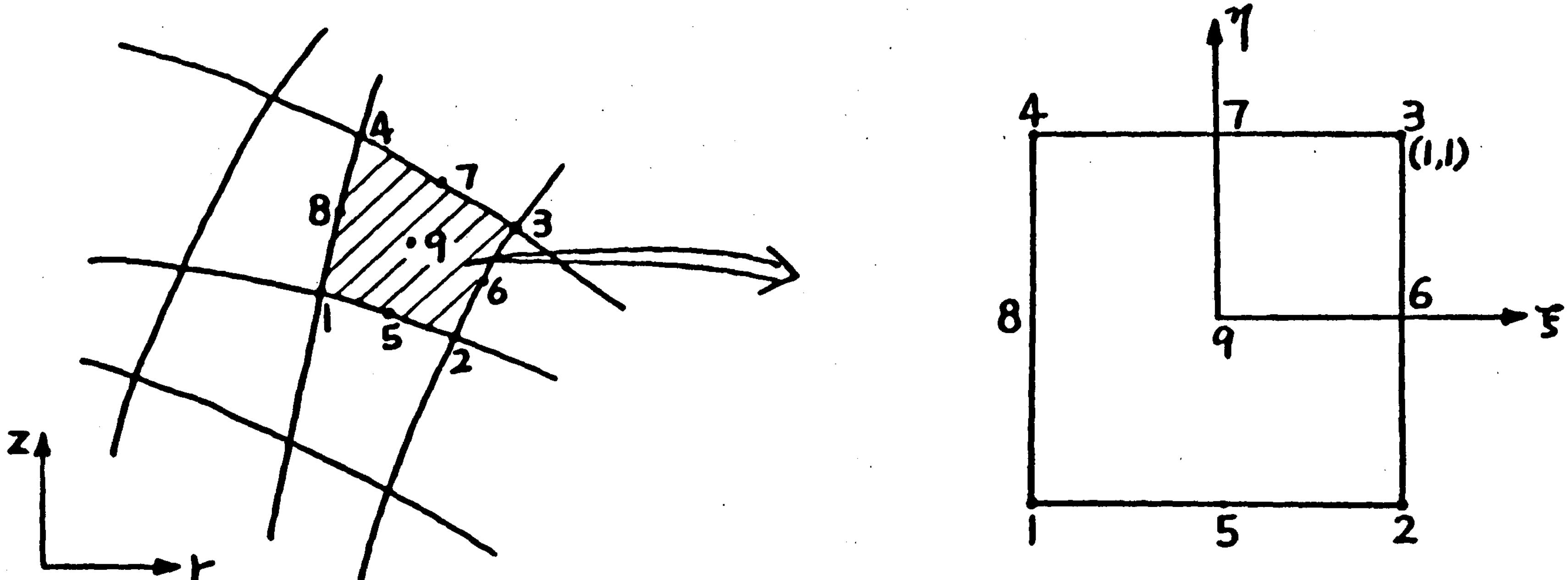
$$\begin{aligned} r &= r(\xi,\eta) \qquad -1 < \xi < 1 \\ z &= z(\xi,\eta) \qquad -1 < \eta < 1 \end{aligned} \tag{3.11}$$

by the chain rule of differentiation,

$$\begin{Bmatrix} \dfrac{\partial}{\partial \xi} \\[2ex] \dfrac{\partial}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \dfrac{\partial r}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\[2ex] \dfrac{\partial r}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \dfrac{\partial}{\partial r} \\[2ex] \dfrac{\partial}{\partial z} \end{Bmatrix} \quad , \tag{3.12}$$

where the Jacobian transformation matrix [J] is defined by

$$[J] = \begin{bmatrix} \dfrac{\partial r}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\[2ex] \dfrac{\partial r}{\partial \eta} & \dfrac{\partial z}{\partial \xi} \end{bmatrix} \quad .$$



$$\phi_1 = \tfrac{1}{4}(1-\xi)(1-\eta), \quad \phi_2 = \tfrac{1}{4}(1+\xi)(1-\eta)$$

$$\phi_3 = \tfrac{1}{4}(1+\xi)(1+\eta), \quad \phi_4 = \tfrac{1}{4}(1-\xi)(1+\eta)$$

$$\phi_5 = \tfrac{1}{2}(1-\xi^2)(1-\eta), \quad \phi_6 = \tfrac{1}{2}(1-\eta^2)(1+\xi)$$

$$\phi_7 = \tfrac{1}{2}(1-\xi^2)(1+\eta), \quad \phi_8 = \tfrac{1}{2}(1-\eta^2)(1-\xi)$$

$$\phi_9 = (1-\xi^2)(1-\eta^2)$$

FIGURE 3.1. Domains mapping from global coordinates r-z to local coordinates ξ-η.

The function $|J|$ is called the Jacobian of transformation,

$$|J| = \frac{\partial r}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial r}{\partial \eta}\frac{\partial z}{\partial \xi} \quad . \tag{3.14}$$

In order to guarantee that the transformation is unique, that is, no gaps or overlappings among elements, it is necessary to ensure that $|J| > 0$ for all elements. Then the evaluations of Eq. (3.10) will be performed on the master element.

The nodes in the ξ-η plane may be mapped into corresponding nodes in the r-z plane by defining

$$\begin{aligned} r(\xi,\eta) &= \sum_{i=1}^{N} \phi_i(\xi,\eta) r_i \\ z(\xi,\eta) &= \sum_{i=1}^{N} \phi_i(\xi,\eta) z_i \quad , \end{aligned} \tag{3.15}$$

where the r_i and z_i are nodal coordinates of an element. It is noted that the so-called isoparametric element is employed by adopting the same interpolation functions that were used previously to interpolate the temperature, i.e.,

$$T(\xi,\eta) = \sum_{i=1}^{N} \phi_i(\xi,\eta) T_i \quad , \tag{3.2}$$

where ϕ_i are no longer functions of r and z, but of ξ and η. The formula given by Eq. (3.15) is standard [88]. This transformation is also shown in Fig. 3.1. The ϕ_i have the characteristic that, for example, at node 1, $\phi_1 = 1$ and all other ϕ_i are zero, such that Eq. (3.2) is satisfied automatically.

Hence, each integration in Eq. (3.10) is evaluated by integrating over the square master element. For example, [C] becomes

$$[C] = \int_{-1}^{1}\int_{-1}^{1} \rho C\{\phi(\xi,\eta)\}\ [\phi(\xi,\eta)]\ |J(\xi,\eta)|\ r\ d\xi\ d\eta \quad . \tag{3.16}$$

The integrations are normally carried out by the method of Gauss-Legendre quadrature:

$$[C] = \sum_{i=1}^{NG}\sum_{j=1}^{NG} \omega_i\omega_j\rho C\{\phi(\xi_i,\eta_i)\}\ [\phi(\xi_i,\eta_j)] \cdot |J(\xi_i,\eta_j)|\ r_{i,j} \tag{3.17}$$

where ω_i and ω_j are Gauss weights, ξ_i and ξ_j are the coordinates of Gaussian points, and NG is the number of Gaussian points in each integration direction.

All other matrices in Eq. (3.10) are evaluated in a similar way, except for the temperature gradient interpolation matrix [B]. This matrix is transformed from r-z coordinates to ξ-η coordinates such that

$$[B] = \begin{bmatrix} \dfrac{\partial\phi_i}{\partial r} \\ \dfrac{\partial\phi_i}{\partial z} \end{bmatrix} = [J(\xi,\eta)]^{-1}\begin{Bmatrix} \dfrac{\partial\phi_i}{\partial\xi} \\ \dfrac{\partial\phi_i}{\partial\eta} \end{Bmatrix} \quad . \tag{3.18}$$

The matrix $[K_c]$ becomes

$$[K_c] = \int_{-1}^{1}\int_{-1}^{1} [B(\xi,\eta)]^T[K][B(\xi,\eta)]\ |J(\xi,\eta)|\ r\ d\xi\ d\eta \quad , \tag{3.19}$$

which is evaluated by Gauss-Legendre quadrature:

$$[K_c] = \sum_{i=1}^{NG}\sum_{i=1}^{NG}\omega_i\omega_j[B(\xi_i,\eta_j)]^T[K]\cdot[B(\xi_i,\eta_j)]\ |J(\xi_i,\eta_j)|\ r_{i,j} \quad . \tag{3.20}$$

3.3 TIME DISCRETIZATION

The general formulation of the weak form of governing equation (3.1) has been derived as

$$[C(T)]\left\{\frac{dT}{dt}(t)\right\} + [K(T,t)]\{T(t)\} = \{R(T,t)\} \quad . \tag{3.10}$$

This is a set of nonlinear ordinary differential equations with time t as the independent variable. This equation will be solved by numerical time integration. Two methods of solution are proposed. First, a typical finite different method is used, and then the finite element method in time is employed.

3.3.1 General Finite Difference Method

The general θ method is introduced such that $t_\theta = t_n + \theta\Delta t$, where $0 \leq \theta \leq 1$ and

$$[C(T)]_\theta\left\{\frac{dT}{dt}(t)\right\}_\theta + [K(T,t)]_\theta \{T(t)\}_\theta = \{R(T,t)\}_\theta \quad . \tag{3.21}$$

The subscript θ indicates the values are evaluated at time t_θ. We introduce the following approximations of standard finite difference method:

$$\begin{aligned} \{T\}_\theta &= (1-\theta)\{T\}_n + \theta\{T\}_{n+1} \\ \{R\}_\theta &= (1-\theta)\{R\}_n + \theta\{R\}_{n+1} \\ \left\{\frac{dT}{dt}\right\}_\theta &= \frac{\{T\}_{n+1} - \{T\}_n}{\Delta t} \quad . \end{aligned} \tag{3.22}$$

Similarly for $[C(T)]_\theta$ and $[K(T,t)]_\theta$. Substitute these into Eq. (3.21) to obtain the following formulation:

$$\left[\theta[K] + \frac{[C]_\theta}{\Delta t}\right]\{T\}_{n+1} = \left[-(1-\theta)[K] + \frac{[C]_\theta}{\Delta t}\right]\{T\}_n + \{R\}_\theta \ . \quad (3.23)$$

This equation represents a general family of recurrence relations: for $\theta = 0$, the algorithm is a Euler forward method; for $\theta = \frac{1}{2}$, a Crank-Nicolson method; for $\theta = 2/3$, a Galerkin method; and for $\theta = 1$, a backward method.

3.3.2 Finite Element Method in Time

Equation (3.23) can also be obtained by finite elements in the time domain. First, the time domain is divided into N elements, and for each element the weak form is obtained in the same way as for the spatial domain:

$$\int_{t_n}^{t_{m+1}} \omega(z)\left[C[\{\dot{T}\} + [K]\{T\} - \{R\}\right] dz = 0 \quad , \qquad (3.24)$$

where $\omega(z)$ is an arbitrary weighting function. The approximation is applied in one-dimensional two-node time elements such that

$$T = (1-\xi)T_n + \xi T_{n+1}$$

$$\dot{T} = \frac{dT}{dt} = \frac{T_{n+1} - T_n}{\Delta t_n} \qquad (3.25)$$

$$\xi = \frac{t - t_0}{\Delta t_n} \quad .$$

Substituting in Eq. (3.24) produces

$$\int_{t_n}^{t_{n+1}} \omega(\tau)\left\{\left([C]\,\frac{\{T\}_{n+1} - \{T\}_n}{\Delta t}\right)\right.$$
$$\left. + [K]\left[(1-\xi)T_n + \xi T_{n+1}\right] - \{R\}\right\} d\tau = 0 \qquad (3.26)$$

by the mean value theorem,

$$\int_{t_n}^{t_{n+1}} \xi\omega(\tau)\; d\tau = \int_{t_n}^{t_{n+1}} \omega(\tau)\; d\tau \quad , \qquad (3.27)$$

where $0 < \theta < 1$. Hence,

$$\int_{t_n}^{t_{n+1}} \omega(\tau)\left\{[C]\,\frac{\{T\}_{n+1} - \{T\}_n}{\Delta t}\right.$$
$$\left. + [K]\left[(1-\theta)T_n + \theta T_{n+1}\right] - \{R\}\right\} d\tau = 0 \; . \qquad (3.28)$$

Since the weighting function $\omega(\tau)$ is arbitrary,

$$[C]\left\{\frac{T_{n+1} - T_n}{\Delta t}\right\} + [K]\left\{(1-\theta)T_n + \theta T_{n+1}\right\} - \{R\} = 0 \qquad (3.29)$$

is derived.

In general, [C], [K], and {R} are not constants but functions of θ.

$$\left[\theta[K]_\theta + \frac{[C]_\theta}{\Delta t}\right]\{T\}_{n+1} = \left[-(1-\theta)[K] + \frac{[C]_\theta}{\Delta t}\right]\{T\}_n + \{R\} \quad . \qquad (3.30)$$

Thus the same form is obtained in this way as was derived by the finite difference method.

3.4 NUMERICAL STABILITY ANALYSIS

Consider the following set of linear differential equations:

$$[C]\{\dot{T}\} + [K]\{T\} = \{R\} \ , \tag{3.10}$$

where the coefficients [C], [K], and {R} are constants. To study their numerical stability, these differential equations are transformed into the modal form, i.e., a set of independent scalar equations. Then the solution of Eq. (3.10) is just the superposition of the solution of each scalar equation. The stability analysis is concentrated on each scalar mode. First, assume the case of free response with R = [0]. The general solution of Eq. (3.10) can be assumed as

$$\begin{aligned} \{T\} &= \{C_i\}\, e^{-\lambda_i t} \quad , \quad i = 1,\ldots,n \\ \{\dot{T}\} &= -\lambda_i\ \{C_i\}\, e^{-\lambda_i t} \quad , \end{aligned} \tag{3.31}$$

where $\{C_i\}$ is a modal vector of unknown amplitude and λ_i is a modal decay constant. Substituting into Eq. (3.10), we derive

$$\left[-\lambda_i[C] + [K]\right]\{C_i\}e^{-\lambda_i t} = [0] \ , \tag{3.32}$$

where

$$\lambda_i[C] + [K] = [0]$$

is the characteristic polynomial for λ_i. Obviously, this is a standard eigenvalue problem; therefore, the following equality has to be satisfied:

$$[K]\{\phi_i\} = \lambda_i[C]\{\phi_i\} \ , \tag{3.33}$$

where $\{\phi_i\}$ are eigenvectors corresponding to eigenvalues λ_i. These eigenvectors are subject to the orthogonality condition,

$$\{\phi_i\}^T[C]\{\phi_i\} = \delta_{ij} = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases} , \tag{3.34}$$

so

$$\{\phi_J\}^T[K]\{\phi_i\} = \lambda_i\{\phi_J\}^T[C]\{\phi_i\} = \lambda_i\delta_{ij} \quad . \tag{3.35}$$

Hence, the solutions are the eigenpairs $(\lambda_i,\{\phi_i\})$.

We define

$$[\Phi] = [\{\phi_1\} \ \ldots \ \{\phi_n\}] \quad . \tag{3.36}$$

Then the solution $\{T\}$ may be expressed as a linear combination of all eigenvectors:

$$\begin{aligned} \{T\} &= [\Phi]\{v\} \\ \{\dot{T}\} &= [\Phi]\{\dot{v}\} \quad , \end{aligned} \tag{3.37}$$

where $\{v\}$ is a vector of generalized modal unknowns. Substituting into Eq. (3.10), we get

$$\begin{aligned} &[C][\Phi]\{\dot{v}\} + [K][\Phi]\{v\} = \{R\} \\ &[\Phi]^T[C][\Phi]\{\dot{v}\} + [\Phi]^T[K][\Phi]\{v\} = [\Phi]^T\{R\} \quad . \end{aligned} \tag{3.38}$$

It is noted that

$$\begin{aligned} &\{\Phi\}^T[C]\{\Phi\} = [I] \\ &\{\Phi\}^T[K]\{\Phi\} = [\Lambda] \\ &[\Lambda] = [\lambda_1 \ \ldots \ \lambda_n] \quad , \end{aligned} \tag{3.39}$$

and

$$\{\dot{v}\} + [\Lambda]\{v\} = \{g\} \tag{3.40}$$

is obtained, where $\{g\} = \{\Phi\}^T\{R\}$.

Finally the decoupled differential equations are derived for each node:

$$\dot{v}_i + \lambda_i v_i = g_i \quad . \tag{3.41}$$

Discarding the subscript i and applying the θ method gives

$$\begin{aligned}
\dot{v} &= (v_{n+1} - v_n)/\Delta t \\
v &= (1-\theta)v_n + \theta v_{n+1} \ , \quad 0 < \theta < 1 \\
v_{n+1} &= \frac{1-(1-\theta)\lambda\Delta t}{1+\Delta t\lambda\theta} v_n + \frac{1}{1+\Delta t\lambda\theta} g_n \ .
\end{aligned} \tag{3.42}$$

The amplification factor is

$$r_i = \frac{v_{n+1}}{v_n} = \frac{1-(1-\theta)\lambda_i\Delta t}{1+\Delta t\lambda_i\theta} \quad . \tag{3.43}$$

The requirement of stable solutions is

$$|r_i| < 1 \ , \tag{3.44}$$

which corresponds to the condition

$$\lambda_i\Delta t(-1+2\theta) > -2 \quad . \tag{3.45}$$

It is noted that for positive $\lambda_i\Delta t$, if $\theta \geq \frac{1}{2}$ the algorithm is unconditionally stable. For $\theta < \frac{1}{2}$, a conditionally stable method is used and the critical time step is determined by

$$\Delta t_{cr} = \frac{2}{1-2\theta}\frac{1}{\lambda_i} \ , \quad 0 \leq \theta < \frac{1}{2} \ . \tag{3.46}$$

Hence, if $\Delta t \leq \Delta t_{cr}$, it produces a stable method, and if $\Delta t > \Delta t_{cr}$, it results in an unstable method.

The numerical stability for various θ are summarized in Fig. 3.2.

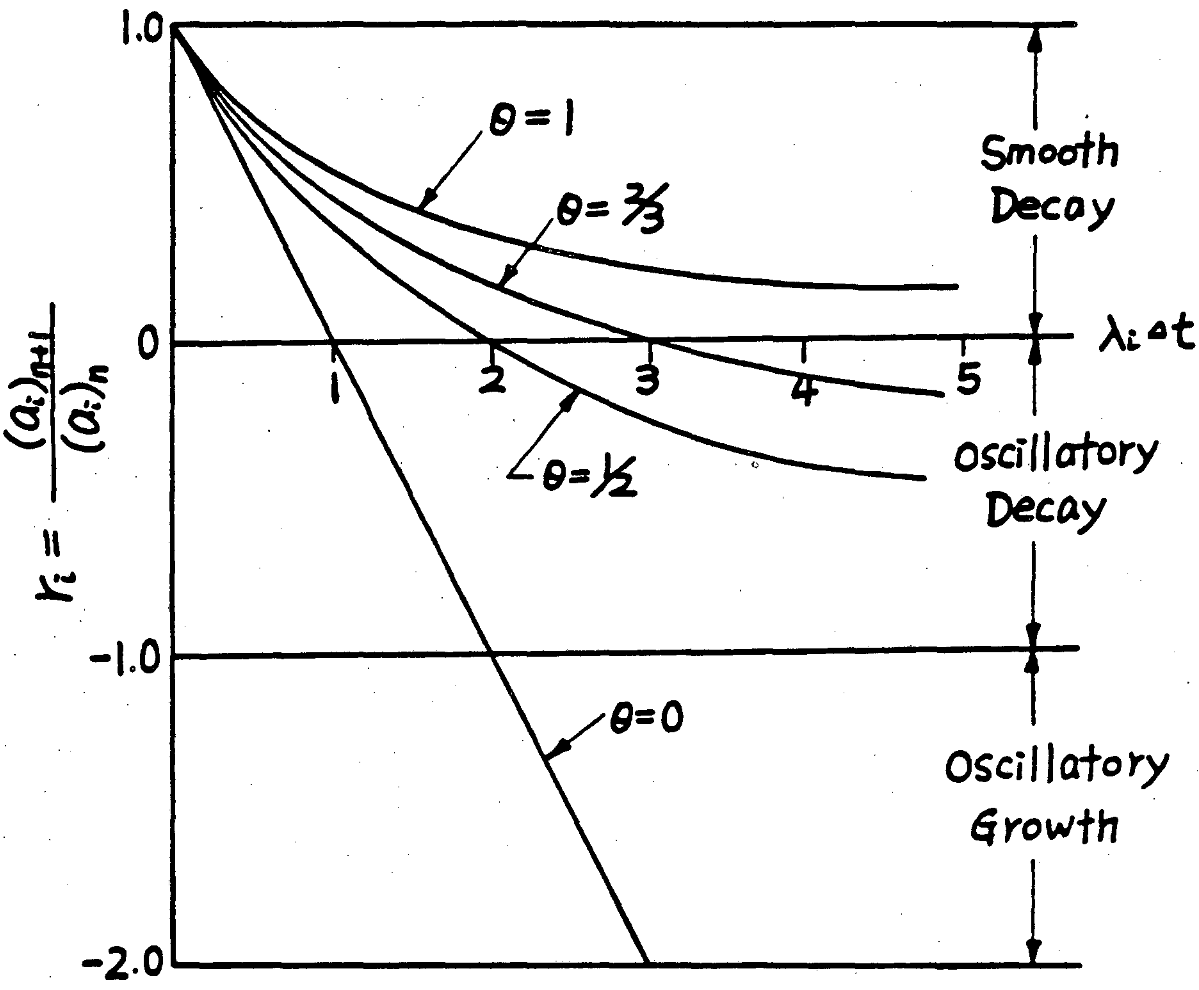


FIGURE 3.2. Stability behavior of θ method. θ = 0, Euler forward; θ = ½, Crank Nicolson, θ = 2/3, Galerkin; θ = 1, Euler backward.

3.5 INTERFACE MOVING SCHEME

The most important aspect of this work is a new numerical procedure for the solution of the interface condition, Eq. (2.4). The condition will be solved as an independent equation to obtain the new interface positions in time. For convenience, this equation is rewritten here:

$$(K_1 \nabla T_1 - K_2 \nabla T_2) \cdot \vec{n} = \rho_1 L(\vec{v} \cdot \vec{n}) \quad \text{on } S(t) . \tag{2.4}$$

Several methods have been proposed for the solution of this equation [50,51,75,79,85]. Those methods, however, are only applicable to situations in which the interface is isothermal. The isothermal interface occurs when the pure liquid is freezing and the effect of interfacial curvature is neglected. In a general solidification problem such as the one studied here, the temperature can vary along the interface as a function of interfacial concentration and interfacial curvature. This is shown in Eq. (2.6). Therefore, in the present study a new method is developed for the solution of Eq. (2.4).

The simplified example shown in Fig. 3.3 will be used to illustrate this method. Equation (2.4) is integrated along the interface S(t) by the finite element method. First the domain S(t) is divided into N two-node elements. The corresponding four-node isoparametric elements near the interface for both solid and liquid domains are also shown in the same figure. Each four-node element on both sides of the interface has one side that coincides with the interface. The heat flux terms in the left-hand side of Eq. (2.4), $k_1(\partial T_1/\partial n)$ and $k_2(\partial T_2/\partial n)$, will be evaluated on the corresponding four-node isoparametric elements of Ω_1 and Ω_2 , respectively.

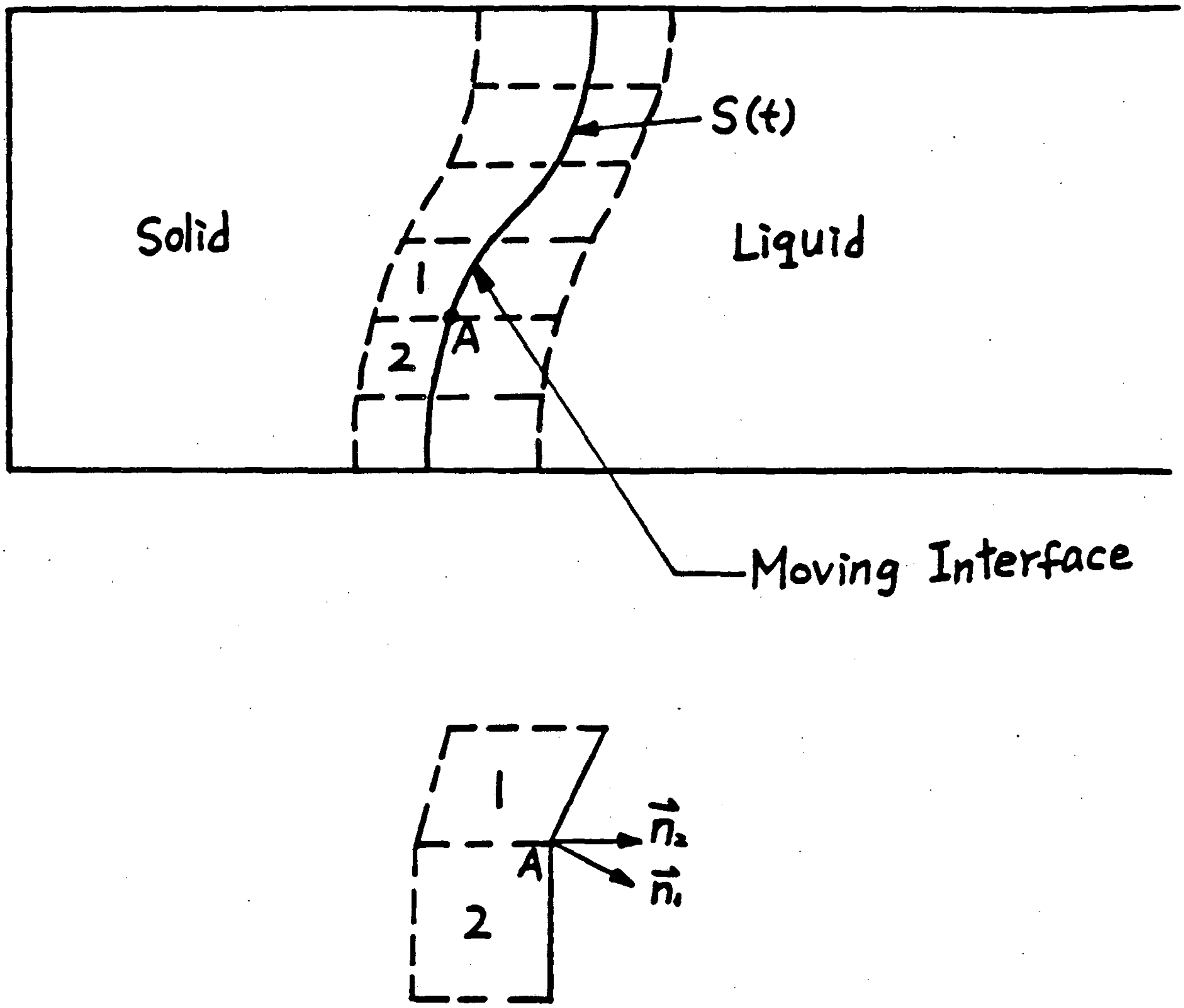


FIGURE 3.3. Illustration of interface moving scheme.

The velocity of the interface that appears in the right-hand side of Eq. (2.4) can be expressed as

$$\vec{v} \cdot \vec{n} = v_n = \frac{dn}{dt} \quad , \tag{3.47}$$

where $\vec{n}$ indicates the direction normal to the interface S and dn represents the magnitude of the displacement of the interface in that direction.

The normal displacement dn can be expressed within each element along the interface as:

$$dn = \sum_{i=1}^{N} N_i \; dn_i \quad , \tag{3.48}$$

or in matrix form as dn = [N] {dn}, where [N] is the displacement interpolation vector and {dn} is the vector of element nodal displacement. Specifically in the simplified two-node element,

$$[N] = [\tfrac{1}{2}(1+\xi) \quad \tfrac{1}{2}(1-\xi)]$$

in the local coordinate system. The finite element Galerkin formulation of Eq. (2.4) is

$$\int_{S_i} (K_1 \nabla T_1 - K_2 \nabla T_2)\cdot\vec{n}\;[N]dS = \int_{S_i} \frac{\rho_1 L}{dt}\{N\}[N]\;\{dn\}\;dS \quad . \tag{3.49}$$

To evaluate the integration, the global coordinate is changed to the local coordinate by the Jacobian transformation:

$$\int_{-1}^{1} (K_1 \nabla T_1 - K_2 \nabla T_2)\cdot\vec{n}[N]|J|\;d\xi = \int_{-1}^{1} \frac{\rho_1 L}{dt}\{N\}[N]\{dn\}|J| \quad d\xi \quad . \tag{3.50}$$

Since the arc length of interface ds can be expressed as

$$dS = \left\{ \left[\frac{\partial r(\xi,1)}{\partial \xi} \right]^2 + \left[\frac{\partial z(\xi,1)}{\partial \xi} \right]^2 \right\}^{\frac{1}{2}} d\xi \quad , \tag{3.51}$$

then

$$|J| = \left\{ \left[\frac{\partial r(\xi,1)}{\partial \xi} \right]^2 + \left[\frac{\partial z(\xi,1)}{\partial \xi} \right]^2 \right\}^{\frac{1}{2}} \quad . \tag{3.52}$$

The value of $|J|$ is the two-node element is given by

$$|J| = \tfrac{1}{2}[(r_1 - r_2)^2 + (z_1 - z_2)^2]^{\frac{1}{2}} \quad , \tag{3.53}$$

the half-length of an element.

The integration is evaluated by the Gauss-Legendre quadrature:

$$\sum_{i=1}^{NG} \omega_i (K_1 \nabla T_1 - K_2 \nabla T_2) \cdot \vec{n} [N(\xi_i)] |J| = \sum_{i=1}^{NG} \omega_i \frac{\rho_1 L}{dt} \{N(\xi_i)\}[N(\xi_i)] \{dn\} |J| \ . \tag{3.54}$$

Next, the evaluation of the heat flux at each Gauss point will be discussed. The value of $K_1 \nabla T_1 \cdot \vec{n}$ is calculated on the interface using the corresponding adjacent four-node isoparametric element in domain Ω_1. The procedures are the same for computing $K_2 \nabla T_2 \cdot \vec{n}$. It is noted that the temperature distributions T_1 and T_2 are known at this stage.

By the chain rule of differentiation,

$$\frac{\partial T_1}{\partial n} = \begin{bmatrix} \frac{\partial r}{\partial n} & \frac{\partial z}{\partial n} \end{bmatrix} \begin{Bmatrix} \frac{\partial T_1}{\partial r} \\ \frac{\partial T_1}{\partial z} \end{Bmatrix} \quad \text{on } S(t) \ , \tag{3.55}$$

and from Eqs. (3.3) and (3.18),

$$\begin{bmatrix} \frac{\partial T_1}{\partial r} \\ \frac{\partial T_1}{\partial z} \end{bmatrix} = [B_1(r,z)]\{T_1(t)\} = [J(\xi,\eta)]^{-1} \begin{Bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \end{Bmatrix} \{T_1(t)\} \ , \tag{3.56}$$

where $[B_1]$ is the temperature gradient interpolation matrix. It is noted that the values of interpolation functions in the two-dimensioal four-node isoparametric element, when evaluated on the interface, are the same as those of the one-dimensional two-node element. For the purpose of illustration, the $[B_1]$ matrix at a Gauss point on the interface is given by

$$[B\ ] = \tfrac{1}{4}J^{-1}\begin{bmatrix} 2 & 2 & 0 & 0 \\ 1+\xi_i & 1-\xi_i & \xi_i-1 & -\xi_i-1 \end{bmatrix} \text{ on } \xi = \xi_i,\ \eta = 1 \qquad (3.57)$$

To simplify notation, the element's directional cosine matrix will be denoted as [U] such that

$$\frac{\partial T_1}{\partial n} = \begin{bmatrix} \frac{\partial r}{\partial n} & \frac{\partial z}{\partial n} \end{bmatrix} \begin{Bmatrix} \frac{\partial T_1}{\partial r} \\ \frac{\partial T_1}{\partial z} \end{Bmatrix} = [U][B_1(\xi,\ \eta=1)]\ \{T_1(t)\} \qquad (3.58)$$

on S(t). The final form of the finite element formulation becomes

$$\sum_{i=1}^{NG} \omega_i [N(\xi_i)][U] \quad K_1[B_1(\xi_i,\eta=1)\{T_1\} - K_2[B_2(\xi_i,\eta=1)]\{T_2\} \quad |J|$$

$$= \sum_{i=1}^{NG} \omega_i \frac{\rho_1 L}{dt} \{N(\xi_i)\}[N(\xi_i)]\ \{dn\}\ |J| \quad . \qquad (3.59)$$

Or, in matrix form,

$$[A_1][T_1] - [A_2][T_2] = [A_3][dn] \quad , \qquad (3.60)$$

where $[A_1]$, $[A_2]$, and $[A_3]$ are matrices obtained from matrix multiplications. It is noted that the vector [U] has the same value for each element when the interface is flat. However, in a general curved interface the vector [U] may vary from element to element. In summary, we consider Eq. (3.60) as a one-dimensional problem; the left-hand side $[A_1][T_1]$ and $[A_2][T_2]$ are the source terms, while [dn] is unknown. The solution of Eq. (3.60) will yield the magnitude of displacement of each node on the interface.

From thermodynamic considerations it can be shown that each point on the interface moves in a direction locally normal to the interface. The direction in which the interface moves is a function of space along the interface and of time. Assume that at any instant in time, there is a node A, as shown in Fig. 3.3, common to elements 1 and 2, where $\vec{n}_1$ and $\vec{n}_2$ are normal directions on the side along the interface of element 1 and element 2, respectively. It is seen that $\vec{n}_1$ and $\vec{n}_2$ are different in general. However, we need a unique normal direction of each node on the interface. Newton's divided difference formula has been used to construct the interpolation polynomial:

$$P_n(x) = f(x_0) + (x-x_0)f[x_0,x_1] + \ldots + (x-x_0)\ \ldots\ (x-x_{n-1})f[x_0,\ldots,x_n] \tag{3.61}$$

where $(x_0,f(x_0))$, $(x_1,f(x_1))$, ..., $(x_n,f(x_n))$ are the coordinates of points to be interpolated. Newton's divided difference, $f[x_0,x_1,\ldots,x_n]$, is given by

$$f[x_0,\ldots,x_n] = \frac{f[x_1,\ldots,x_n] - f[x_0,\ldots,x_{n-1}]}{x_n - x_0} \quad . \tag{3.62}$$

It is well known that an interpolation polynomial of high degree, say n > 8, on the evenly spaced points will result in a larger error when the interpolation point is near both sides of the domain [97-100].

To minimize the interpolation error one has to choose an interpolation point as close as possible to the center of the domain. Hence an interpolation polynomial is not constructed through all the nodes on the interface, but rather, five nodes for a fourth-degree polynomial are constructed to find out the center node normal direction. There are N fourth-degree polynomials. The same polynomial is also used to compute the local radius of curvature on the interface by the formula

$$R = \frac{|P_n''|}{(1 + P_n'^2)^{3/2}} \quad , \tag{3.63}$$

where $P_n(x)$ is the polynomial obtained before.

The magnitude of the displacement of each node calculated using Eq. (3.60) and the direction normal to the interface evaluated using Eq. (3.62) are combined to determine the new locations of the interface. The magnitude of normal velocity of each node equals the displacement of that node divided by the time step size, dn/dt. The velocity is in the same direction as the displacement.

Interface condition, Eq. (2.5),

$$-D\,\frac{\partial C}{\partial n} = (1 - k)C\,\frac{\partial n}{\partial t} \quad \text{on} \quad S(t)$$

is one of the boundary conditions used for the solution of the mass transfer equation. Since the nature boundary condition is imposed not on a node but on the whole segment, the value dn/dt on an element is approximated by averaging the value of dn/dt at those two nodes on the interface for the four-node example.

3.6 SOLUTION ALGORITHM

The differential equations (2.1) to (2.3), together with interface conditions (2.4) to (2.6) and boundary conditions (2.8) have to be solved simultaneously for given initial conditions (2.7). The solution obtained will include the transient temperature distribution in the solid, the transient temperature and concentration distributions in the liquid, and the transient position of phase interface.

In general, the governing equations (2.1) to (2.3) have to be solved in an iterative way such that the interface conditions (2.4) to (2.6) are satisfied at any time. However, in the present study we employ the "front tracking" method, in which the solutions of governing equations (2.1) to (2.3) are sought individually, and Eq. (2.4) is used to "move" the interface. For each equation from (2.1) to (2.3), the θ method of the finite element formulation is used. The characteristic of this numerical method, i.e., implicit, explicit, or mixed, depends on the values of θ. The fully implicit method can be obtained by choosing θ = 1.0. However, the interface condition (2.4) is solved explicitly. The interface position is tracked continuously in time and will be used for the automatic mesh generation of each domain. The moving scheme will be elaborated in the next section.

Through numerical testing, it has been found that the implicit-explicit method for the solutions of the solidification problem gives good results in terms of accuracy and numerical stability, while significantly reducing the computational time. Hence the governing equations are solved in sequence without iteration and marching in time.

Of course, for the problems with nonlinear properties, within each solution of Eqs. (2.1) to (2.3), an iteration scheme has to be used.

In summary, the governing equations will be solved as follows:

(1) Assume the initial interface location and initial distributions of temperature T_1, T_2, and concentration C.

(2) Solve Eq. (2.4) to obtain the new interface locations, interface moving velocities, and interface curvatures in the next time step.

(3) Calculate the new concentration distribution C from Eq. (2.3), together with interface condition (2.5) and boundary condition (2.8). The moving interface velocities are obtained from step (2). The iteration is required for nonlinear properties of Eq. (2.3).

(4) Calculate the new interface temperature distribution form Eq. (2.6), using the thermodynamic relations between temperature and concentration C on the interface, and the interface curvatures. The concentration C and the curvature associated with each node on the interface are obtained from steps (3) and (2), respectively.

(5) Calculate the temperature distribution T_1 from Eq. (2.1) with interface condition (2.6) obtained in step (4) and boundary conditions (2.8). Iterations are required for nonlinear properties of Eq. (2.1).

(6) Calculate the temperature distribution T_2 from Eq. (2.2) with interface condition (2.6) obtained in step (4) and boundary condition (2.8). Iterations are required for nonlinear material properties.

(7) Go to step (2) and march forward in time.

3.7 AUTOMATIC MESH GENERATION

Since the phase interface is changing in time, the size and shape of both the solid and liquid domains vary during the solidification process. An automatic generation of nodes and elements in each domain at every time step is necessary to successfully solve this moving boundary problem. The nodes on the interface are tracked at all times, as discussed in the preceding section. Based on these interfacial nodes, an automatic mesh generation scheme is developed. It is noted that the typical information to be obtained from meshing a domain includes the total number of nodes and their global numbering, the total number of elements and their global numberings, the number of nodes and local numbering of each element, the coordinates of each node, etc.

The governing equations (2.1) to (2.3) indicate that the temperatures and concentration distributions are determined only by the diffusion mechanism. The existence of a concentration "boundary layer" near the solid-liquid interface during solidification in solution was discussed in the introduction. The boundary layer thickness is proportional to the magnitude of the diffusion coefficient. The typical relative order of magnitude of the diffusion coefficients in the analyzed problem are: thermal diffusivity of solid, $\alpha_1 = 0(1)$, thermal diffusivity in liquid, $\alpha_2 = 0(10)^{-1}$, and mass diffusivity of liquid, $D = 0(10)^{-3}$. For example, in the solidification of saline solutions, α_1 of ice is 1.26×10^{-6} m^2/sec, α_2 is 1.33×10^{-7} m^2/sec, and D is 1.29×10^{-9} m^2/sec. The wide range of values for diffusion coefficients implies that the boundary layer thickness for the temperature distribution is significantly different from that for concentration. Numerous numerical difficulties are associated with this fact. In fact, strong oscillations in numerical solutions were observed during the initial studies. To overcome this difficulty, three different meshing systems were designed: for the solid temperature, for the liquid temperature, and for the concentration in the liquid. The meshing strategy will be illustrated on the geometrical configuration shown in Fig. 3.4, which will be later used in the stability study.

The meshing strategy for temperature on domain Ω_1, as shown in Fig. 3.4, will be illustrated. The nodes on $z = 0$ are generated by either predetermining the number of nodes or by determining a reference length between the nodes for a given global dimension, R in the r-direction. The grid size may be uniform, as used in this study, or variable depending on the characteristics of the problems. The number of interfacial nodes is the same as that on the outer boundary. The r coordinate of each interfacial node has the same value as the corresponding node on the outer boundary. Thus every line connecting two nodes, one on the outer boundary and the other on the interface, is parallel to the z-direction. It is noted that this parallel requirement is not necessary, but substantial computer time was saved by using this constraint. This point will be elaborated later in this section.

At every time step the maximum distance of any interfacial node from the outer surface was found. This maximum length was divided by a predetermined reference length. The roundoff integer obtained is the number of segments on each line parallel to the z-direction. Since the

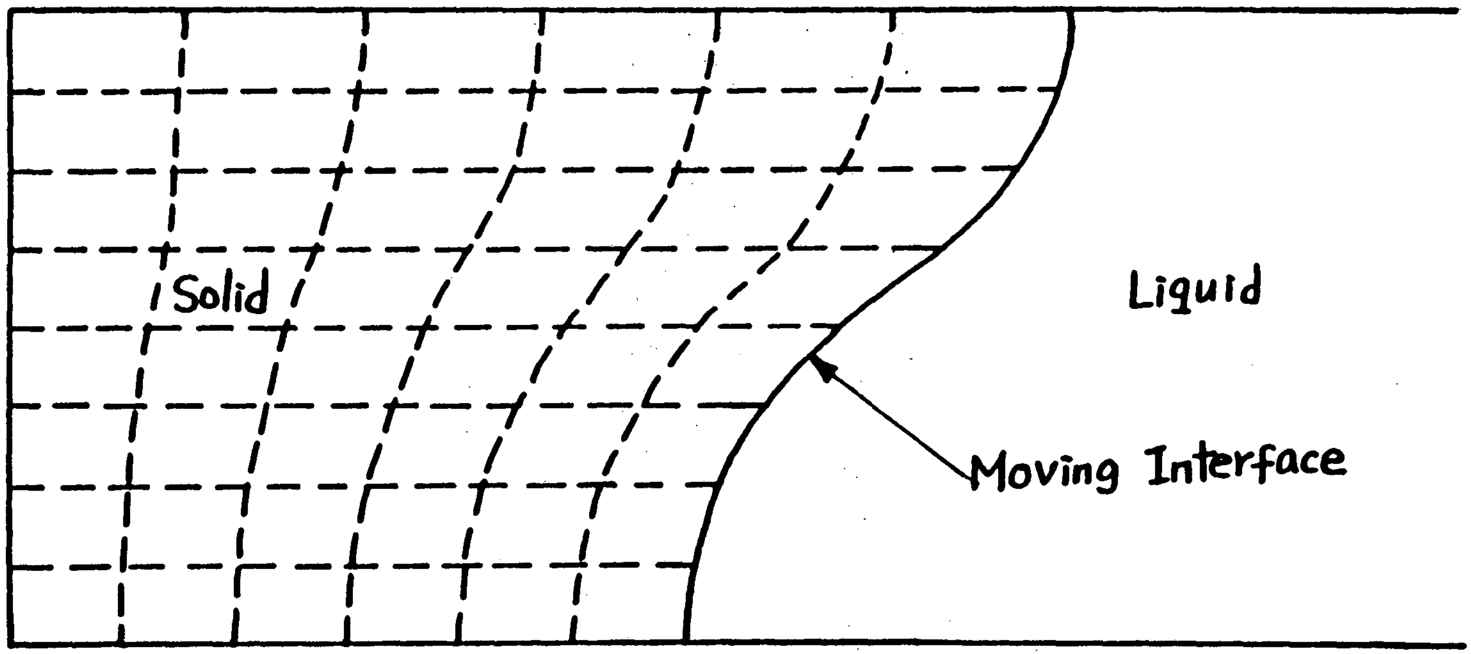


FIGURE 3.4. Automatic mesh generation of temperature domain in solid.

heat flux in Eq. (2.4) must be evaluated on the change of phase interface, the grid size in the z-direction in the vicinity of the interface must be refined for better accuracy. The domains were re-meshed at each time step. The advantage of re-meshing is that smooth solutions are obtained between each time step. The disadvantage is an increase in computer time.

The mesh for temperature T_2 in the domain Ω_2 in Fig. 3.4 is generated exactly the same way as the meshing procedure described above. The reference length for meshing is, however, smaller, since the thermal diffusivity in the liquid is one order of magnitude smaller than that in solid. It is noted that the number of elements and nodes in Ω_1 continuously increase in time, but in Ω_2 they decrease. During the automatic meshing procedures, care was taken to satisfy the compatibility condition between elements in the solid and liquid domains.

The solute concentration boundary layer in the liquid phase is very thin relative to the temperature boundary layer. It is necessary to have good resolution within this layer in order to obtain wiggle-free solutions. Thus from both theoretical and practical points of view, it is inadequate to employ the same fixed mesh size used for the temperature solutions in the solution of mass diffusion equation. It was found that a variable mesh size distribution is more suitable. Experimental results show that the concentration decreases exponentially in a direction normal to the interface [8-21]. Hence a new systematic procedure of using exponential functions to generate the mesh was developed. The resulting mesh size increases exponentially in a direction normal to the boundary. This new method will be illustrated by an example of meshing using one-dimensional two-node elements. This is shown in Fig. 3.5. Suppose that one desires to distribute N nodes within distance L so that the mesh size Δx_i increases exponentially. The grid size is determined by the formulas

$$A(I) = \exp\left[\frac{-I(Ke)}{N-2}\right]$$

$$C = \sum_{i=0}^{N-2} A(I)$$

$$\Delta x(I) = \frac{L}{C} A(I) ,$$

where $I = 0,1,\ldots,N-2$, $\Delta x(I)$ are the grid size as shown in Fig. 3.5. Ke is the meshing coefficient, which determines the mesh size Δx_i. Small adjustments of Ke can significantly affect the meshing. For example, a meshing coefficient of 5.3 will make the distance between the first two nodes smaller by a factor of about 200 than the distance between the last two nodes, Δx_{N-2}, that is, $\Delta x_{N-2} = \Delta x_0(200)$, while a meshing coefficient of 6.4 will result in a factor of about 600. The special case of Ke = 0 corresponds to the equal mesh size. The range to be exponentially gridded can be chosen if desired to be only part of the domain. In situations where the boundary layer thickness changes substantially in time, a time-dependent meshing coefficient could be employed. This concept can be extended to two- and three-dimensional domains.

As the interface advances the liquid domain decreases, and one

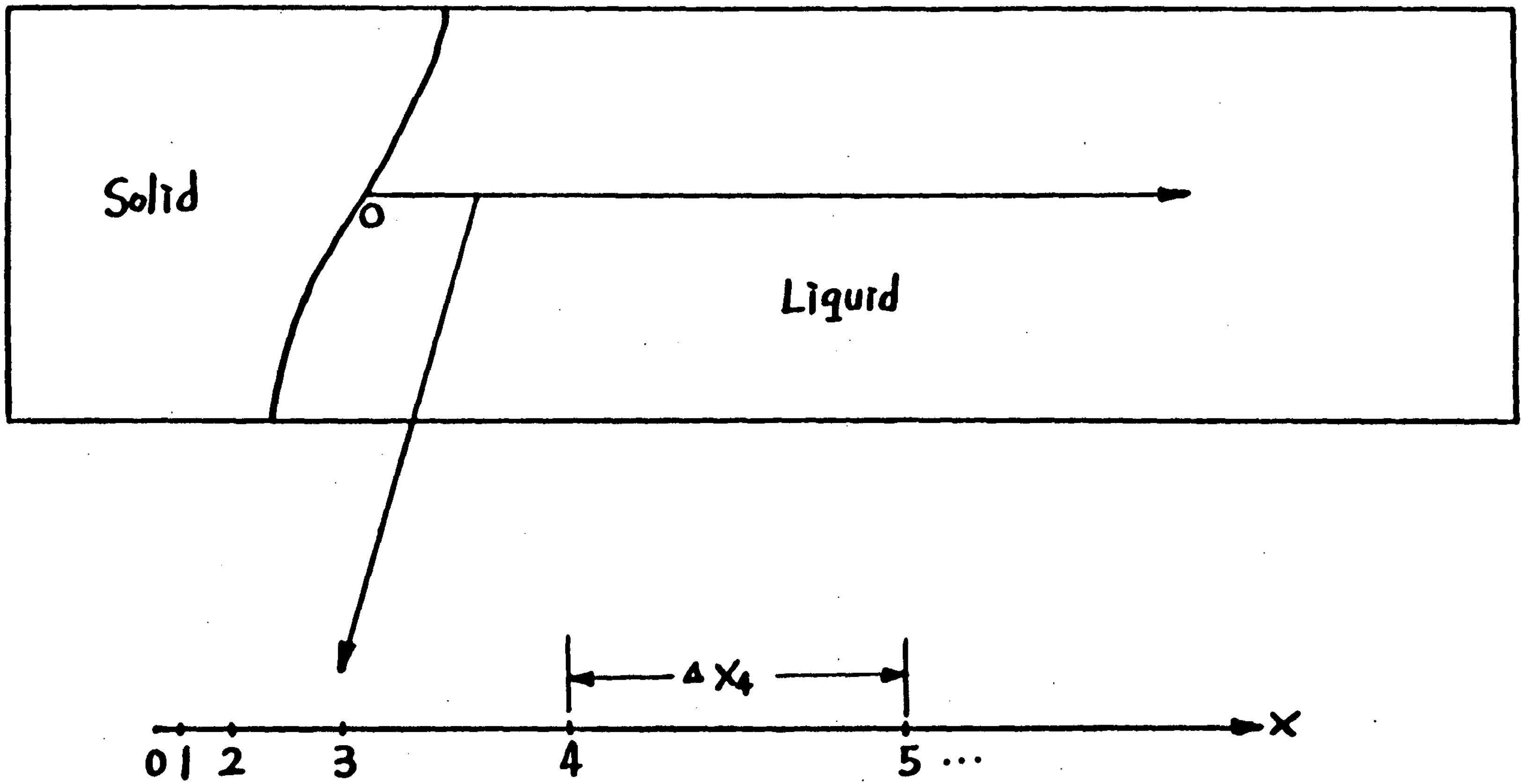


FIGURE 3.5. Exponential meshing of concentration domain, one-dimensional example.

may change the number of nodes along each z-direction and re-mesh the domain at every time step by a method similar to that used in the temperature domains. The alternate method used in this study is simply to squeeze the coordinate of each node on the line parallel to the r-direction by a ratio proportional to the reduction of the corresponding global length. Because the nodes on the interface move with different velocities, except in the planar interface case, the ratio of contraction for each line parallel to the z coordinate is different.

The unique characteristic of a transient problem with increasing domain will be illustrated by a simple example shown in Fig. 3.6. In this figure, the "old" solid lines represent the meshing system at time $t = t_1$, and the "new" dashed lines show the meshing system at time $t = t_1 + \Delta t$. It is noted that at time $t = t_1 + \Delta t$, the temperature, for example, on all nodes in the old meshing system are known. However, in a transient problem we have to know the old temperature on nodes of the new meshing system. Hence the interpolation method has to be employed to find out the old temperatures for these new nodes. As has been indicated before, at every time step all the lines connecting points on the outer boundary and the corresponding points on the interface are parallel to the r-direction. Hence, the old temperature of a new node can be obtained by searching and interpolating only along the line on which it is located. Otherwise, for each node in the new meshing system, it is necessary to identify the old element to which this node belongs, and then use the finite element interpolation functions to calculate its temperature. It is obvious that significant computer time is saved

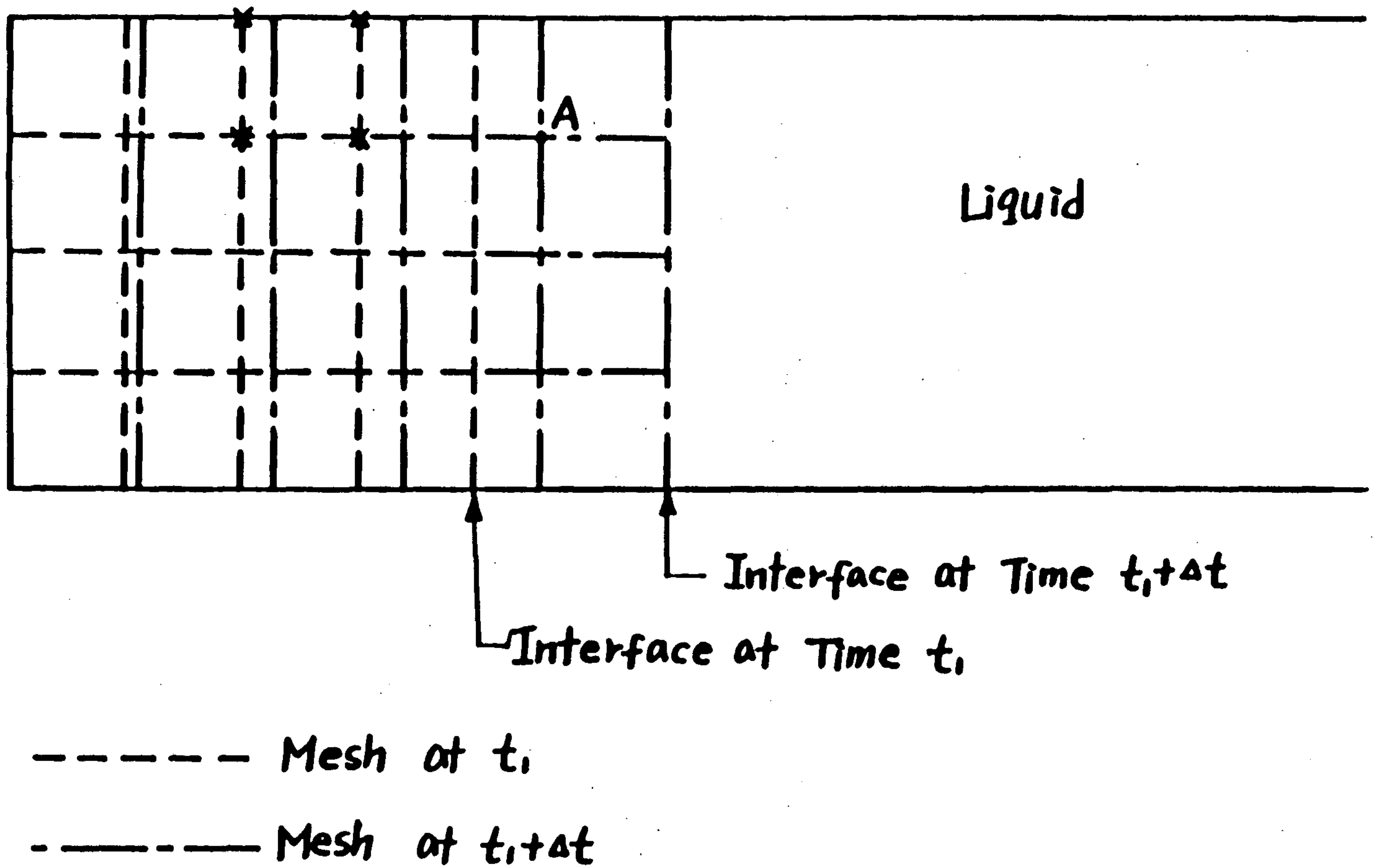


FIGURE 3.6. Special characteristics in meshing the domain of a moving boundary problem.

using our method.

It is interesting to see that at node A, shown in Fig. 3.6, the previous old temperature is not available. The temperature of this node at the previous time step simply did not exist. This situation occurs when the time step is too large or the interface moving velocity is too fast for the grid size adopted. This is an extra constraint in the moving boundary problem, in addition to the limitation of time step imposed by possible numerical stability considerations.

3.8 COMPUTER PROGRAM

The program is written in a modular form, resulting in flexibility and ease of modification. It is composed of about 65 subroutines. The program was initially developed and tried on the Vax 11/750 UNIX system of the Mechanical Engineering Department of the University of California at Berkeley. The results presented in this thesis were obtained from the CDC-7600 at Lawrence Berkeley Laboratory. The typical computer CPU time is 60 minutes per 1000 time steps without iterations. The program can be used in two- or three-dimensional axisymmetric problems with linear or nonlinear properties. The program was designed to deal with four, eight, and nine-node isoparametric elements. Special care was taken to reduce the memory and computation time. Band stiffness matrix was employed, and standard Gaussian LU decomposition, plus forward and backward substitutions, were used to solve the matrix. A listing of the program appears in Appendix 2.

CHAPTER 4:

INTERFACE STABILITY ANALYSIS

4.1 PROBLEM DESCRIPTION

Since, as indicated in the Introduction, we plan to study the morphological stability of a planar interface, a typical dendritic domain taken from Fig. 1.1 (designated by broken lines) is illustrated in Fig. 4.1. Both the upper and lower surfaces in this domain are adiabatic from symmetry considerations. The domain is considered in three-dimensional axisymmetry, which is a typical model of a dendrite. An inertial coordinate system rather than a moving coordinate system attached on the phase interface is chosen and illustrated in Fig. 4.1, where R and Z represent the dimension of domain in the r and z coordinates, respectively; C_i and T_i are the initial concentration and temperature of the solution. The solidifying medium in this study was chosen for illustration purposes to be a saline solution. The thermophysical properties of saline solutions and ice are listed in Table 4.1. As mentioned previously, many analytic studies on solid-liquid interface stability are based on the assumption that the dimension coinciding with the direction of dendritic growth is semi-infinite. Thus the z-dimension of the domain was taken large enough relative to the r-direction to satisfy this assumption.

To apply the general computer program developed in this study to this specific interface stability problem, it is only necessary to specify some

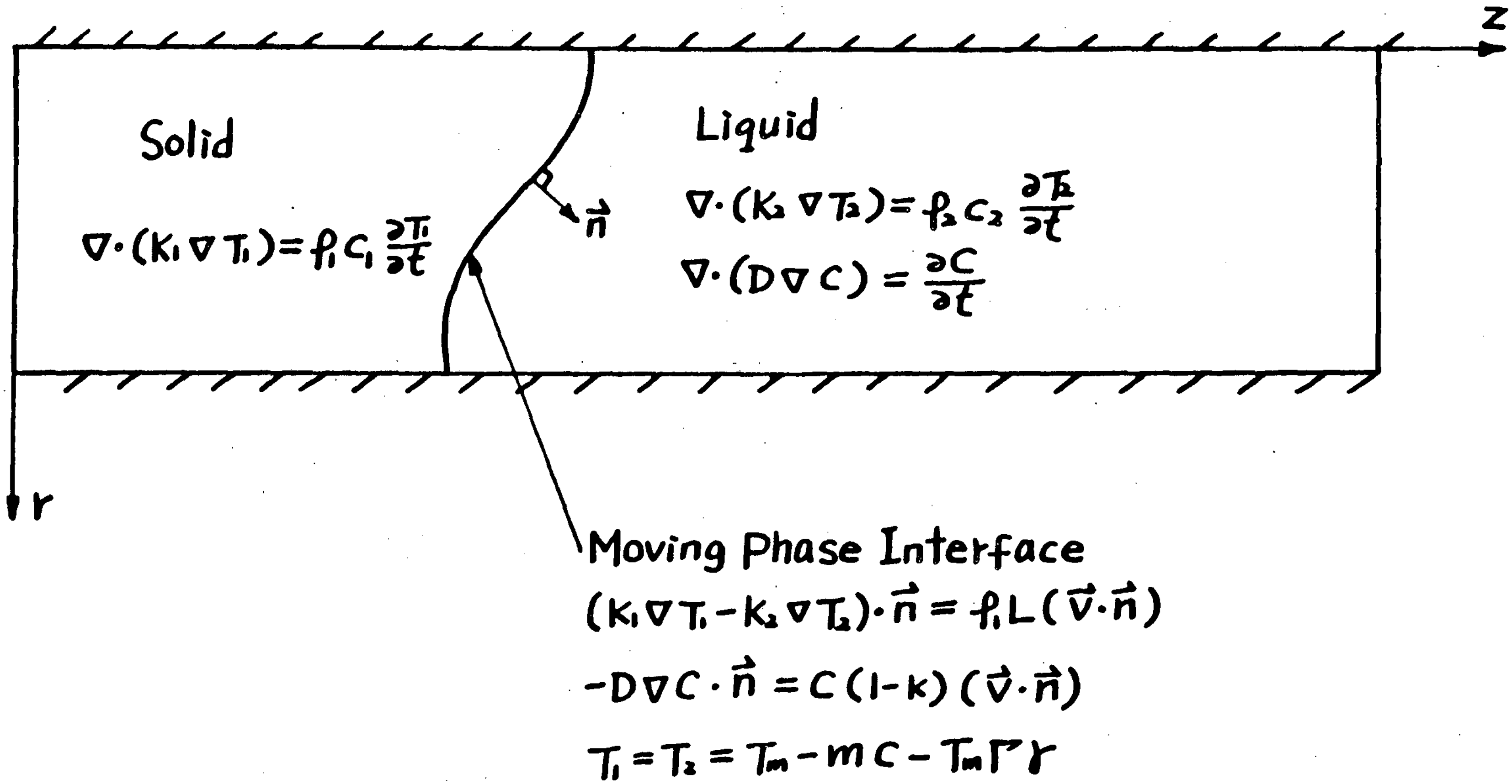


FIGURE 4.1. Domain of a half-dendrite, three-dimensional axisymmetry case.

TABLE 4.1. Thermophysical Properties of Dilute Saline Solution and Ice

	Water	Ice	Units
ρ	999.	999	kg/m^3
k	5.55×10^{-4}	2.25×10^{-3}	kW/mK
C	1.83	4.22	kJ/m^3K
L	353	353	kJ/kg
D	1.18×10^{-9}	-	m^2/sec
m	-1.86	-1.86	k/M
k	Partition coefficient = 0		

parameters at the time of program input data. For example, in the constant properties problem, a parameter is specified such that the subroutine written to solve the nonlinear properties problem will not be called. The boundary conditions are treated in the same way. Since the domain is a three-dimensional axisymmetry, the volume integration in the finite element formulation, dv = r dr dz, is chosen such that r is the average distance of each element from the axis of symmetry. For a two-dimensional domain, r simply equals unity.

4.2 NUMERICAL PERTURBATIONS

To study the stability of a planar interface, two types of numerical perturbations have been used: temperature perturbations on the outer boundary and concentration perturbations on the interface. Each kind of perturbation was imposed in the space and/or time domains. In general, in this study the different types of perturbation have been imposed separately. However, if necessary the computer program can handle simultaneously any combination of the various perturbations. In this study only spatial perturbations of these two types have been emphasized, since the main purpose here is to study the stability of a spatially perturbed planar interface. In fact, some perturbations on the time domain were attempted in this study, but their physical significance needs further investigation. Since any arbitrary functions can be represented as the Fourier series of sinusoidal functions, cosine function was used as the perturbation. The procedures of numerical perturbations are illustrated in this section, but the discussion on the physical meanings of the perturbations and the results will be given in Chapter 5.

4.2.1 Temperature Perturbation on the Outer Boundary

The schematic illustration of spatial temperature perturbation on the outer boundary is shown in Fig. 4.2. On the outer boundary a constant temperature T_∞ below the freezing temperature of the solution is initially imposed. The temperature difference between the outer boundary T_∞ and the interfacial temperature is the driving force for advancing the interface. It is obvious that under this condition the interface will move in such a way that a planar surface is continuously maintained. Then at time $t = t_1$ a sinusoidal temperature perturbation described by

$$T = T_0 + A \cos\left(\frac{\pi r_i}{R}\right), \quad t_1 < t \tag{4.1}$$

is suddenly imposed, where A is the small amplitude of perturbation, R is the global dimension of outer boundary in the r-direction, and r_i is the coordinate of node i on the outer boundary. The number of nodes i is preselected. It was expected that a perturbed interface similar to a sinusoidal shape would be gradually generated during the time when the spatial temperature perturbation was applied. In some of the problems, the perturbation was removed at time $t = t_2$, and the morphology of the interface was continuously examined. Notce that π/R is equivalent to the frequency, and by changing R any kind of wavelength for the perturbation can be obtained.

A different kind of temperature perturbation in the time domain can be obtained through the boundary condition

$$T = T_0 + A' \cos\left(\frac{\pi \Delta t}{T}\right), \quad t_1 < t < t_2 \quad , \tag{4.2}$$

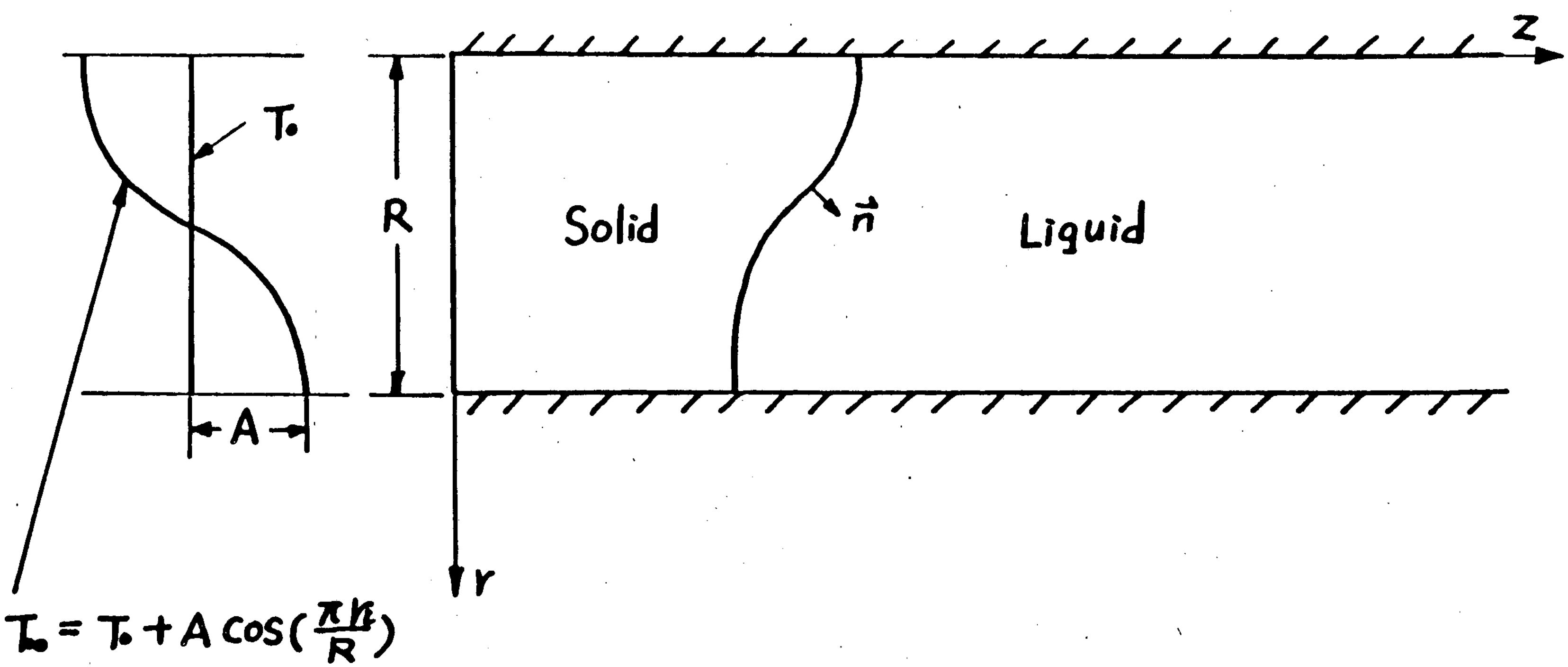


FIGURE 4.2. Temperature perturbation on the outer boundary.

where A' is the small amplitude of perturbation, $T = t_2 - t_1$ is the half period of a sinusoidal function, and $\Delta t = t - t_1$. Here the example only gives a half cycle of cosine wave with an appropriate period of T. This perturbation could be extended in a straightforward manner to any length of time during which arbitrary periods of sinusoidal perturbation is applied. These two perturbations (4.1) and (4.2) can be combined by simple multiplication of the perturbed terms.

4.2.2 Concentration Perturbation on the Interface

A perturbed solid-liquid interface can also be created through a concentration perturbation on the interface. In this case the temperature at the outer boundary is kept at constant value. Similar procedures to those described in the previous paragraph on temperature perturbations are used to initiate a moving planar interface. Then an artificial numerical perturbation of concentration

$$C(r) = C_0(r) + B \cos\left(\frac{\pi r_i}{R}\right), \quad t_1 < t \tag{4.3}$$

is performed on the interface, where $C_0(r)$ is the original concentration distribution along the interface $S(t)$ and B is the small amplitude of the perturbation. The distribution $C_0(r)$ is a constant at the initial perturbation; thereafter, it will be a function of r and z, i.e., it will vary along the interface $S(t)$. The concentration perturbation in the time domain on the interface is similar to that in Eq. (4.2).

In this study it is assumed that the amplitude of all kinds of perturbation is constant. This is not necessarily true, since it also can be a function of time. The results obtained from this study on the effect of various numerical perturbations, discussed in the next chapter, will provide a much deeper understanding of the interface stability phenomena.

CHAPTER 5:
RESULTS AND DISCUSSION

The computer program developed in this work was used to study the stability of a solid-liquid interface during transient solidification processes.

5.1 INTRODUCTION

First the effects on the interface stability of transient temperature fluctuations on the outer surface of the domain were studied. The study was performed for the rectangular geometry shown in Fig. 5.1. The rectangular enclosure contained a liquid solution initially at a constant temperature. The transient solidification process was started by suddenly changing the temperature on one of the narrow walls of the rectangular enclosure to a constant value below the phase transformation temperature. Adiabatic boundary conditions were imposed on the other walls, resulting in a time-dependent propagation of the planar, solid-liquid interface in a direction normal to the constant-temperature wall.

To study the effects of temperature fluctuations on the stability of the moving interface, spatially sinusoidal temperature perturbations were superimposed on the constant temperature boundary for various periods of time and then removed. This was done at different times following the onset of the solidification process. The position and

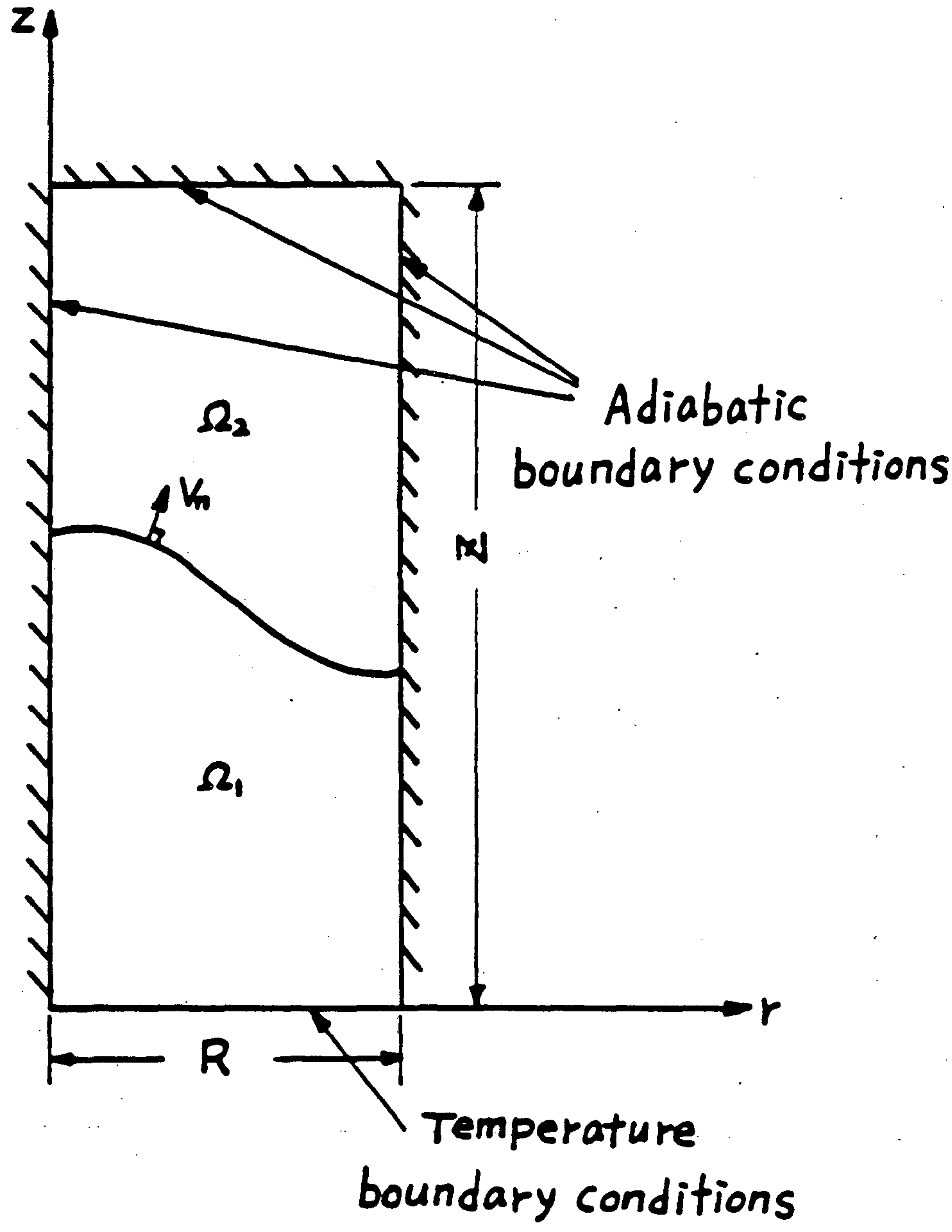


FIGURE 5.1. Computational domain for studying the interface stability.

velocity of the solid-liquid interface and the multi-dimensional temperature and concentration distributions were continuously calculated using the front tracking finite element method developed in this work. The temperature fluctuation superimposed on the constant temperature boundary condition affected the shape of the planar interface, which became perturbed as well. It was anticipated that on a morphologically stable interface the spatial perturbation would disappear after the removal of the temperature perturbation, while on an unstable interface the spatial perturbation would continue to grow. Numerous computer runs were performed for a medium with the thermophysical properties of a saline solution (Table 4.1). Numerical experiments with different length scales and time scales were attempted. The results of all numerical experiments indicate that a planar solid-liquid interface is stable, during the transient solidification process, to temperature perturbations on the outer boundary, i.e., the spatial perturbation of the change of phase interface disappears after the removal of the temperature perturbation. The stability of a planar interface to temperature fluctuations was observed in all the computer runs, including situations in which the liquid in front of the change of phase interface was thermodynamically supercooled. In such a situation, the constitutional supercooling theory predicts an unstable interface.

To illustrate the numerical procedure and the observations described above, one set of typical results for certain geometrical and thermal conditions will be initially presented. The length scales used in this analysis are compatible with experimentally determined values for the dimensions of a perturbed planar interface at the onset of instability (15 μm) during the transient freezing of a saline solution. The length of the narrow wall in the enclosure was, consequently, taken to be 15 μm and the length of the longer wall 2 mm. A large enough ratio between the longer and the narrow wall of the rectangular enclosure was chosen to ensure that a semi-infinite domain could be effectively simulated. The initial concentration of the saline solution in the enclosure was 34.0 gmol/m^3 and the initial temperature was -0.121°C. This temperature corresponds to the phase transformation temperature in a 34.0 gmol/m^3 saline solution. To start the transient solidification process a constant temperature of -2.0°C was imposed on one of the narrow walls of the enclosure. Adiabatic boundary conditions were imposed on all the other walls.

5.2 PLANAR INTERFACE

First, the transient position of the one-dimensional interface and the transient temperature and concentration profiles were calculated using the front tracking finite element method. Figure 5.2 shows typical temperature and concentration distributions in the solid and liquid regions at various times. The results illustrate several well-known physical phenomena, which will be discussed in detail since they are of importance in understanding the morphological stability of solid-liquid interfaces during transient solidification processes.

One of these phenomena is the narrow concentration boundary layer adjacent to the change of phase interface in the liquid region.

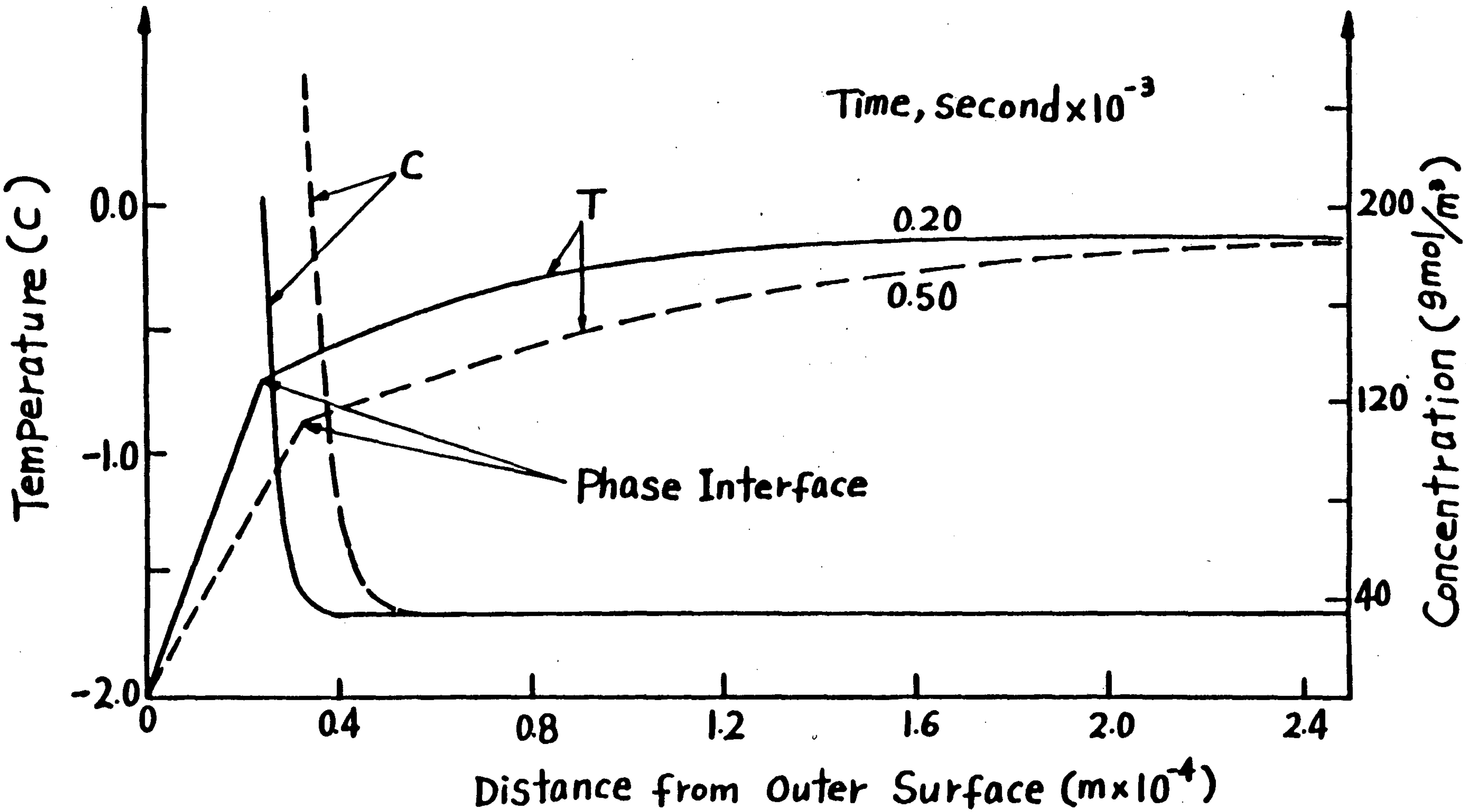


FIGURE 5.2. Temperature and concentration distributions of the solid and liquid; planar interface. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 35.0$ gmol/m³, $T_i = -0.121°C$, $T_\infty = -2.0°C$.

According to the constitutional phase diagram for saline solutions, ice cannot contain any solute. Consequently, saline is rejected in front of the change of phase interface during solidification. The concentration distribution in front of the change of phase interface is affected by two competitive mechanisms. One is the rejection of solute in front of the moving interface, which is directly related to the interfacial velocity and results in an increase in the solute concentration. In the second method, solute is transported away from the interface by the diffusion mechanism, which is proportional to the concentration gradient and the magnitude of the mass diffusion coefficient.

Figures 5.2, 5.3, and 5.4 show a continuous increase in the solute concentration on the change of phase interface, implying that the solute rejection rate exceeds the rate of solute being diffused away. The relative effect of these two competitive mechanisms is of fundamental importance in undertsanding the stability of a planar interface during a transient solidification process.

The phase transformation temperature is inversely related to solute concentration according to the constitutional phase diagram. Consequently, during the transient solidification process analyzed in this work the temperature on the change of phase interface will continuously decrease in time. This can be observed in the temperature distribution curve in Fig. 5.2.

The velocity of the change of phase interface is proportional to the heat conducted through the solid and removed at the outer boundary. The heat removed is directly related to the temperature gradient in the solid region. This temperature gradient is determined by the temperature

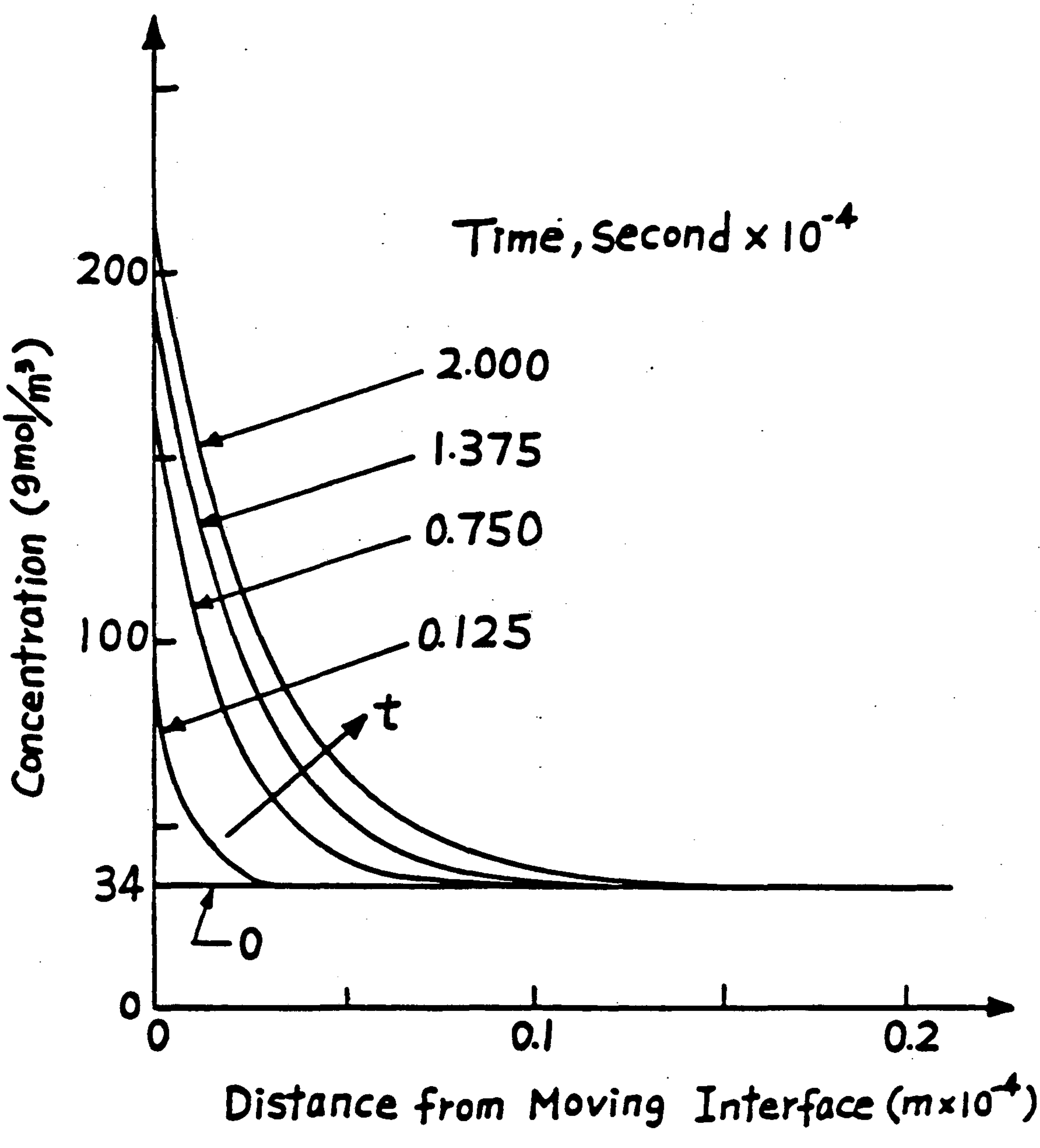


FIGURE 5.3. Concentration distribution in the liquid, planar interface. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121°C$, $T_\infty = -2.0°C$.

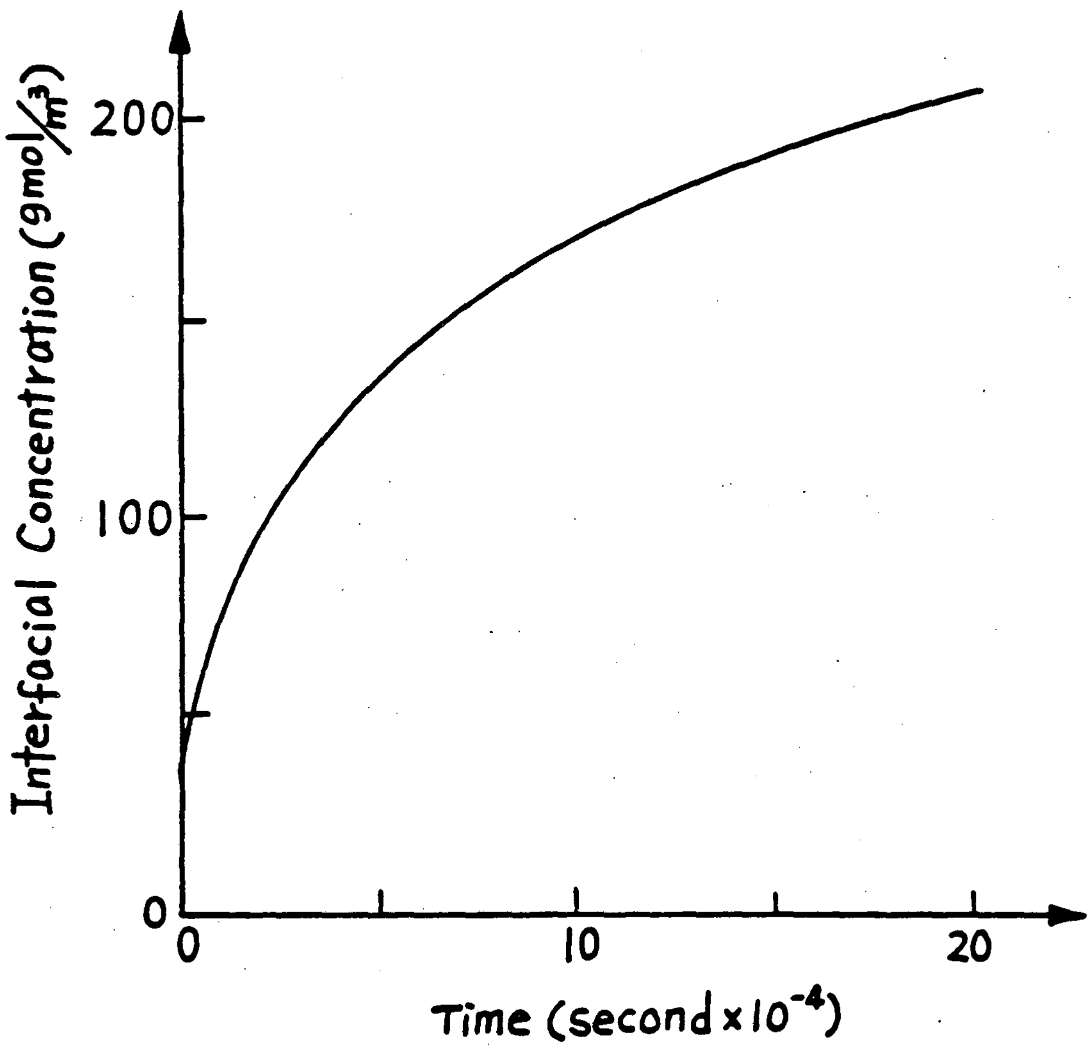


FIGURE 5.4. Interfacial concentration, planar interface. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C, $T_\infty = -2.0$°C.

difference between the outer boundary temperature and the interfacial temperature divided by the solid layer thickness. During the transient solidification process the temperature gradient is continuously attenuated in time due to two factors: the continuous increase in distance between the moving interface and the outer boundary and the decrease in change of phase temperature due to solute accumulation on the interface. This phenomenon is illustrated by the results in Figs. 5.5 and 5.6. In Fig. 5.5 the position of the change of phase interface is plotted as a function of time for the freezing of the 34.0 gmol/m^3 saline solution and for the freezing of pure water. The pure water was at an initial temperature of 0°C and was frozen by imposing a constant temperature of -1.879°C on the outer surface. In Fig. 5.6 the velocity of the change of phase interface is shown as a function of time. The figures show that the solidification process is faster in pure water. As explained above, this is due to the decrease in the change of phase temperature due to solute accumulation on the interface. The observation that solute accumulation on the change of phase interface slows the velocity of the interface during transient solidification processes will be of importance in the forthcoming analysis on the change of phase morphological stability.

Figures 5.7 to 5.10 show results obtained for a solidification process in a rectangular enclosure in which the length of the narrowed wall was taken to be 10 μm and the length of the longer wall was taken to be 80 μm. The initial concentration of saline solution was taken to be 34.0 gmol/m^3 and the initial temperature was -0.121°C. A constant

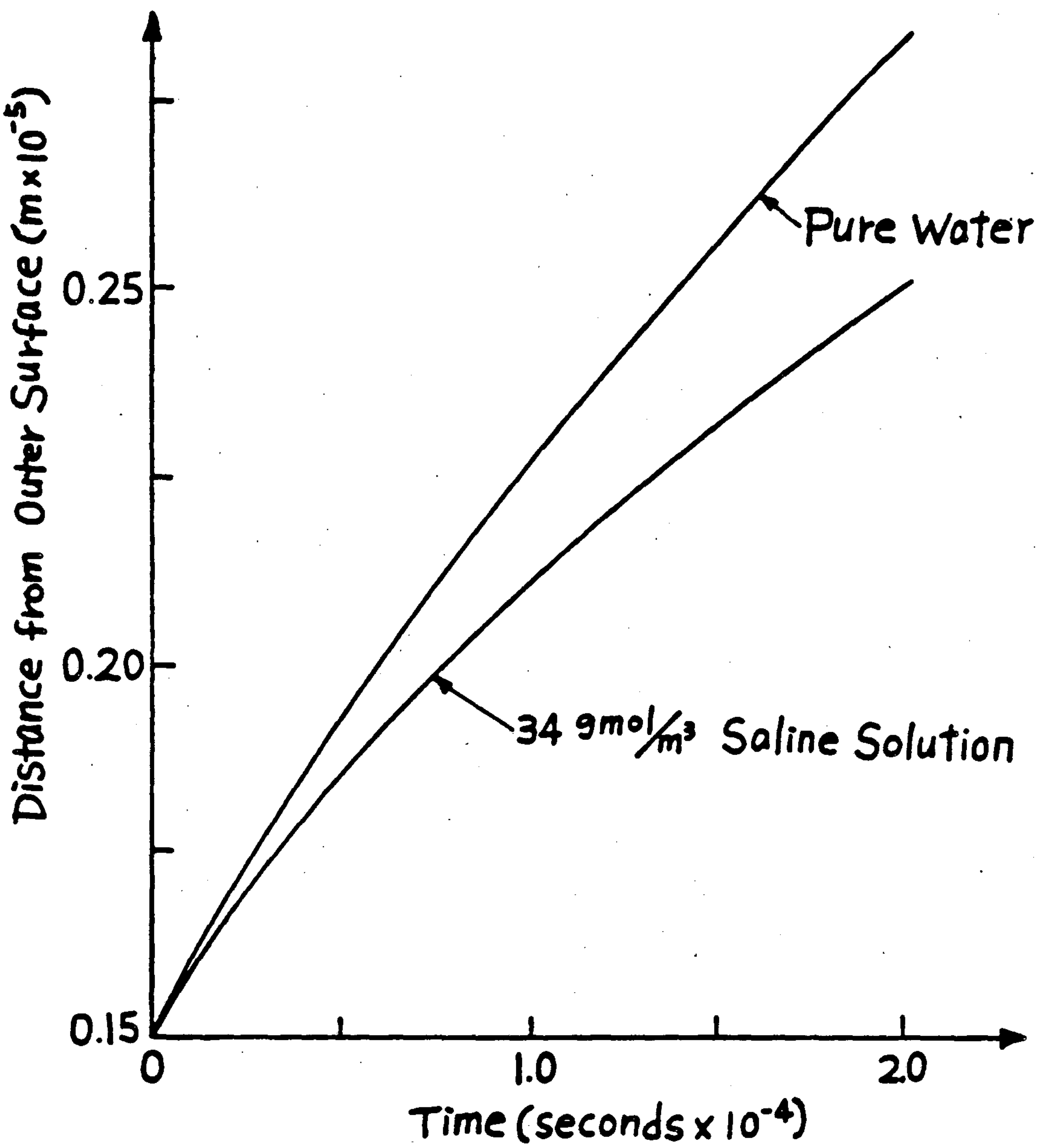


FIGURE 5.5. Interfacial position, planar interface. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121°C$, $T_\infty = -2.0°C$.

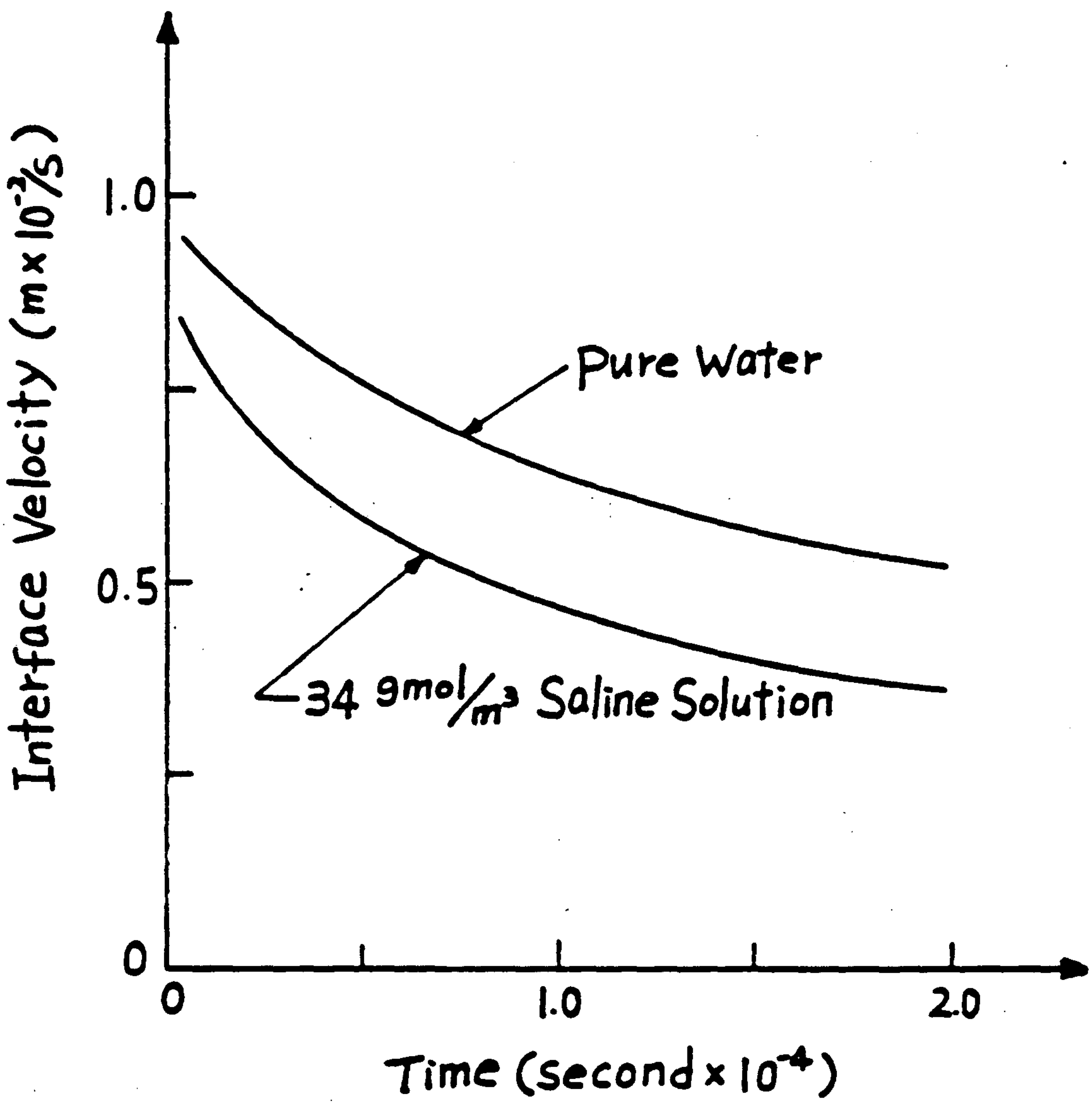


FIGURE 5.6. Interfacial velocity, planar interface. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121°C$, $T_\infty = -2.0°C$.

temperature of -0.3°C was imposed on one of the narrow walls of the domain. Despite the much lower temperature gradient in the solid region, essentially similar phenomena to those observed in the previous case occurred. Figures 5.7 and 5.8, showing the concentration distribution in the liquid region and the concentration on the interface, indicate a continuous increase in the concentration on the interface. The rate of concentration increase is slower, however, than in the previous example. This can be explained by the lower velocity of the solidification process caused by the lower temperature gradient. The lower velocity can be observed by comparing Figs. 5.9 and 5.10 with Figs. 5.5 and 5.6. Despite this difference, the fundamental behavior remains unchanged, i.e., the solute accumulation on the interface and the distance of the change of phase interface from the outer surface cause a continuous decrease in the change of phase interface velocity.

Figure 5.2 shows that in the liquid region the concentration gradient is much steeper than the temperature gradient. This well-known phenomenon is directly related (according to the various stability criteria) to the solid-liquid phase transformation interface instability.

The "constitutional supercooling" stability theory predicts that an interface will become unstable if the solute in front of the change of phase interface is thermodynamically supercooled. This can be expressed by the relation

$$\frac{m \frac{\partial C}{\partial n}}{\frac{\partial T}{\partial n}} > 1 \quad , \tag{5.1}$$

with the concentration and temperature gradients evaluated on the

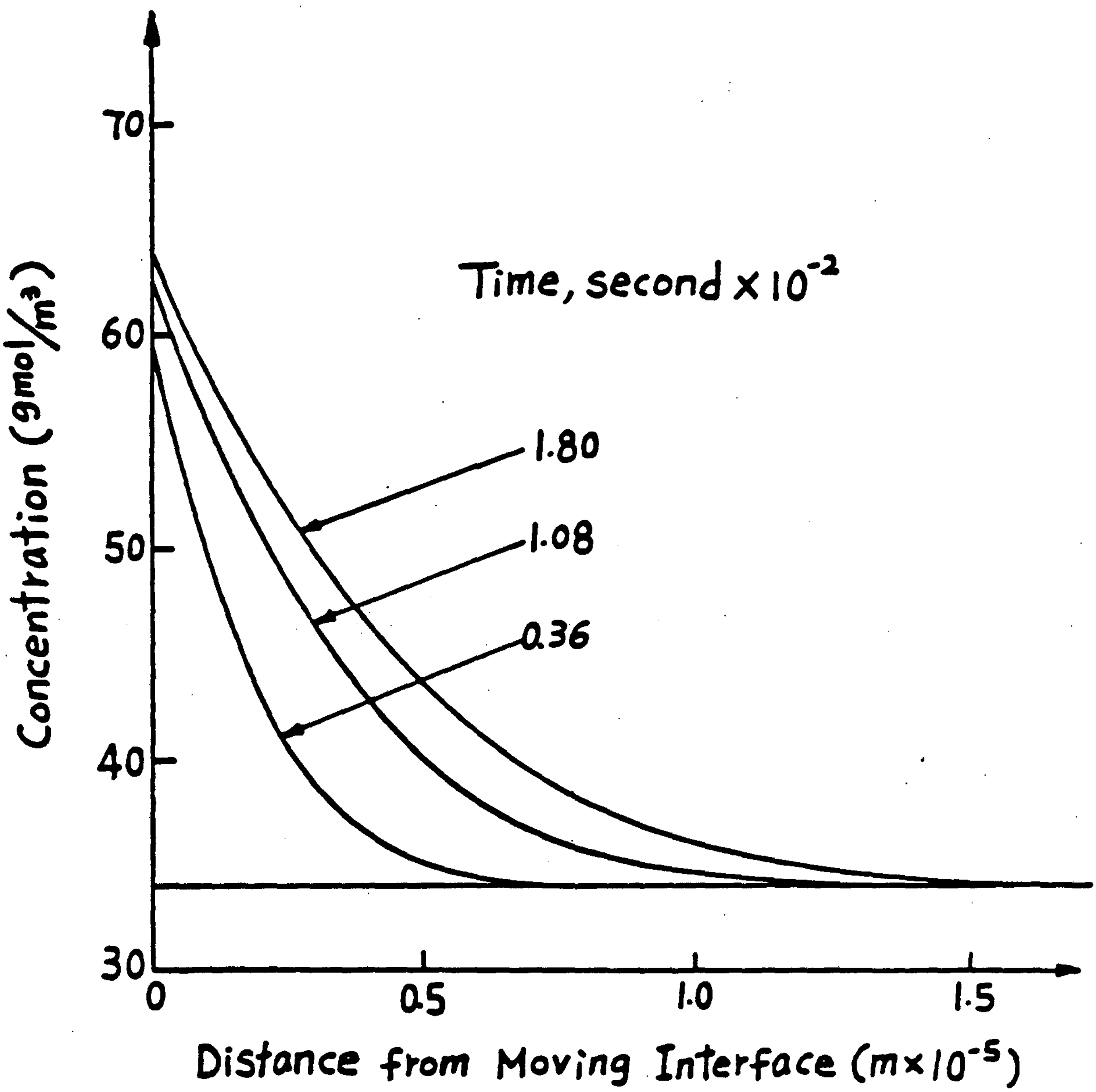


FIGURE 5.7. Concentration distribution in the liquid, planar interface. $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, C_i = 34.0 gmol/m^3, T_i = -0.121°C, T_∞ = -0.3°C.

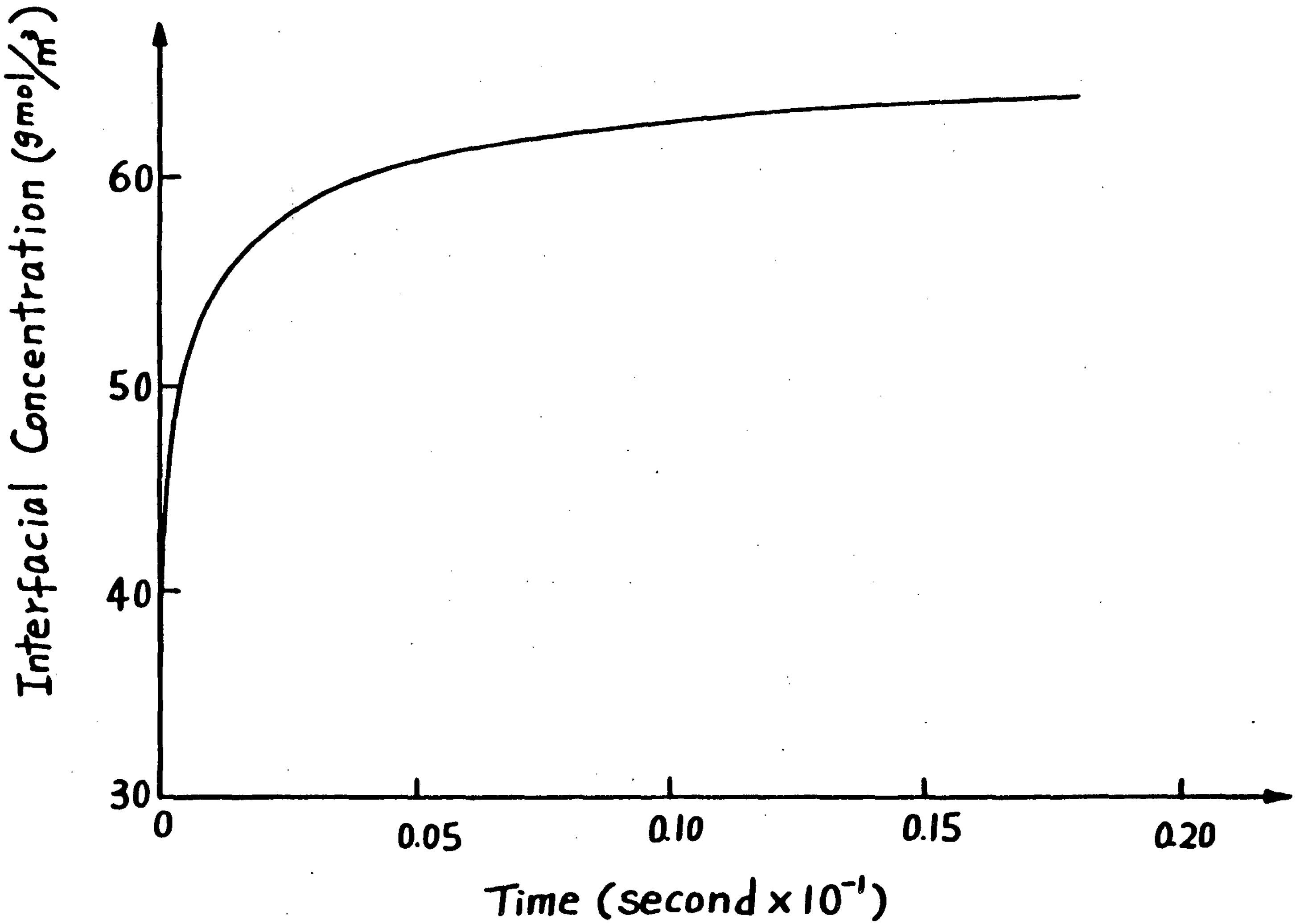
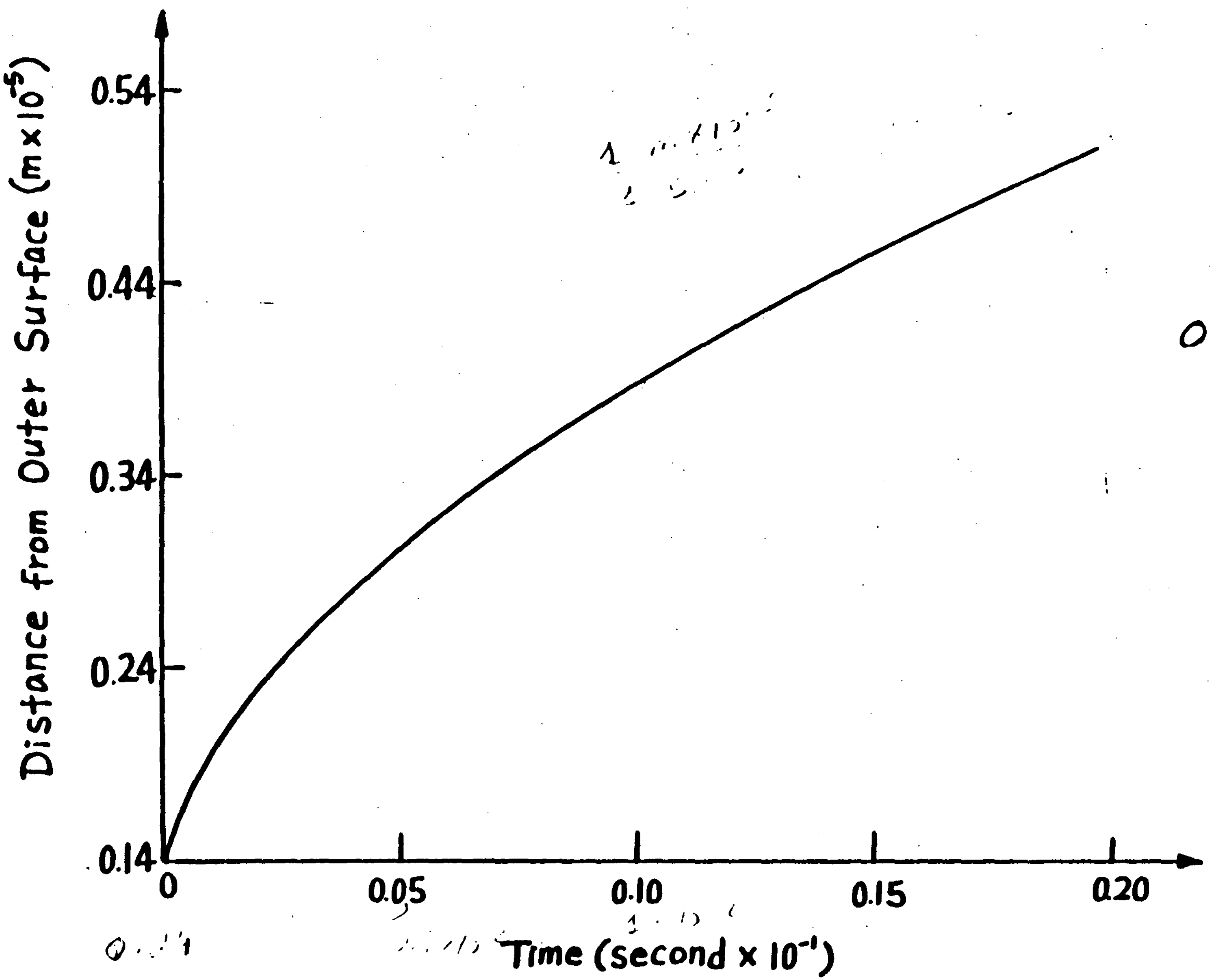


FIGURE 5.8. Interfacial concentration, planar interface. $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C, $T_\infty = -0.3$°C.



$\frac{0.1 \times 10^{-5}}{10^{-2}}$

10^{-4}

FIGURE 5.9. Interfacial position, planar interface. $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C, $T_\infty = -0.3$°C.

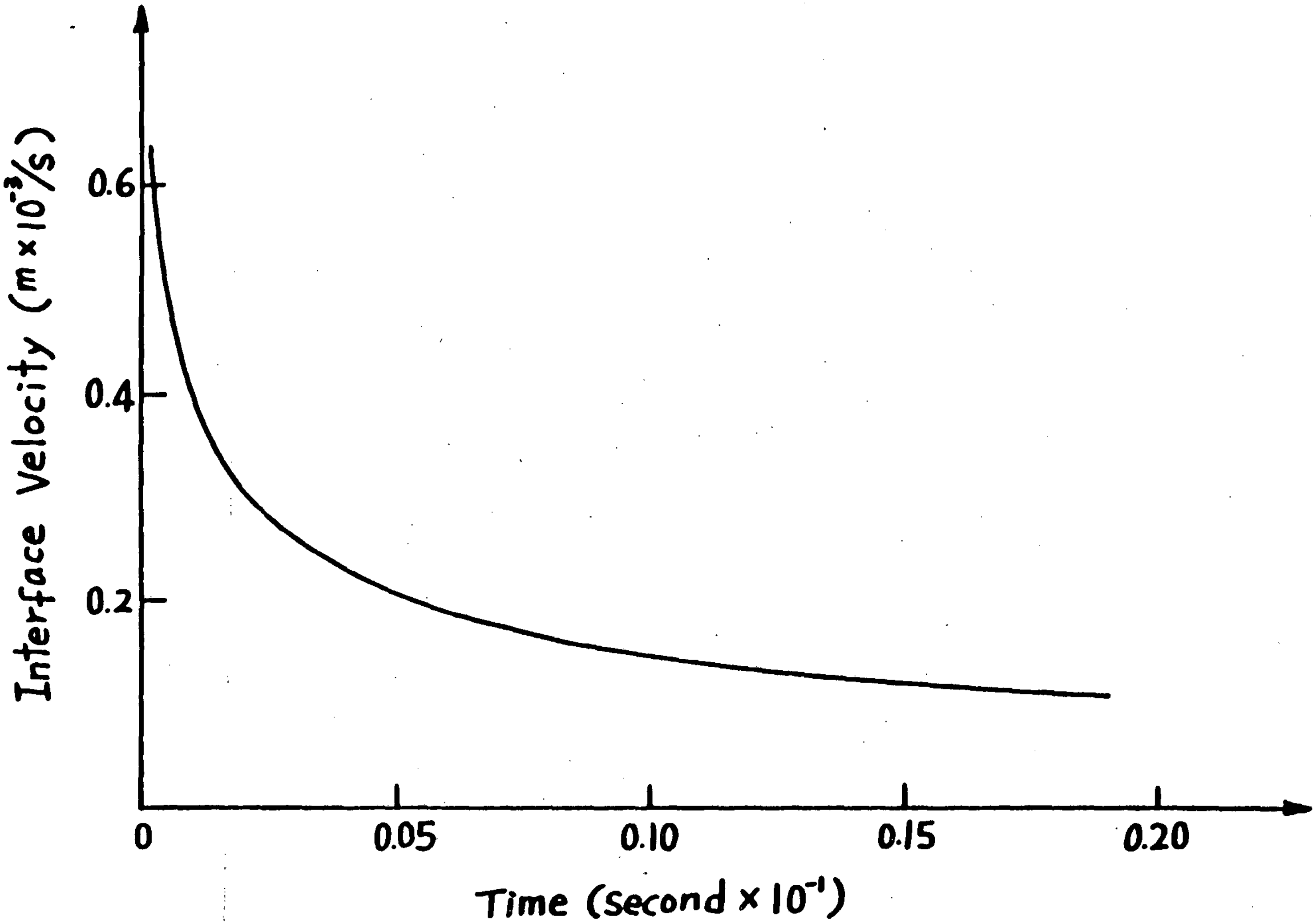


FIGURE 5.10. Interfacial velocity, planar interface. $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121°C$, $T_\infty = -0.3°C$.

interface in the liquid region in a direction normal to the change of phase interface.

According to the M-S general stability criteria, a surface will become unstable if

$$\frac{m\,\frac{\partial C}{\partial n}}{\left(\frac{K_1\,\frac{\partial T_1}{\partial n} + K_2\,\frac{\partial T_2}{\partial n}}{K_1 + K_2}\right)} > 1 \quad , \tag{5.2}$$

where the concentration and temperature gradients are evaluated on the interface in a direction normal to the change of phase interface. The M-S stability criterion presented here is for the special situation in which capillary effects are neglected. It should be emphasized that the M-S criterion was brought up for completeness only. The criterion is not applicable to the situation discussed here since several of the major assumptions are different in this model. The M-S criterion pertains to a solidification process that is steady in a moving frame of reference and inifinite in domain, whereas the analyzed problem is transient in a finite domain.

5.3 TEMPERATURE PERTURBATION ON THE OUTER BOUNDARY

The second step in our analysis was to superimpose on the outer boundary a spatially sinusoidal temperature perturbation. Here we will discuss the results obtained when such a perturbation, with a magnitude of $-0.02 \cos(\pi r_i/R)$ °C, was imposed on 0.5×10^{-4} sec after the onset of the first solidification process. A perturbation with a magnitude of $-0.01 \cos(\pi r_i/R)$ °C was imposed 0.24×10^{-2} sec after the onset of the second case solidification process. At that instant in time the ratio in Eq. (5.1) and (5.2) was on the order of 10^3, indicating that the liquid adjacent to the change of phase interface was supercooled and the interface unstable, according to considerations of equilibrium thermodynamics. The results were obtained using the front tracking finite element method developed in this work. Since the stability criteria do not include capillary effects, the stabilizing effect of capillarity in this example was not included. However, the computer program utilized can incorporate this effect. The dendrites of the liquid and the solid were also assumed to be the same so as not to introduce convection effects.

Figures 5.11 and 5.12 show the location of the solid-liquid interface relative to that of the central node on the interface during the freezing of a saline solution and of water in the first case. Figure 5.13 shows the solid-liquid interface in the second case. The location of the interface is shown at different times after the perturbation was imposed. Figs. 5.11 and 5.13 present results by continuously imposing a temperature perturbation on the outer boundary.

The results in Fig. 5.12 were obtained by imposing the temperature perturbation 0.5×10^{-4} sec after the onset of the solidification process, followed by removal of temperature perturbation 0.5×10^{-4} sec later. Figures 5.11 and 5.13 indicate that the temperature perturbation resulted in a spatial perturbation on the change of phase interface in both the saline solution and the pure water. It is interesting to notice that the amplitude of the perturbation is much larger in pure water.

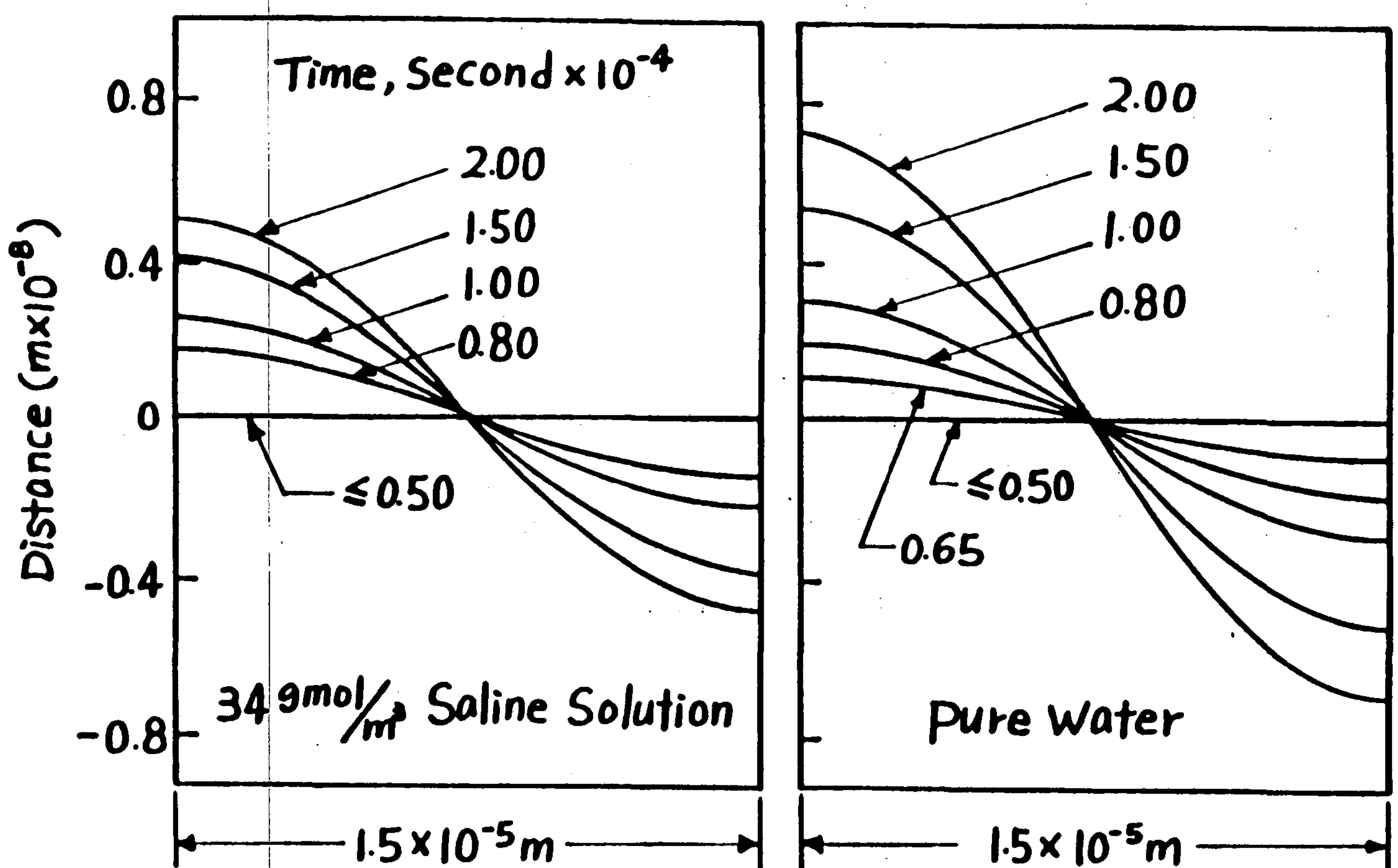


FIGURE 5.11. Interfacial position, temperature perturbation. T_∞ = $-2.0 - 0.02 \cos(\pi r_i/R)$°C at $0.5 \times 10^{-4} < t < 2.0 \times 10^{-3}$ S. $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m³, $T_i = -0.121$°C.

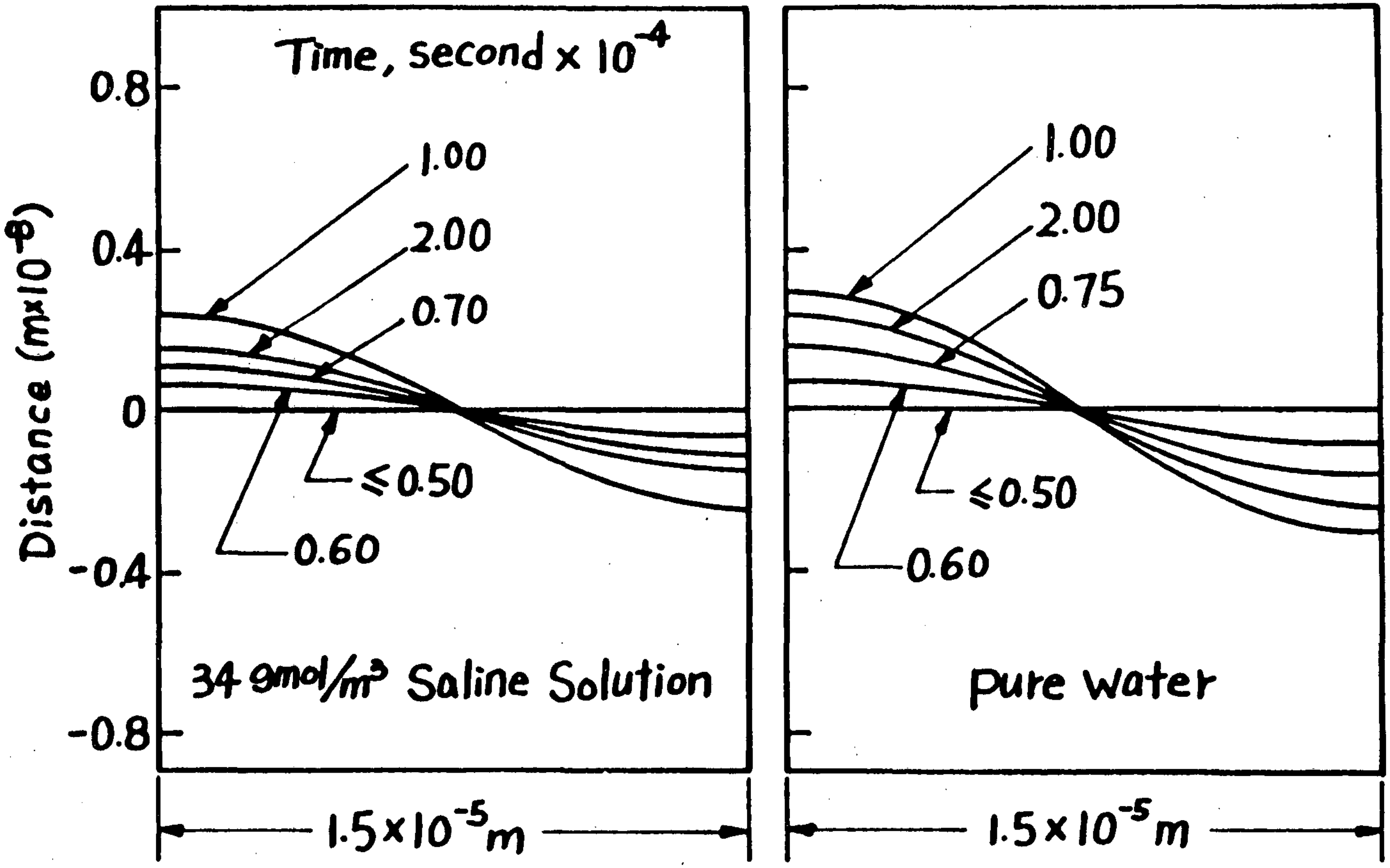


FIGURE 5.12. Interfacial position, temperature perturbation. $T_\infty = -2.0 - 0.02 \cos(\pi r_i/R)$°C in $0.5 \times 10^{-4} < t < 1.0 \times 10^{-3}$ S, $T_\infty = -2.0$°C in $1.0 \times 10^{-4} < t$ S, $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m³, $T_i = -0.121$°C.

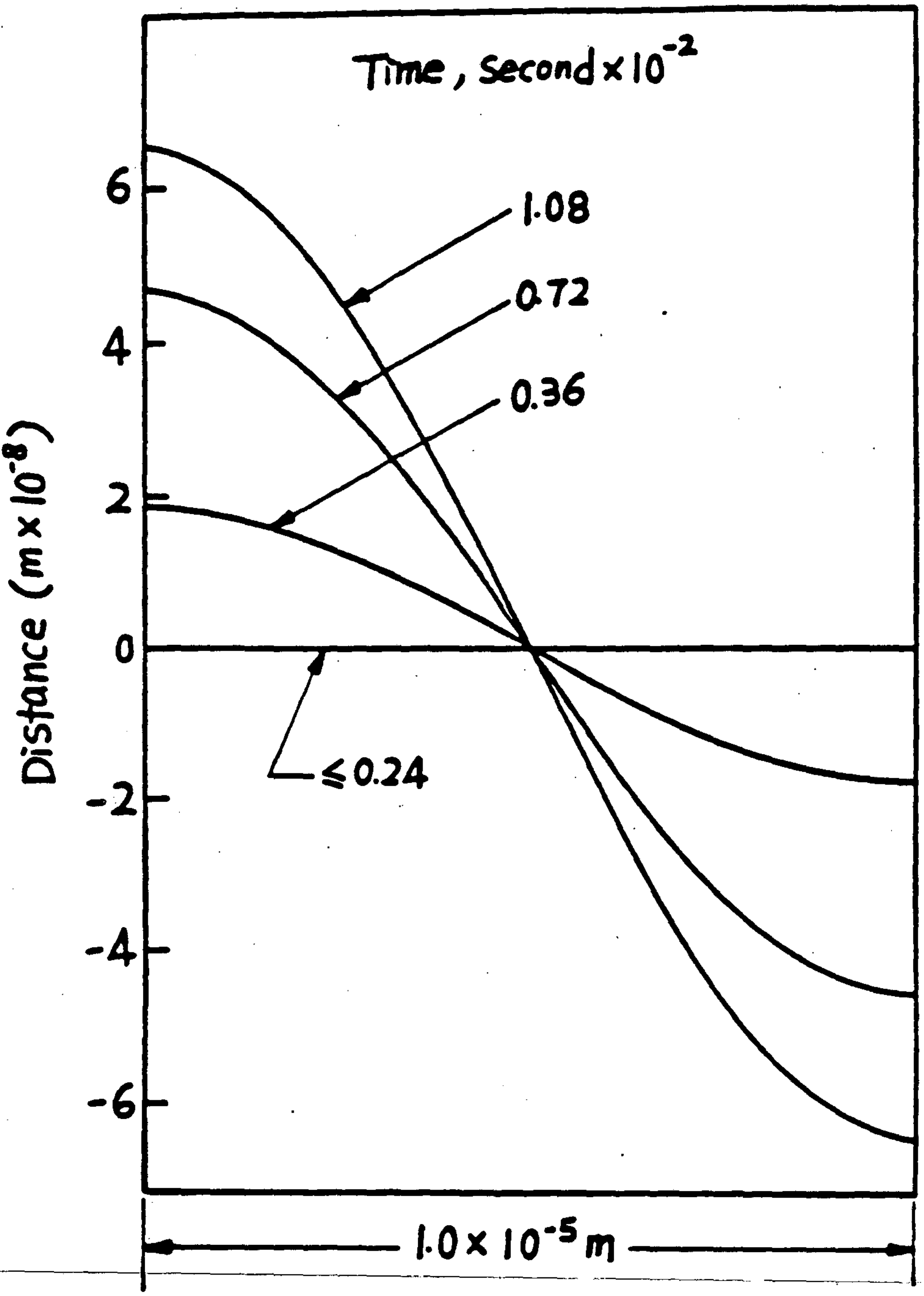


FIGURE 5.13. Interfacial position, temperature perturbation. $T_\infty = -0.3 - 0.01 \cos(\pi r_i/R)$°C, in $0.24 \times 10^{-2} < t < 0.192 \times 10^{-2}$ S, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C.

The most important results of this analysis is probably that shown in Fig. 5.12. This figure indicates that after the temperature perturbation was removed, the change of phase interface spatial perturbation disappeared and the interface became planar again. The interface became planar more quickly in the saline solution than in pure water. This result indicates that the change of phase interface during a transient solidification process in a saline solution is stable to temperature perturbations on the outer surface. Furthermore, the interface is dynamically stable in situations in which the solution in front of the change of phase interface is supercooled and should be unstable from considerations of equilibrium thermodynamics. Following is an explanation of these results using Figs. 5.14-5.18.

Figures 5.14 and 5.15 show the concentration distribution on the change of phase interface relative to the concentration on the central node on the interface during the solidification processes described in Figs. 5.11-5.13. Figures 5.16-5.18 show the normal velocity of the solid-liquid interface relative to that of the center node on the interface during the two solidification processes in Figs. 5.11-5.13, respectively. The concentration distribution shown in Figs. 5.14 and 5.15 is of major importance in understanding the morphological stability of the change of phase interface during the analyzed transient solidification process.

According to concepts of static thermodynamic equilibrium, the change of phase interface is supposed to be unstable when the solute in front of the change of phase interface is thermodynamically supercooled. From considerations of static thermodynamic equilibrium, if the change

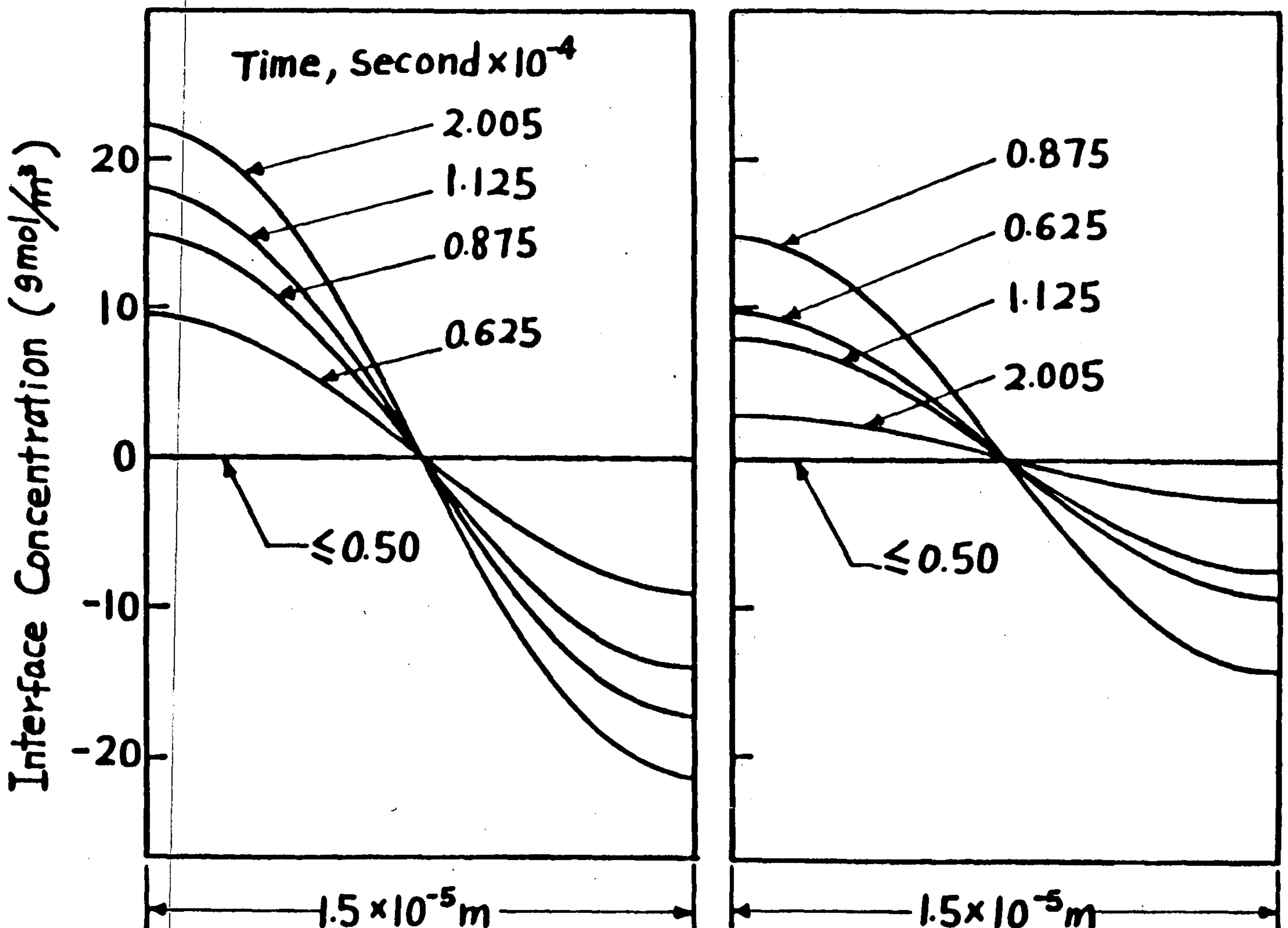


FIGURE 5.14. Interfacial concentration, temperature perturbation. T_∞ = $-2.0 - 0.02\cos(\pi r_i/R)$°C in $0.5\times10^{-4} < t < 2.005\times10^{-4}$ S, R = 0.15×10^{-4} m, Z = 0.2×10^{-3} m, C_i = 34.0 gmol/m³, T_i = -0.121°C.

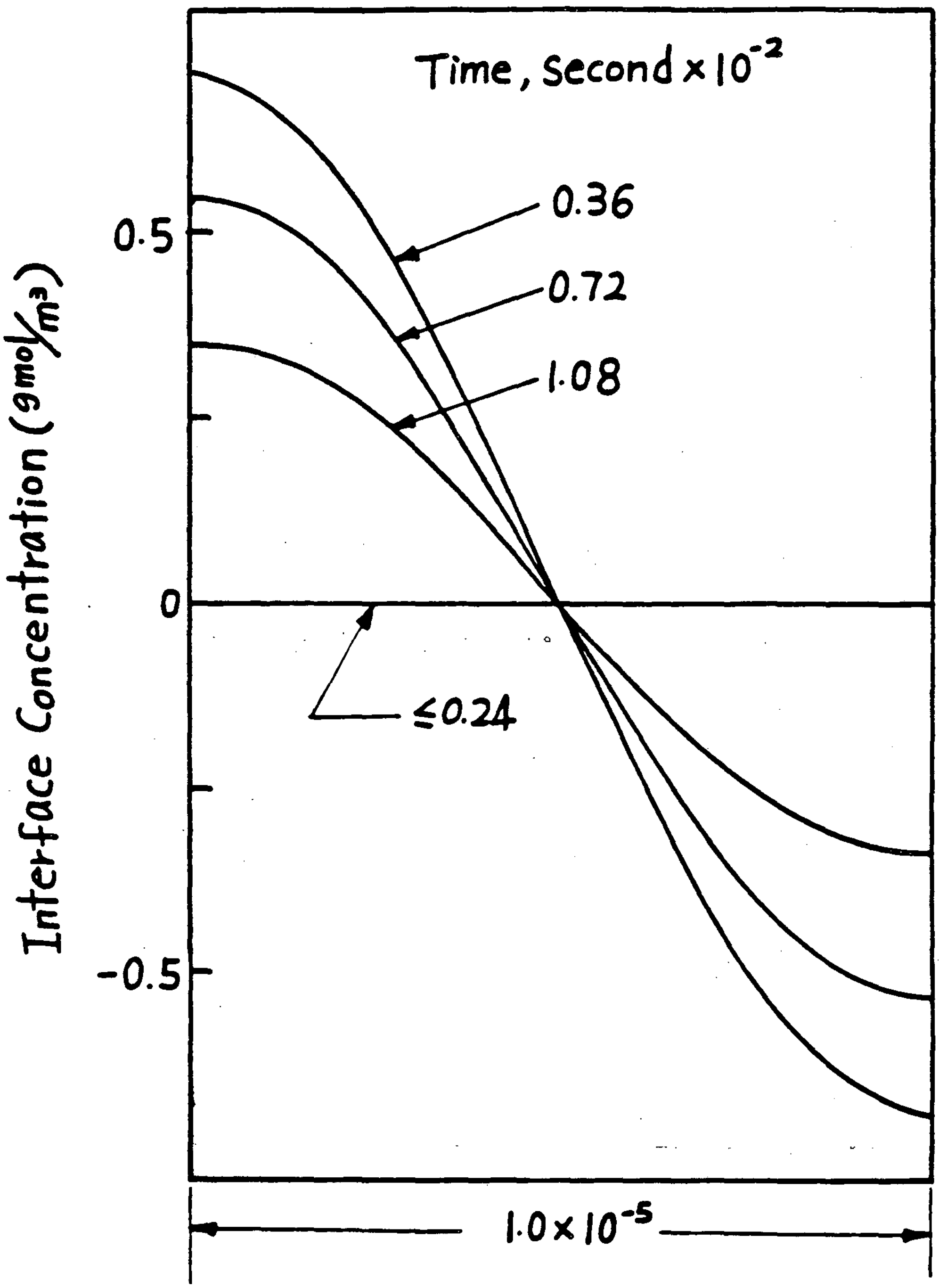


FIGURE 5.15. Interfacial concentration, temperature perturbation. $T_\infty = -0.3 - 0.01\cos(\pi r_i/R)$°C in $0.24 \times 10^{-2} < t < 0.192 \times 10^{-2}$ S, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m³, $T_i = -0.121$°C.

of phase interface is perturbed, the "tip" of the perturbed interface will suddenly find itself in the supercooled region surrounded by a lower solute concentration than the "groove." As a consequence, the tip will grow faster and the planar interface will break. Our rigorous transient numerical analysis shows that during a dynamic transient solidification process, such a phenomenon cannot occur. We would like to emphasize again that this analysis deals with a specific transient solidification process to which the M-S stability criterion cannot be applied.

Figures 5.14 and 5.15 show that the concentration profile on the interface obtained through a rigorous and exact analysis of the dynamic solidification process is different from that assumed in the static thermodynamic equilibrium stability criterion. The concentration of solute is actually higher at the tip of the perturbed interface, i.e., the point furthest away from the outer surface on which the transient temperature boundary condition was imposed. The concentration is lowest in the groove, the point on the interface closer to the outer surface.

This result, although different from that assumed in the static thermodynamic equilibrium stability criterion, is consistent with the intuitively obvious results obtained previously for the planar solidification process and shown in Figs. 5.2-5.4 and 5.7. These figures indicate that during the transient planar solidification process the solute accumulation in front of the change of phase interface is affected by the solute rejection mechanism due to the transient solidification process and by the solute diffusion mechanism in the liquid. Since

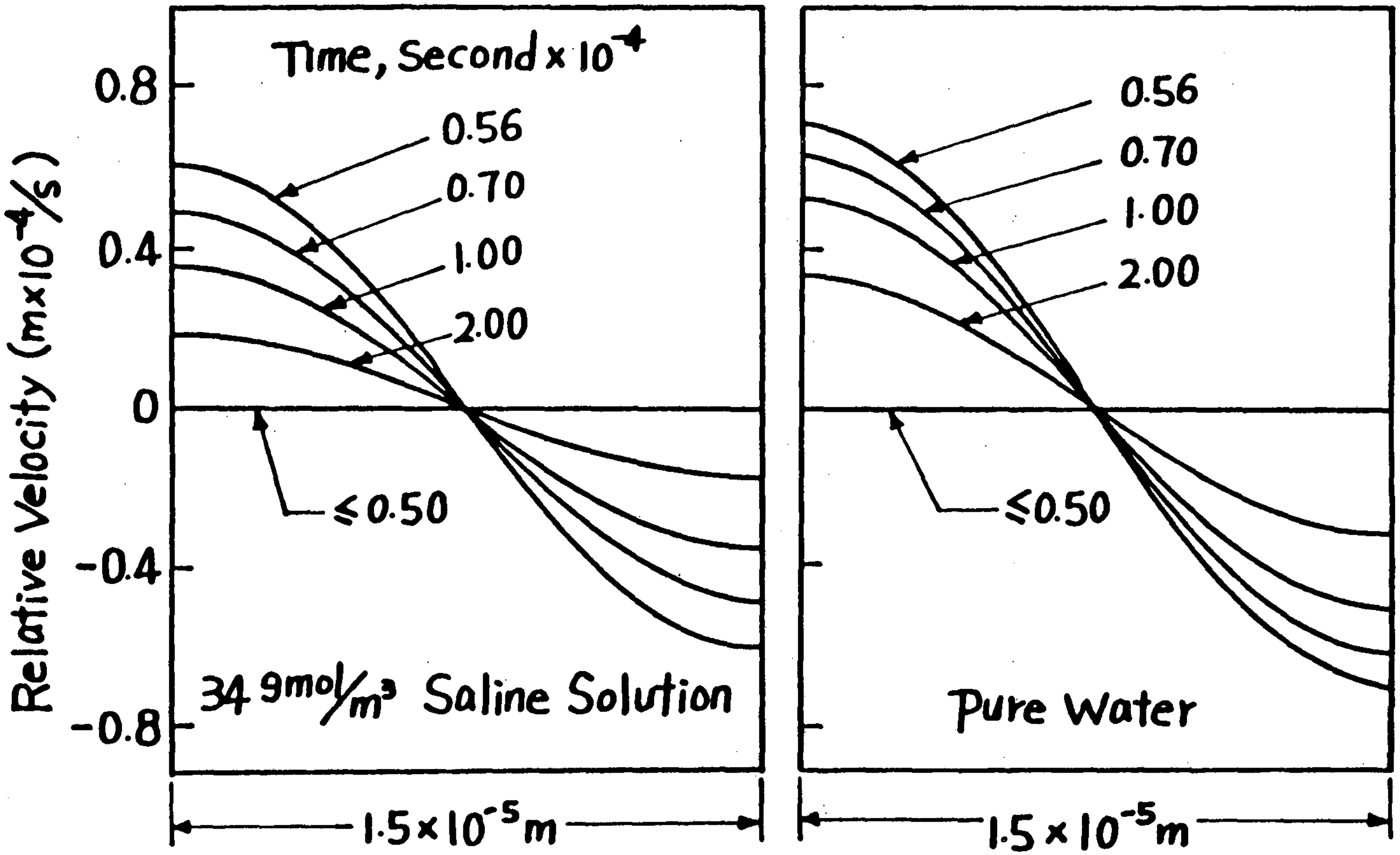


FIGURE 5.16. Interfacial velocity, temperature perturbation. T_∞ = -2.0 - 0.02 cos $(\pi r_i/R)$°C in $0.5 \times 10^{-4} < t < 2.0 \times 10^{-4}$ S, R = 0.15×10^{-4} m, Z = 0.2×10^{-3} m, C_i^l = 34.0 gmol/m³, T_i = -0.121°C.

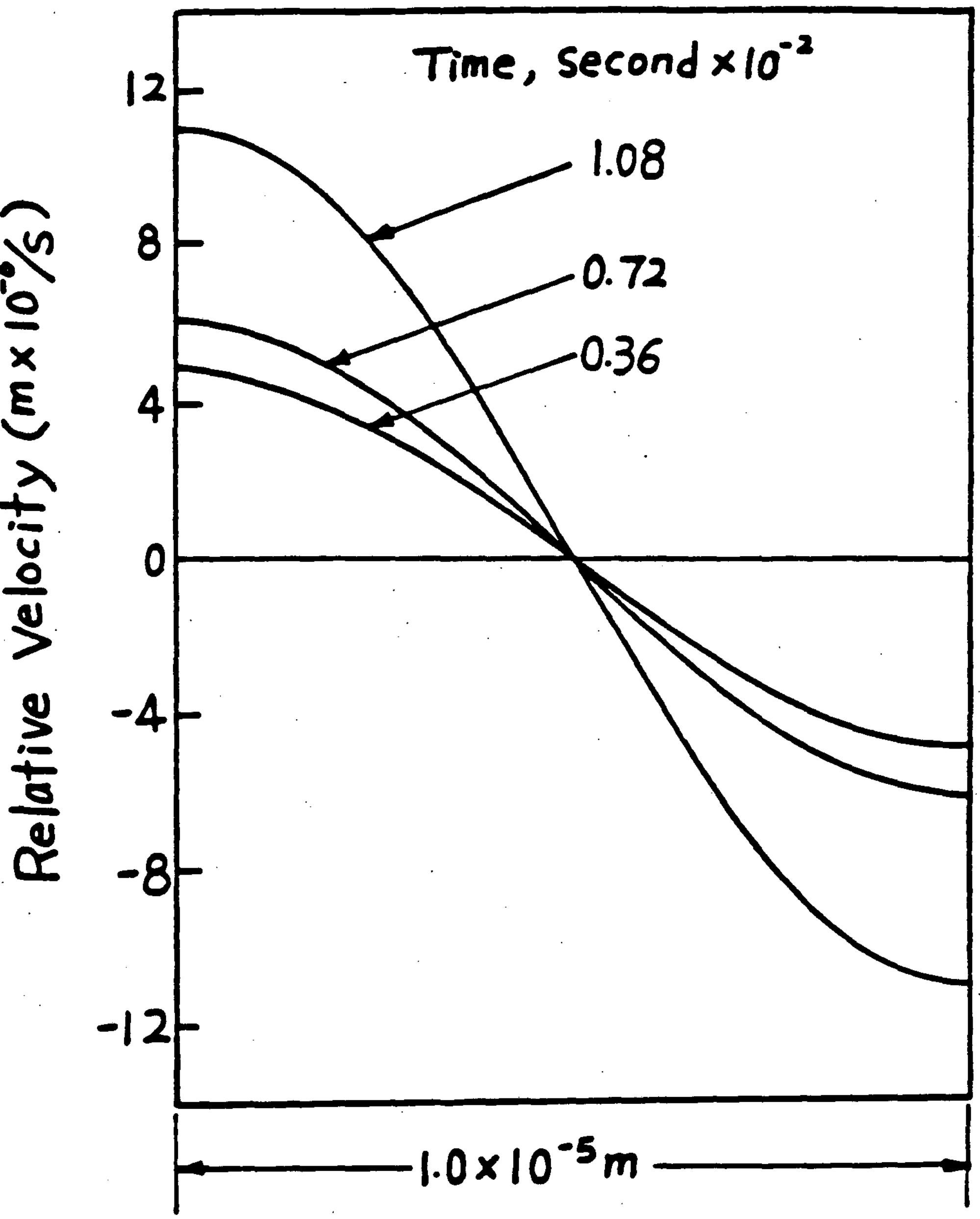


FIGURE 5.17. Interfacial velocity, temperature perturbation. $T_\infty = -0.3 - 0.01 \cos(\pi r_i/R)$°C in $0.24 \times 10^{-2} < t < 0.192 \times 10^{-2}$ S, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C.

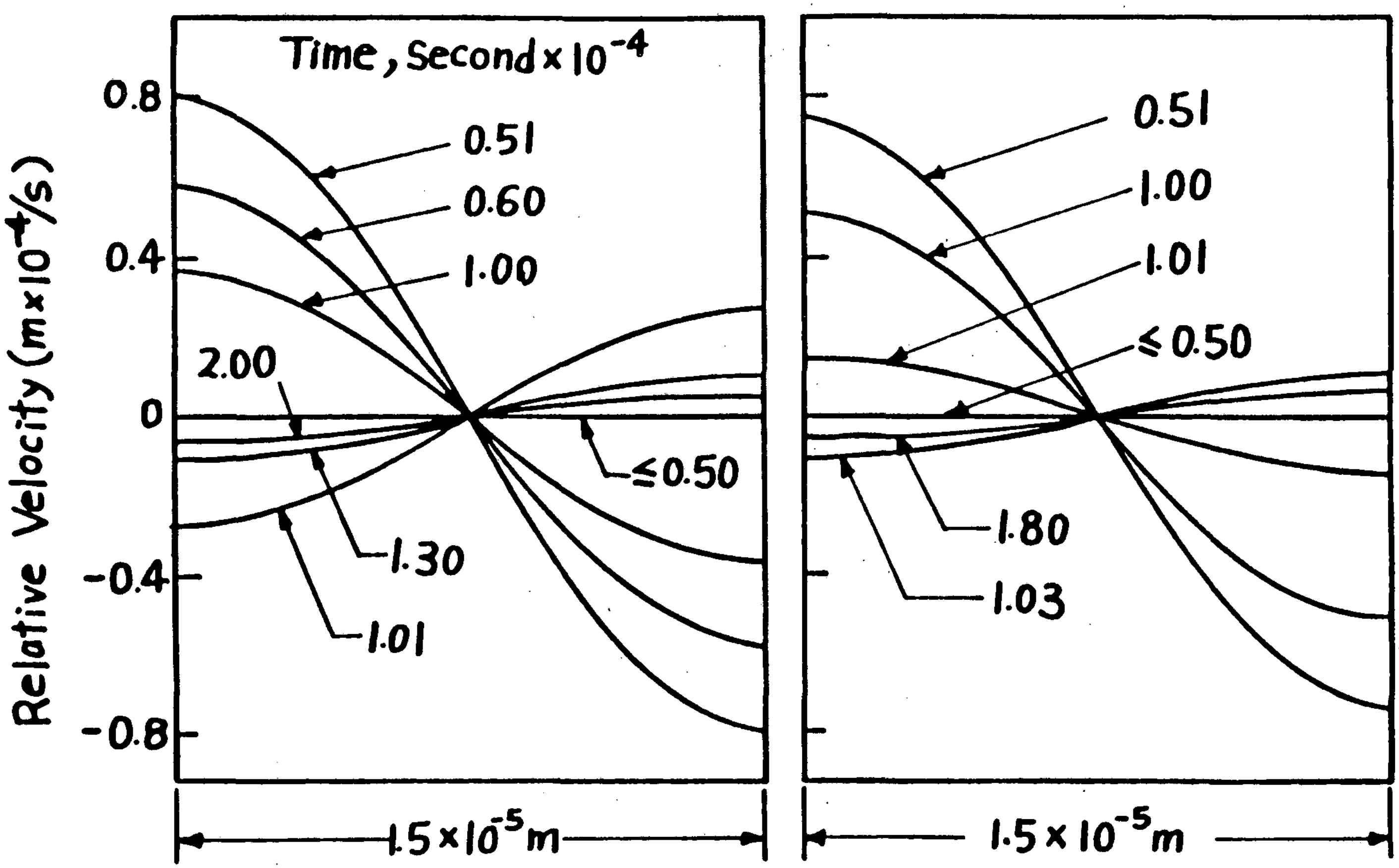


FIGURE 5.18. Interfacial velocity, temperature perturbation. $T_\infty = -2.0 - 0.02 \cos(\pi r_i/R)$°C in $0.5 \times 10^{-4} < t < 1.0 \times 10^{-4}$ S, $T_\infty = -2.0$°C in $1.0 \times 10^{-4} < t$ S, $R = 0.15 \times 10^{-4}$ m, $Z = 0.2 \times 10^{-3}$ m, $C_i = 34.0$ gmol/m^3, $T_i = -0.121$°C.

during the analyzed transient solidification process the solute rejection rate exceeds the rate of solute being diffused away, i.e., the time scale of the solidification process is much shorter than that of the mass diffusion process, the solute will continuously accumulate on the interface. Therefore, the further the planar change of phase interface is from the outer surface, the higher the solute concentration on the interface.

The temperature perturbation superimposed on the outer surface temperature resulted in a gradually growing spatial perturbation on the change of phase interface. This is similar to the planar solidification case, since in this problem the time scale of the solidification process and the consequent solute rejection rate are much shorter than that of the mass diffusion process. The solute concentration on the tip of the perturbed interface, which at any instant in time is farther from the outer surface, is higher than that in the groove, which is closer to the outer surface. This results illustrates the importance of incorporating the transient dynamic effects in a study of the morphological stability of the change of phase interfaces.

Figures 5.16-5.18 will be analyzed next in conjunction with Figs. 5.14 and 5.15, and an explanation will be presented for the result in Fig. 5.13. In the previous study on the planar solidification process it was shown that the velocity of the change of phase interface is proportional to the heat conducted through the solid and removed at the outer boundary. The heat transported is determined by the temperature difference between the outer surface and the interfacial temperature divided by the solid layer thickness. During a planar solidification process, when a temperature perturbation is superimposed on the outer surface temperature, a higher temperature difference will appear between certain points on the change of phase interface and the outer surface. This will increase the solidification rate at these points and consequently will increase the distance between these points on the interface and the outer surface. The increased distance will reduce the temperature gradient and consequently continuously reduce the velocity of the solidification process. The solute accumulation on the change of phase interface discussed with respect to Figs. 5.14 and 5.15 will decrease the change of phase temperature on the moving interface. This temperature will be lower for points on the interface farther away from the outer surface (the tip of the perturbed interface). The increased solute concentration on the interface will also reduce the temperature gradient and consequently will reduce the velocity of the interface.

When the temperature perturbation on the outer surface is removed (i.e., a constant temperature is imposed on the outer surface), the temperature gradient and the consequent heat flux will be lower at points on the change of phase interface furthest away from the outer surface. For the case of pure water, the difference in temperature gradient is affected only by the perturbed interface and the difference in the thickness of the solid layer. For saline solution, the difference is enhanced by the lower temperature on the change of phase interface at points further from the outer surface. Consequently, after removing the temperature perturbation, the velocity of the interface at points closer to the outer surface will increase relative to that at points further from the outer surface until the perturbed interface returns to a stable planar.

The perturbed interface becomes planar faster during the freezing of a solution than during the freezing of pure water. These results are evident in Figs. 5.17 and 5.12. This result indicates that during the analyzed transient solidification process the increased concentration of saline on the interface has a stabilizing effect. The results of this work prove that in the analyzed transient solidification process, a solid-liquid interface surrounded by a thermodynamically supercooled liquid cannot become unstable by means of a transient temperature perturbation on the outer surface. It should be emphasized that this statement is restricted to the transient solidification process analyzed in this work. Since experimental evidence indicates that the solid-liquid interface became unstable during the solidification process, new studies are required to promote the understanding of the physical phenomena associated with the perturbed growth of a solid-liquid interface during transient solidification.

5.4 CONCENTRATION PERTURBATION ON THE INTERFACE

In the previous section it was shown that a temperature perturbation on the outer surface of a solidifying domain cannot induce the instability commonly observed on such an interface during transient solidification. It has been shown that the solute concentration on the interface has a stabilizing effect. Consequently, a study was performed to determine the effects of perturbing the concentration on the interface on the stability of that interface. The study was performed for the same rectangular enclosure discussed in the previous section. The length of the narrow wall was 10 μm and that of the long wall 80 μm. The initial concentration of the saline solution was 34.0 gmol/m^3 and the initial temperature -0.121°C. A constant temperature of -0.3°C was imposed on one of the narrow walls of the enclosure and the other walls were adiabatic. The details of the unperturbed solidification process in this system are shown in Figs. 5.7-5.10 and have been discussed in the previous section.

In this part of the study, a concentration perturbation with a magnitude of 2.0 cos $(\pi r_i/R)$ gmol/m^3 was imposed on the change of phase interface concentration at different instants in time and for various periods of time. This concentration perturbation yielded a continuous decrease in the concentration with a magnitude of 2.0 gmols on the left-hand side of the enclosure relative to the central node in the enclosure, a continuous increase with a magnitude of 2.0 gmols on the right-hand side of the enclosure relative to the central node, and a sinusoidal variation between these two extreme points. Specifically, the concentration on the interface is taken at all times as $C = C_0(r) + 2.0 \cos(\pi r_i/R)$ gmol/m^3, where $C_0(r)$ is the concentration on the interface in the previous time step. Obviously, since the temperature at points with lower concentration is higher, according to the discussion in the previous section these points will experience a higher temperature gradient and move faster. This is confirmed by the results shown in Fig.

5.16. Figure 5.17 shows the location of the solid-liquid interface relative to that of the central node on the interface during a solidification process in which the concentration perturbation is continuously imposed.

The results clearly indicate that the continuous concentration perturbation results in a continuous perturbation of the interface. This behavior has been anticipated as discussed above. It should be emphasized that the results reported in the previous section indicate that a continuous temperature perturbation on the outer surface also yields a continuous perturbation of the interface (see Fig. 5.13). There is, however, a fundamental difference in the behavior of a system perturbed by a concentration perturbation relative to that perturbed by a temperature perturbation. This difference can be observed in Fig. 5.20 and can be explained by Fig. 5.21.

Figure 5.20 shows the velocity of the interface at various instants in time during the perturbation. This figure is especially enlightening when compared with the interface velocity during the temperature perturbation shown in Fig. 5.17. It is seen that for the concentration perturbation the difference in velocity between the tip and the groove of the interface increases in time, while for the concentration perturbation it decreases in time. Thus, while for the temperature perturbation the effects of the temperature perturbation on the interface morphology perturbation will decrease in time, in the concentration perturbation the morphological perturbation on the interface increases in time, eventually leading to the unstable interface observed in experiments.

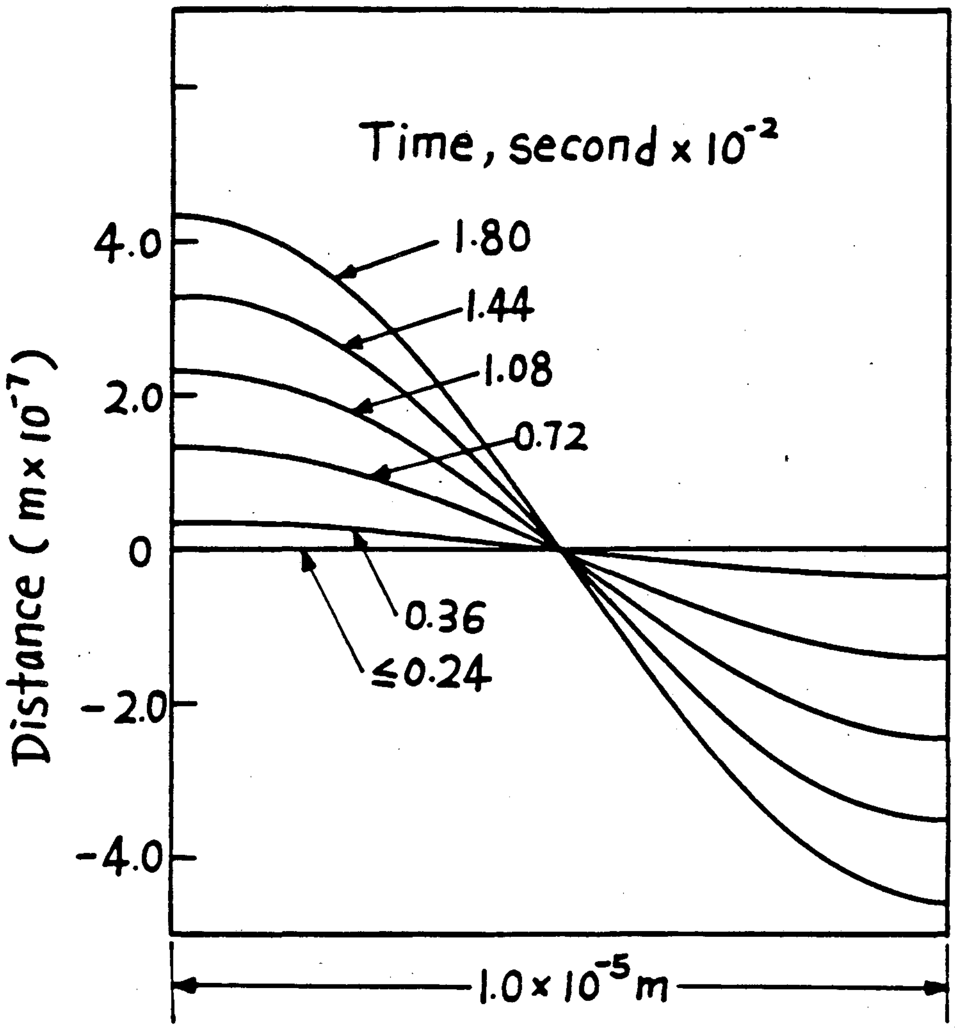


FIGURE 5.19. Interfacial position, concentration perturbation $C = C_0 + 2.0 \cos(\pi r_i/R)$ gmol/m^3, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i^{\,l} = 34.0$ gmol/m^3, $T_\infty = -0.3$°C, $T_i = -0.121$°C.

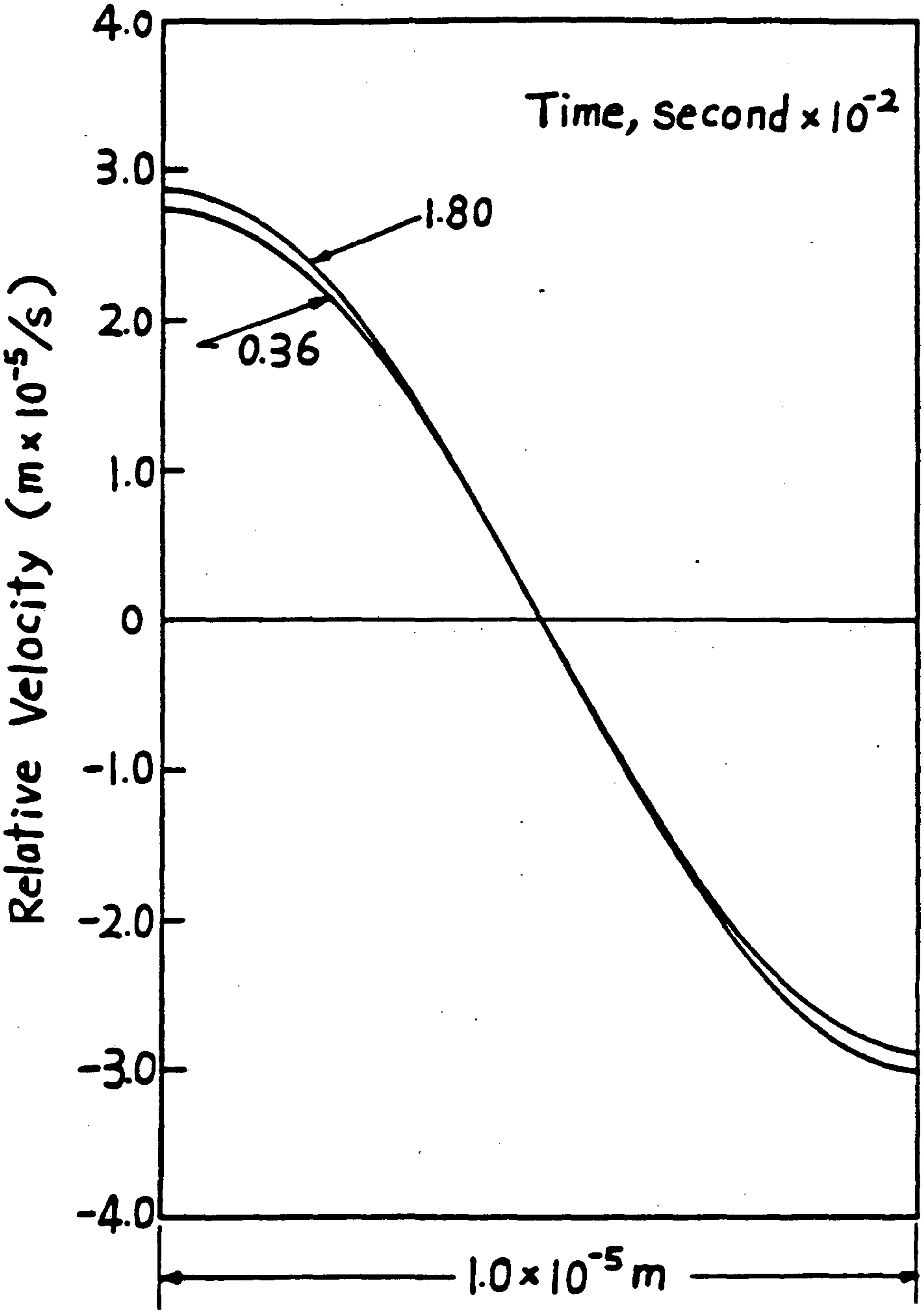


FIGURE 5.20. Interfacial velocity, concentration perturbation. $C = C_0 + 2.0 \cos(\pi r_i/R)$ gmol/m^3, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m^3, $T_\infty = -0.3$°C, $T_i = -0.121$°C.

An explanation for this phenomenon can be obtained through a comparison of Figs. 5.11 and 5.15. These figures show the interface concentration in time in the case of a temperature perturbation and a concentration perturbation, respectively. It is seen in Fig. 5.15 that following a temperature perturbation, the saline concentration on the tip of the perturbed interface increases in time relative to that in the groove. As explained in the previous section, this has a stabilizing effect since it decreases the temperature gradient on the tip relative to that in the groove and reduces the velocity of the tip relative to that in the groove. Figure 5.22 shows that in the case of a concentration perturbation, a completely different phenomenon occurs. The concentration on the tip of the perturbed interface is lower than that in the groove and consequently a higher temperature gradient is expected on the tip than on the groove. This leads to a higher velocity of the tip relative to the groove, which is essentially what happens during an unstable solidification process.

It should be emphasized, however, that the phenomenon in which the tip velocity continuously increases relative to that in the groove was observed only during a continuous concentration perturbation. In numerical experiments in which concentration perturbations were imposed only for one step and then removed, different behavior was observed. Here the interface became perturbed after the single step concentration perturbation. However, since no mechanism was available to remove the solute from the fastest-growing region of the interface (the tip), the concentration on the tip eventually became larger than that in the groove. This result, shown in Fig. 5.23, led to the disappearance of the morphological

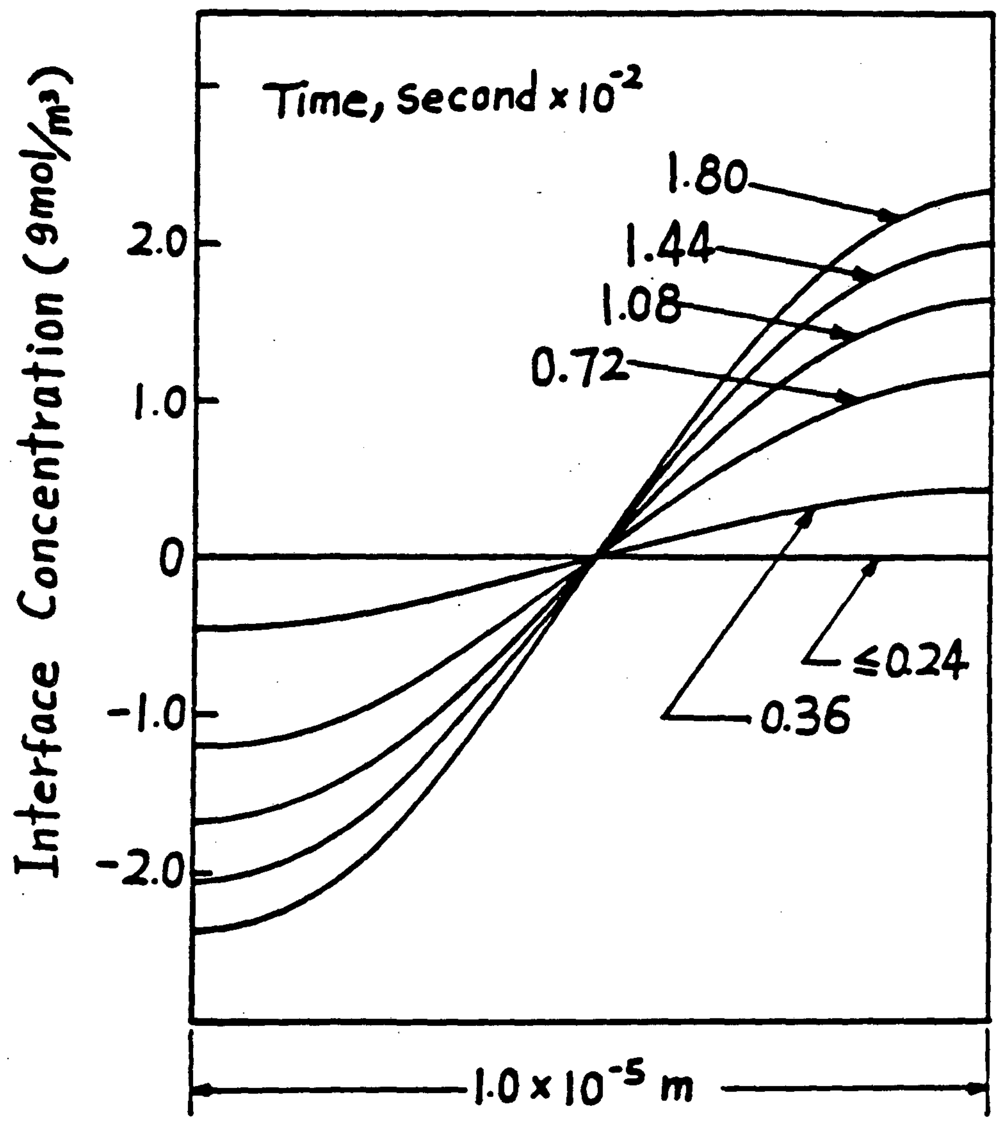


FIGURE 5.21. Interfacial concentration, concentration perturbation. $C = C_0 + 2.0 \cos(\pi r_i/R)$ gmol/m³, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m³, $T_\infty = -0.3$°C, $T_i = -0.121$°C.

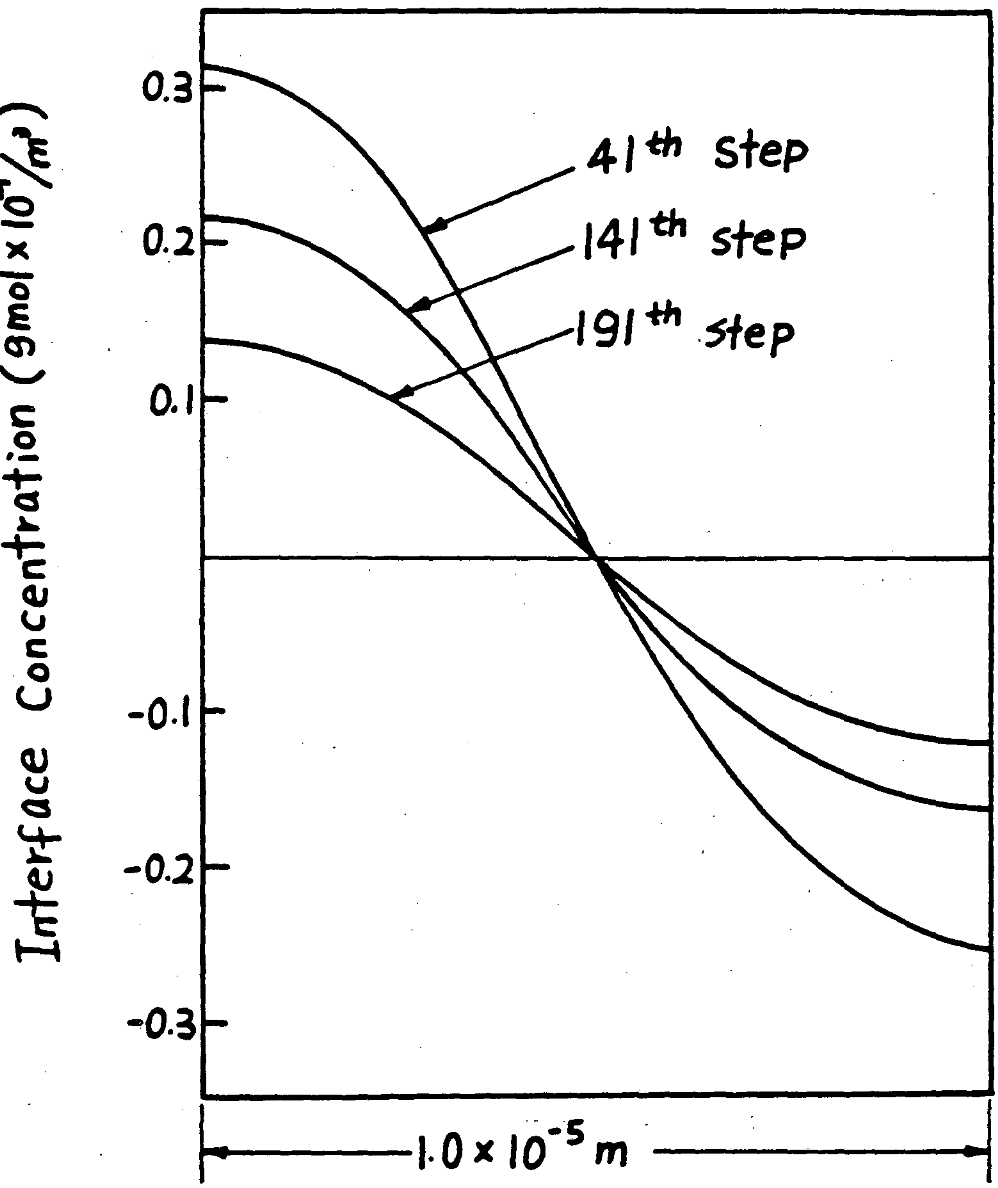


FIGURE 5.22. Interfacial concentration, concentration perturbation, one step only. $C = C_0 + 2.0 \cos(\pi r_i/R)$ gmol/m³, $R = 0.1 \times 10^{-4}$ m, $Z = 0.8 \times 10^{-4}$ m, $C_i = 34.0$ gmol/m³, $T_\infty = -0.3$°C, $T_i = -0.121$°C.

perturbation on the interface in a similar form to that occurring during transient temperature perturbations.

The results of this part of the study indicate that a continuous concentration perturbation on the interface can lead to an unstable interface during a transient solidification process. This is an extremely important observation since it might indicate the fundamental mechanism responsible for the experimentally observed unstable interfaces during solidification in solution.

CHAPTER 6:
CONCLUSIONS

A new numerical method using "front tracking" finite elements has been developed to solve the multi-dimensional transient heat and mass diffusion equations associated with the solidification processes in binary solutions. The numerical method can incorporate realistic thermodynamic conditions on the interface (including surface tension effects) and can accommodate non-isothermal interfaces and irregular transient geometries of the interface. At present this is the only method with these capabilities.

In the front tracking method, the thermal and concentration field equations are solved implicitly while the energy balance equation on the interface is treated as an independent equation being solved explicitly to obtain the new interface position in time. Hence the governing equations of solidification of binary solutions are solved in sequence and marching in time without iteration. The front tracking method developed here is unconditionally stable. Essentially there is no constraint on the size of the time step in terms of numerical stability. Specific to this work is the special procedure by which the change of phase interface is tracked in time.

The interface is tracked in time by two steps. First the magnitude of displacement and normal direction are independently obtained for each node on the interface; then they are superimposed to determine the new interface position. This procedure, which is fundamentally different from that in other front tracking methods, was necessary because of the special conditions on the interface in this problem. A new systematic exponential meshing technique was designed for the concentration field. The novel meshing procedure is essential to obtain an accurate result in this problem.

This method was used to study the physical phenomena occurring during a transient solidification process in binary solutions. The numerical analysis of a transient solidification process in a saline solution has shown that a thin concentration boundary layer will develop in front of the interface during the process. This is caused by the rejection of solute from the solid, which occurs at a faster rate than the diffusion of solute from the interface in the bulk of the solution. It was shown using equilibrium thermodynamics that this phenomenon causes the liquid in front of the interface to be "constitutionally supercooled." According to existing stability criteria, the planar interface must be morphologically unstable.

The new numerical method was employed in a study of the stability of such an interface for different perturbations. The results of the numerical analysis indicate that a transient temperature fluctuation on the outer surface of the solidifying domain cannot generate the instability of the moving interface even in situations in which the solute in front of the interface is thermodynamically supercooled.

Furthermore, it was shown that the solute concentration in front of the interface has a stabilizing effect. These results contradict the theoretical predictions of commonly accepted stability criteria, which are based on equilibrium thermodynamics. The discrepancy is caused by the important effect of the transient heat and mass transfer phenomena on the solidification process, a factor that must not be neglected in the stability analysis.

The numerical method was also employed to study the effect of a concentration perturbation on the interface. It was shown that a continuous concentration perturbation can lead to an unstable interface, and it is tentatively proposed that the instability of the interface must be related with the solute convection process. The results of this work demonstrate the importance of this new method, and show that new fundamental studies are needed to promote the understanding of the physical phenomena associated with the perturbed growth of a solid-liquid interface during transient solidification.

REFERENCES

1. B. Chalmers, Principles of Solidification, Wiley, New York, 1964.

2. M.C. Flemings, Solidification Processing, McGraw-Hill, New York, 1974.

3. K.M. Fisher, "The Effects of Fluid Flow on the Solidification of Industrial Castings and Ingots," Physiocochem. Hydro., 2(4), 311-326, 1981.

4. C.L. Jones and P. Capper, "Thermal Modelling of Casting of Cd, Hg, Te," J. Crystal Growth, 63, 145-153, 1983.

5. B.R. Pamplin, Ed., Crystal Growth, 2nd ed., Pergamon, New York, 1981.

6. F. Rosenberger, Fundamentals of Crystal Growth I, Springer-Verlag, New York, 1979.

7. M.E. Glicksman, R.J. Schaefer, and J.D. Ayers, "Dendritic Growth: A Test of Theory," Metal. Trans. A, 7A, 1747-1759.

8. J.S. Langer, R.F. Sekerka, and T. Fujioka, "Evidence for a Universal Law of Dendritic Growth Rates," J. Crystal Growth, 44, 414-418, 1978.

9. H. Muller-Krumbhaar and J.S. Langer, "Siderbranching Instabilities in a Two-Dimensional Model of Dendritic Solidification," Acta Metal., 29, 145-157, 1981.

10. S.C. Huang and M.E. Glicksman, "Fundamentals of Dendritic Solidification - I. Steady-State Tip Growth," Acta Metal., 29, 701-715, 1981.

11. S.C. Huang and M.E. Glicksman, "Fundamentals of Dendritic Solidification - II. Development of Siderbranch Structure," Acta Metal., 29, 717-734, 1981.

12. D.J. Fisher and W. Kurz, "A Theory of Branching Limited Growth of Irregular Eutectics," Acta Metal., 28, 777-794, 1980.

13. W. Kurz and D.J. Fisher, "Dendrite Growth at the Limit of Stability: Tip Radius and Spacing," Acta Metal., 29, 11-20, 1981.

14. M. Solari and H. Bilonia, "Microsegregation in Cellular and Cellular Dendritic Growth," J. Crystal Growth, 49, 451-457, 1980.

15. M.H. Burden and J.D. Hunt, "Cellular and Dendritic Growth - I," J. Crystal Growth, 22, 99-108, 1974.

16. M.H. Burden and J.D. Hunt, "Cellular and Dendritic Growth - II," J. Crystal Growth, 22, 109-116, 1974.

17. S. Witzke, J.P. Riquet, and F. Durand, "Diffusion Field Ahead of a Growing Columnar Front: Discussion of the Columnar-Equiaxed Transition," Acta Metal., 29, 365374, 1981.

18. B.W. Grange, R. Viskanta, and W.H. Stevenson, "Diffusion of Heat and Solute During Freezing of Salt Solutions," Int. J. Heat & Mass Transf., 19, 373-384, 1976.

19. R.L. Levin, "The Freezing of Finite Domain Aqueous Solutions: Solute Redistribtuion," Int. J. Heat & Mass Transf., 24(9), 1443-1455, 1981.

20. J.P. Terwilliger and S.F. Dizio, "Salt Rejection Phenomena in the Freezing of Saline Solution," Chem. Eng. Sci., 25, 1331-1349, 1970.

21. C. Korber,M.W. Scheiwe, and K. Wollhoever, "Solute Polarization During Planar Freezing of Aqueous Salt Solutions," Int. J. Heat & Mass Transf., 26(8), 1241-1253, 1983.

22. K. Shibrta, J. Sato, and G. Ohira, "The Solute Distributions in Dilute Al-Ti Alloys During Unidirectional Solidification," J. Crystal Growth, 44, 435-445, 1978.

23. T.W. Clyne, "Heat Flow in Controlled Directional Solidification of Metals - II. Mathematical Model," J. Crystal Growth, 50, 691-700, 1980.

24. M.G. O'Callaghan,E.G. Cravalho, and C.E. Huggins, "An Analysis of the Heat and Solute Transport During Solidification of an Aqueous Binary Solution - I. Basal Plane Region," Int. J. Heat & Mass Transf., 25(4), 553-561, 1982.

25. M.G. O'Callaghan, E.G. Cravalho, and C.E. Huggins, "An Analysis of the Heat and Solute Transport During Solidification of an Aqueous Binary Solution - II. Dendrite Tip Region," Int. J. Heat & Mass Transf., 25(4), 563-573, 1982.

26. B.W. Grange, R. Viskanta, and W.H. Stevenson, "Solute and Thermal Redistribution During Freezing of Salt Solutions," ASME, Cu3.3.

27. R. Siegel, "Analysis of Solidification Interface Shape Resulting from Applied Sinusoidal Heating," J. Heat Transf., 104, 13-18, 1982.

28. S.R. Coriell and R.F. Sekerka, "Lateral Solute Segregation During Unidirectional Solidification of Binary Alloy with a Curved Solid-Liquid Interface," J. Crystal Growth, 46, 479-482, 1979.

29. S.R. Coriell, R.F. Boisvert, R.G. Rehm, and R.F. Sekerka, "Lateral Solute Segregation During Unidirectional Solidification of a Binary Alloy with a Curved Solid-Liquid Interface - II. Large Departures from Planarity," J. Crystal Growth, 54, 167-175, 1981.

30. T.W. Clyne, "Heat Flow in Controlled Directional Solidification of Metals, I. Experimental Investigation," J. Crystal Growth, 50, 684-690, 1980.

31. C. Korber and M.W. Scheiwe, "Observations on the Nonplanar Freezing of Aqueous Salt Solutions," J. Crystal Growth, 61, 307-316, 1983.

32. I. Jin and G.R. Purdy, "Controlled Solidification of a Dilute Binary Alloy - II. Experiment," J. Crystal Growth, 23, 37-44, 1974.

33. I. Jin and G.R. Purdy, "Controlled Solidification of a Dilute Binary Alloy - I. Theory," J. Crystal Growth, 23, 29-36, 1974.

34. R. Trivedi, "Theory of Dendritic Growth During the Directional Solidification of Binary Alloys," J. Crystal Growth, 49, 219-232, 1980.

35. M.H. Johnston, C.S. Griner, R.A. Parr, and S.J. Robertson, "The Direct Observation of Unidirectional Solidification as a Function of Gravity Level," J. Crystal Growth, 50, 831-838, 1980.

36. S.R. Coriell, M.R. Cordes, W.J. Boettinger, and R.F. Sekerka, "Convective and Interfacial Instabilities During Unidirectional Solidification of a Binary Alloy," J. Crystal Growth, 49, 13-28, 1980.

37. W.A. Tiller, K.A. Jackson, J.W. Rutter, and B. Chalmers, Acta Metal., 1, 428, 1953.

38. W.W. Mullins and R.F. Sekerka, "Stability of a Planar Interface During Solidification of a Dilute Binary Alloy," J. Appl. Phys., 35(2), 444-451, 1964.

39. W.W. Mullins and R.F. Sekerka, "Morphological Stability of a Particle Growing by Diffusion or Heat Flow," J. Appl. Phys., 34(2), 323-329, 1963.

40. R.F. Sekerka, "A Stability Function for Explicit Evaluation of the Mullin-Sekerka Interface Stability Criterion," J. Appl. Phys., 36(1), 264-268, 1965.

41. R. F. Sekerka, "Morphological Stability," J. Crystal Growth, 3-4, 71-81, 1968.

42. G. Horvay and J.W. Cahn, "Dendritic and Spheroidal Growth," Acta Metal., 9, 695-705, 1961.

43. D.P. Woodruff, The Solid-Liquid Interface, Cambridge University Press, 1973.

44. R.T. Delves, "Theory of Interface Stability," in Crystal Growth, B.R. Pamplin, Ed., 2nd ed., Pergamon, New York, 1981.

45. S.R. Coriell and R.F. Sekerka, "Oscillatory Morphological Instability due to Non-Equilibrium Segregation," J. Crystal Growth, 61, 499-508, 1983.

46. S.R. Coriell and R.F. Sekerka, "Effect of Convective Flow on Morphological Stability," Physicochem. Hydro., 2(4), 281-293, 1981.

47. J.S. Langer and H. Muller-Krumbhaar, "Theory of Dendritic Growth - I. Element of a Stability Analysis," Acta Metal., 26, 1681-1687, 1978.

48. J.S. Langer and H. Muller-Krumbhaar, "Theory of Dendritic Growth - II. Instability in the Limit of Vanishing Surface Tension," Acta Metal., 26, 1689-1695, 1978.

49. H. Muller-Krumbhaar and J.S. Langer, "Theory of Dendritic Growth - III. Effects of Surface Tension," Acta Metal., 26, 1697-1708, 1978.

50. N. Ramachandran, J.P. Gupta, and Y. Jaluria, "Two-Dimensional Solidification with Natural Convection in the Melt and Convective and Radiative Boundary Conditions," Num. Heat Transf., 4, 469-484, 1981.

51. N. Ramachandran, J.P. Gupta, and Y. Jaluria, "Thermal and Fluid Flow Effects During Solidification in a Rectangular Enclosure," Int. J. Heat & Mass Transf., 25(2), 187-194, 1982.

52. T.K. Sinha and J.P. Gupta, "Solidification in an Annulus," Int. J. Heat & Mass Transf., 25(11), 1771-1773, 1982.

53. N. Shamsundar and E.M. Sparrow, "Effect of Density Change on Multi-Dimensional Conduction Phase Change," J. Heat Transf., 551-557, Nov. 1976.

54. B. Cantor and A. Vogel, "Dendritic Solidification and Fluid Flow," J. Crystal Growth, 41, 109-123, 1977.

55. D.T.J. Hurle, E. Jakeman, and A.A. Wheeler, "Effect of Solutal Convection on the Morphological Stability of a Binary Alloy," J. Crystal Growth, 58, 163-179, 1982.

56. M.A. Azouni, "Local Temperature Measurements Over Ice Water Interface and Convective Flows Patterns," J. Crystal Growth, 47, 109-114, 1979.

57. J.H. Quenisset and R. Naslain, "Effect of Forced Convection on Eutectic Growth," J. Crystal Growth, 54, 465-474, 1981.

58. M.E. Glicksman, "Capillary Phenomena During Solidification," J. Crystal Growth, 42, 347-356, 1977.

59. R. Trivedi, H. Franke, and R. Lacmann, "Effects of Interface Kinetics on the Growth Rate of Dendrites," J. Crystal Growth, 47, 389-396, 1979.

60. D.T.J. Hurle, "The Effect of Soret Diffusion on the Morphological Stability of a Binary Alloy Crystal," J. Crystal Growth, 61, 463-472, 1983.

61. B. Cantor and A. Vogel, "Dendritic Solidification and Fluid Flow," J. Crystal Growth, 41, 109-123, 1977.

62. C. Gau and R. Viskanta, "Flow Visualization During Solid-Liquid Phase Change Heat Transfer, I. Freezing in a Rectangular Cavity," Int. Comm. Heat Mass Transf., 10, 173-181, 1983.

63. C. Gau, R. Viskanta, and C.J. Ho, "Flow Visualization During Solid-Liquid Phase Change Heat Transfer, II. Melting in a Rectangular Cavity," Int. Comm. Heat Mass Transf., 10, 183-190, 1983.

64. R.J. Schaefer, "The Validity of Steady-State Dendrite Growth Models," J. Crystal Growth, 43, 17-20, 1978.

65. R. Trivedi, "Comments on the Validity of Steady-State Dendrite Growth Models by R.J. Schaefer," J. Crystal Growth, 44, 110-111, 1978.

66. D.G. Wilson, A.D. Soloman, and P.T. Boggs, Eds., Moving Boundary Problems, Academic, New York, 1978.

67. J.R. Ockenden and W.R. Hodgkins, Eds, Moving Boundary Problems in Heat Flow and Diffusion, Clarendon Press, Oxford, 1977.

68. V.J. Lunandini, Heat Transfer in Cold Climates, Van Nostrand Reinhold, New York, 1981.

69. M.N. Ozisik, Heat Conduction, Wiley, New York, 1980.

70. B. Rubinsky and G. Cravalho, Trans. ASME, J. Heat Transf., 326-330, May 1979.

71. T.R. Goodman, Trans. ASME, 80, 335-342, 1956.

72. J. Stefan, Ann. Phy. Chem. Newefalge, 42(2), 269-286, 1881.

73. N. Shamsundar and E.M. Sparrow, "Analysis of Multidimensional Conduction Phase Change Via the Enthalpy Model," J. Heat & Mass Transf., 333-340, Aug. 1975.

74. N. Shamsundar, "Approximation Calculation of Multidimensional Solidification by Using Conduction Shape Factors," J. Heat Transf., 104, 8-12, Feb. 1982.

75. J.L. Sproston, "Two Dimensional Solidification in Pipes of Rectangular Section" Int. J. Heat & Mass Transf., 24(9), 1493-1501, 1981.

76. V.R. Voller and M. Cross, "Estimating the Solidification/Melting Times of Cylindrically Symmetric Regions," Int. J. Heat & Mass Transf., 24(9), 1457-1462, 1981.

77. V. Voller and M. Cross, "Accurate Solutions of Moving Boundary Problems Using the Enthalpy Method," Int. J. Heat & Mass Transf., 24, 545-556, 1981.

78. R. Bonerot and P. Jamet, "Numerical Computation of the Free Boundary for the Two-Dimensional Stefan Problem by Space-Time Finite Elements," J. Comp. Phys., 25, 163-181, 1977.

79. H.M. Ettouney and R.A. Brown, "Finite Element Methods for Steady Solidification Problems," J. Comp. Phys., 49, 118-150, 1983.

80. D.R. Lynch and K. Q'nell, "Continuously Deforming Finite Elements for the Solution of Parabolic Problems, With and Without Phase Change," Int. J. Num. Meth. in Eng., 17, 81-96, 1981.

81. K. Miller and R.M. Miller, "Moving Finite Elements I," SIAM J. Num. Anal., 18(6), 1019-1032, 1981.

82. K. Miller, "Moving Finite Elements II," SIAM J. Num. Anal., 18(6), 1033-1057, 1981.

83. B. Rubinsky and E.G. Cravalho, "A Finite Elemenet Method for the Solution of One-Dimensional Phase Change Problems," Int. J. Heat & Mass Transf., 24(12), 1987-1989, 1981.

84. B. Rubinsky, "Solidification Processes in Saline Solutions," J. Crystal Growth, 62, 513-522, 1983.

85. J. Yoo and B. Rubinsky, Num. Heat Transf., 6, 205-222, 1983.

86. G. Comini and S. del Guidice, Int. J. Num. Math. Eng., 8, 613-624, 1979.

87. J. Ronel and B.R. Baliga, ASME Paper No. 79-wa/HT-54.

88. E.B. Becker, G.F. Carey, and J.T. Oden, Finite Elements: An Introduction, Vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1981.

89. G.F. Carey and J.T. Oden, Finite Elements: A Second Course, Vol. 2, Prentice-Hall, Englewood Cliffs, NJ, 1983.

90. K.J. Bathe, Finite Element Procedures in Engineering Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1982.

91. K.H. Huebner and E.M. Thornton, The Finite Element Method for Engineers, 2nd ed., Wiley-Interscience, New York, 1982.

92. O.C. Zienkiewicz, The Finite Element Method, 3rd ed., McGraw-Hill, New York, 1977.

93. O.C. Zienkiewicz and K. Morgan, Finite Elements and Approximation, Wiley-Interscience, New York, 1982.

94. J.E. Akin, Application and Implementation of Finite Element Methods, Academic, New York, 1982.

95. T.J.R. Highes, "Finite Element Methods for Convection Dominated Flow," ASME AMO, Vol. 34, 1979.

96. A.K. Noor and W.D. Pilkey, Eds., State of the Art Surveys on Finite Element Technology, ASME, New York, 1983.

97. P.J. Davis, Introduction and Approximation, Dover, 1975.

98. K.E. Atkinson, An Introduction to Numerical Analysis, Wiley, New York, 1978.

99. G. Dahlquist and A. Bjorck, Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, 1974.

100. S.D. Conte and C. Boor, Elementary Numerical Analysis, McGraw-Hill, New York, 1980.

APPENDIX 1:

DERIVATION OF MULLINS-SEKERKA CRITERION, EQUATION (1.17)

A sinusoidal perturbation of very small amplitude δ applying to the planar interface in the constant moving coordinate can be described by

$$z = \phi(x,t) = \delta(t) \sin \omega x \quad , \tag{A.1}$$

where $\omega = 2\pi/\lambda$ is the wave frequency. Notice that a two-dimensional model is used.

The governing equations of solute diffusion in the liquid and heat diffusion of both the solid and liquid in a constant moving frame of reference are:

$$D\left(\frac{\partial^2 C}{\partial z^2} + \frac{\partial^2 C}{\partial x^2}\right) + V\frac{\partial C}{\partial z} = 0 \tag{A.2}$$

$$\alpha_S\left(\frac{\partial^2 T_S}{\partial z^2} + \frac{\partial^2 T_S}{\partial x^2}\right) + V\frac{\partial T_S}{\partial z} = 0 \tag{A.3}$$

$$\alpha_L\left(\frac{\partial^2 T_L}{\partial z^2} + \frac{\partial^2 T_L}{\partial x^2}\right) + V\frac{\partial T_L}{\partial z} = 0 \quad . \tag{A.4}$$

Equation (A.2) will be solved first.

The solution to (A.2) can be written as

$$C(z,x) = C_L(z) + C_1(z) \sin \omega x \quad , \tag{A.5}$$

where C is the solute perturbation of order δ and $C_L(z)$ is the basic solution of unperturbed situation derived in Eq. (1.5). For convenience, Eq. (1.5) is rewritten here:

$$C_L(z) = C_0 + \frac{DG_C}{V}\ [1 - \exp(-Vz/D)]\ . \qquad (1.5)$$

It is noted that $z = \phi(x,t)$ is the coordinate of the perturbed interface, and not $z = 0$. Substituting Eq. (A.5) into (A.2), $C(z)$ should satisfy

$$D\left(\frac{\partial^2 C_1}{\partial z^2} - \omega^2 C_1\right) + V\frac{\partial C_1}{\partial z} = 0\ . \qquad (A.6)$$

At the perturbed interface the assumptions of linear perturbation on temperature and concentration yield

$$T_\phi = T_0 + a\phi(x,t) = T_0 + a\delta(t)\sin\omega x \qquad (A.7)$$

$$C_\phi = C_0 + b\phi(x,t) = T_0 + b\delta(t)\sin\omega x\ , \qquad (A.8)$$

where T_0 and C_0 are the temperature and concentration at the unperturbed planar interface, respectively, the same as those obtained in Section 1.2.1. The second terms in Eqs. (A.7) and (A.8) are the first-order corrections corresponding to the infinitesimal perturbation. Subscript ϕ of T_ϕ and C_ϕ is used to emphasize that these values are derived on the perturbed interface. a and b are constants to be determined. The boundary conditions for Eq. (A.6) are

$$C_1 = \delta b - \frac{\delta DG_C}{V\phi}[1 - \exp(-V\phi/D)] \quad \text{at} \quad z = \phi \qquad (A.9)$$

$$C_1 \to 0 \quad \text{as} \quad z \to \infty\ . \qquad (A.10)$$

The solution of Eq. (A.6) under the above two boundary conditions is

$$C_1 = \delta(b - G_C)\ \exp\ (-\omega_C z) \quad , \tag{A.11}$$

where

$$G_C = -\frac{V}{D}(C_L - C_S) \tag{A.12}$$

$$\omega_C = \frac{V}{2D} + \left[\left(\frac{V}{2D}\right)^2 + \omega^2\right]^{\frac{1}{2}} \quad . \tag{A.13}$$

The complete solution of Eq. (A.2) is obtained by substituting Eqs. (1.5) and (A.9) into (A.5) to yield

$$C(z,x) = \left\{C_0 + \frac{DG_C}{V}\left[1 - \exp\left(\frac{-VZ}{D}\right)\right]\right\} + \delta(b-G_C)\ \exp\ (-\omega_C z)\ \sin\ \omega x \ . \tag{A.14}$$

Following exactly the same steps above, the solutions of Eqs. (A.3) and (A.4) are

$$T_L(z,x) = \left\{T_0 + \frac{\alpha_L G_L}{V}[(1 - \exp\ (-Vz/\alpha_L)]\right\}$$

$$+ \ [\delta(a - G_L)\ \exp\ (-\omega_L z)\ \sin\ \omega x] \tag{A.15}$$

$$T_S(z,x) = \left\{T_0 + \frac{\alpha_S G_S}{V}[1 - \exp\ (-Vz/\alpha_S)]\right\}$$

$$+ \ [\delta(a - G_S)\ \exp\ (-\omega_S z)\ \sin\ \omega x] \quad , \tag{A.16}$$

where

$$\omega_L = \frac{V}{2\alpha_L} + \left[\left(\frac{V}{2\alpha_L}\right)^2 + \omega^2\right]^{\frac{1}{2}} \tag{A.17}$$

$$\omega_S = \frac{V}{2\alpha_S} - \left[\left(\frac{V}{2\alpha_S}\right)^2 + \omega^2\right]^{\frac{1}{2}} \quad . \tag{A.18}$$

It is seen that the first part of the solutions in (A.11) and (A.12) are the unperturbed solutions. If the capillarity is considered on the perturbed interface, then

$$T_{L\phi} = T_{S\phi} = T_\phi = T_m + mC_\phi - T_m\Gamma R \quad , \tag{A.19}$$

where $\Gamma = \gamma/L$, the surface free energy divided by the latent heat per unit volume, T_m is the melting point of pure liquid at the planar situation, m is the slope of liquid line, and $R = (1/r_1)+(1/r_2)$ is the curvature of the interface.

Since the perturbed interface is a sinusoidal function, r = in the y-direction and R is obtained only from the $1/r_2$ by the following formula:

$$R = -\frac{\partial^2\phi}{\partial x^2}\left[1 + \left(\frac{\partial\phi}{\partial x}\right)^2\right]^{-3/2} . \tag{A.20}$$

Since $\partial\phi/\partial x \sim \delta$, which is negligible compared with 1, and $\phi = \delta \sin \omega x$, then R is obtained as

$$R = \delta\omega^2 \sin \omega x \quad . \tag{A.21}$$

Hence

$$T_\phi = T_m + mC_\phi - T_m\Gamma\delta\omega^2 \sin \omega x \quad . \tag{A.22}$$

Substituting Eqs. (A.7) and (A.8) into (A.18), the relationship between a and b is obtained by

$$a = mb - T_m\Gamma\omega^2 \quad . \tag{A.23}$$

The energy and solute balance at the perturbed interface requires

$$\left(k_S \frac{\partial T_S}{\partial z} - k_L \frac{\partial T_L}{\partial z}\right)_\phi = Lv(x) \tag{A.24}$$

$$\left[D(k-1) \frac{\partial C}{\partial z} \right]_\phi = C_\phi v(x) \tag{A.25}$$

$$\frac{1}{L}\left(k_S \frac{\partial T_S}{\partial z} - k_L \frac{\partial T_C}{\partial z} \right)_\phi = \frac{1}{C_\phi}\left[D(k-1) \frac{\partial C}{\partial z} \right]_\phi . \tag{A.26}$$

The gradients of concentration in the liquid and of temperatures in the liquid and solid at the interface to the first order of δ are:

$$\left(\frac{\partial C}{\partial z}\right)_\phi = -\omega_C \left[b - G_C\left(1 - \frac{V}{\omega_C D}\right)\right] \delta \sin \omega x + G_C \tag{A.27}$$

$$\left(\frac{\partial T_L}{\partial z}\right)_\phi = -\omega_L \left[a - G_L\left(1 - \frac{V}{\omega_L \alpha_L}\right)\right] \delta \sin \omega x + G_L$$

$$\approx -\omega(a - G_L)\delta \sin \omega x + G_L \tag{A.28}$$

$$\left(\frac{\partial T_S}{\partial z}\right)_\phi = \omega_S \left[a - G_S\left(1 - \frac{V}{\omega_S \alpha_S}\right)\right] \delta \sin \omega x + G_S$$

$$\approx \omega(a - G_S)\delta \sin \omega x + G_S \quad , \tag{A.29}$$

where the approximations $\alpha_L \omega_L >> V$ and $\alpha_S \omega_S >> V$ have been made. It is noted that in all cases of practical interest, $V < 3.0 \times 10^{-5}$ m/sec, $\alpha_L > 10^6$ m^2/sec, $\omega >> V/\alpha_L \sim 3.0 \times 10^{-1}$/m, and $\lambda = 2\pi/\omega << \alpha_L/V \sim 10^{-1}$ cm (typical values for a metallic system). Also, by Eqs. (A.13) and (A.14), $\omega_S \sim \omega_L \sim \omega$ can be obtained.

Substituting these gradients into Eq. (A.22), and by Eq. (A.19), the constant b to the first order of δ is obtained by

$$b = \frac{2G_C T_m \Gamma \omega^2 + \omega G_C(\xi_S + \xi_L) + G_C\left(\omega_C - \frac{V}{D}\right)(\xi_S - \xi_L)}{2\omega m G_C + (\xi_S - \xi_L)[\omega_C - (V/D)P]} , \tag{A.30}$$

where

$$\xi_L = \frac{k_L}{\bar{k}} G_L \quad , \qquad \xi_S = \frac{k_S}{\bar{k}} G_S$$

$$\bar{k} = \tfrac{1}{2}(k_S + k_L) \ , \quad P = (1 - k) \quad . \tag{A.31}$$

Notice that

$$v(x) = V + \frac{d\delta(t)}{dt} \sin \omega x \quad . \tag{A.32}$$

Substitute Eq. (A.23) into (A.21) to obtain

$$v(x) = \frac{\bar{k}}{L} \left\{ (\xi_S - \xi_L) + \omega[2a - (\xi_S + \xi_L)\delta \sin \omega x] \right\} \ . \tag{A.33}$$

Equating the coefficients of Eqs. (A.27) and (A.28) yields

$$V = \frac{\bar{k}}{L} (\xi_S - \xi_L) \tag{A.34}$$

$$\dot{\delta} = \frac{d\delta}{dt} = \frac{2\bar{k}}{L} \omega[a - \tfrac{1}{2}(\xi_S + \xi_L)]\delta \quad . \tag{A.35}$$

The value of a can be obtained by substituting b from (A.26) into (A.29). Then (A.30) becomes the central result,

$$\frac{\dot{\delta}}{\delta} = \frac{V\left[-2T_m\Gamma\omega^2\left(\omega_C - \frac{V}{D} P\right) - (\xi_S+\xi_L)\left(\omega_C - \frac{V}{D} P\right) + 2mG_C\left(\omega_C - \frac{V}{D}\right)\right]}{(\xi_S - \xi_L)\left(\omega_L - \frac{V}{D} P\right) + 2\omega mG_C} \ . \tag{A.36}$$

APPENDIX 2:

LISTING OF COMPUTER PROGRAMS

```
      program aatsai(input,output,tape5=input,tape6=output)
      level 2, gk,gf,nz
c....Common block for general purpose
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/celem/ ne(1000),node(9,1000)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c....Common block for region 1 and 2, both are temp. field
      common/cnode/ x12(2,1200),u12(1200)
c....Common block for region 1, Temp. of solid
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
     .             ,npt1(10),vpt1(10)
      common/cscal1/ rlen1,rval1
c....Common block for region 2, Temp. of liquid solution
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .             ,npt2(10),vpt2(10)
      common/cscal2/ rlen2,rval2
c....Common block for region 3, concentration of liquid solution
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cnode3/ x3(2,1200),u3(1200)
      common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
     .             ,npt3(10),vpt3(10)
c....Common block for interface, region 4
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .              ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/celem4/ ne4(25),nodes4(3,25)
c
      call data
      call comn
      call init
      icount=0
100   icount=icount+1
      time=icount*delt
      call move
      call sov3
      call gett
      call sov1
      call sov2
      if(time.lt.tmax) go to 100
      stop
      end
      subroutine adst4(xim)
c
c....To adjust the interface nodal coordinates
c....To compute the interface normal directions
c....To compute the principal curvatures along the interface
c....To construct the finer interface mesh for computing concentration
c....To use 5-node interpolation polynomial
```

```
c
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cdoman/ xdim,ydim,prop(4,4)
      dimension  xim(2,25),xx(5),xy(5)
c....Deal with the end nodes
c....Symmetry on both sides
      const=1.57079632679489
      angl(1)=dasin(1.d0)
      angl(max124)=angl(1)
      ratio(1)=(ydim-xim(2,1))/(ydim-xi3(2,1))
      ratio(nmax3)=(ydim-xim(2,max124))/(ydim-xi3(2,nmax3))
      xi3(2,1)=xim(2,1)
      xi124(2,1)=xim(2,1)
      xi3(2,nmax3)=xim(2,max124)
      xi124(2,max124)=xim(2,max124)
      capa(1,1)=0.0
      capa(1,max124)=0.0
      capa(2,1)=0.0
      capa(2,max124)=0.0
c....Use 3-node interpolation poly. at the side nodes
      m=2
      do 10 i=1,3
        xx(i)=xim(1,i)
        xy(i)=xim(2,i)
10    continue
c....Get divided differences
      call divdi4(xx,3,xy)
c....Get finer mesh for concentration
      call int4(xx,3,xy,xi3(1,2),p)
      ratio(m)=(ydim-p)/(ydim-xi3(2,m))
      xi3(2,m)=p
c....Get common node for temp. and concen.
      call inter4(xx,3,xy,xi124(1,2),p,thita,curv)
      ms=2*m-1
      ratio(ms)=(ydim-p)/(ydim-xi3(2,ms))
      xi3(2,ms)=p
      xi124(2,m)=p
      angl(m)=thita
      capa(1,m)=curv
      if(xi124(1,m).eq.0.0) then
        capa(2,m)=capa(1,m)
        else
          if(thita.gt.const) then
            capa(2,m)=0.0
          else
            capa(2,m)=cos(thita)/xi124(1,m)
          end if
      end if
      do 50 m=3,max124-2
        mi=m-2
        do 40 i=1,5
          xx(i)=xim(1,mi)
          xy(i)=xim(2,mi)
```

```
          mi=mi+1
40      continue
        call divdi4(xx,5,xy)
        mm=2*(m-1)
        call int4(xx,5,xy,xi3(1,mm),p)
        ratio(mm)=(ydim-p)/(ydim-xi3(2,mm))
        xi3(2,mm)=p
        call inter4(xx,5,xy,xi124(1,m),p,thita,curv)
        ms=2*m-1
        ratio(ms)=(ydim-p)/(ydim-xi3(2,ms))
        xi124(2,m)=p
        xi3(2,ms)=p
        angl(m)=thita
        capa(1,m)=curv
        if(thita.gt.const) then
          capa(2,m)=0.0
        else
          capa(2,m)=cos(thita)/xi124(1,m)
        end if
50    continue
      m=max124-2
      do 60 i=1,3
        xx(i)=xim(1,m)
        xy(i)=xim(2,m)
        m=m+1
60    continue
      mm=nmax3-3
      call divdi4(xx,3,xy)
      do 70 i=mm,mm+2
        call int4(xx,3,xy,xi3(1,i),p)
        ratio(i)=(ydim-p)/(ydim-xi3(2,i))
        xi3(2,i)=p
70    continue
      np=max124-1
      call inter4(xx,3,xy,xi124(1,np),p,thita,curv)
      xi124(2,np)=p
      angl(np)=thita
      capa(1,np)=curv
      if(thita.gt.const) then
        capa(2,np)=0.0
      else
        capa(2,np)=cos(thita)/xi124(1,np)
      end if
      return
      end
      subroutine aply1
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
     .             ,npt1(10),vpt1(10)
      common/celem/ ne(1000),node(9,1000)
```

```
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      dimension pe(9,9),game(9),xx(2,9),nod(9),uu(9)
c
c....Apply point loads
      if(npot1.eq.0) go to 20
      do 10 i=1,npot1
        n=npt1(i)
10      gf(n)=gf(n)+vpt1(i)
c....Apply essential boundary conditions
20    if(nbc11.eq.0) go to 40
        big=1.0e100
      do 30 i=1,nbc11
        nn=ndbc11(i)
        gf(nn)=big*vbc11(i)
        gk(nn,1)=big
30    continue
c....Apply natural boundary conditions
40    if(nbc12.eq.0) go to 70
      do 60 itt=1,nbc12
        nel=neb12(itt)
        ns=nsde12(itt)
        nee=ne(nel)
c....Pick out nodal coordinates
      do 50 j=1,nee
        nod(j)=node(j,nel)
        nj=nod(j)
        xx(1,j)=x12(1,nj)
        xx(2,j)=x12(2,nj)
        uu(j)=u12(nj)
50    continue
      call bcint(vbc12(1,itt),vbc12(2,itt),xx,pe,game,nee,ns,uu)
      call assmb(pe,game,nee,nod)
60    continue
70    return
      end
      subroutine aply2
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .             ,npt2(10),vpt2(10)
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      dimension pe(9,9),game(9),xx(2,9),nod(9),uu(9)
c
c....Apply point loads
      if(npot2.eq.0) go to 20
      do 10 i=1,npot2
        n=npt2(i)
```

```
10        gf(n)=gf(n)+vpt2(i)
c....Apply essential boundary conditions
20      if(nbc21.eq.0) go to 40
        big=1.0e100
        do 30 i=1,nbc21
          nn=ndbc21(i)
          gf(nn)=big*vbc21(i)
          gk(nn,1)=big
30      continue
c....Apply natural boundary conditions
40      if(nbc22.eq.0) go to 70
        do 60 itt=1,nbc22
          nel=neb22(itt)+nelem1
          ns=nsde22(itt)
          nee=ne(nel)
c....Pick out nodal coordinates
        do 50 j=1,nee
          nod(j)=node(j,nel)
          nj=nod(j)+nnode1
          xx(1,j)=x12(1,nj)
          xx(2,j)=x12(2,nj)
          uu(j)=u12(nj)
50      continue
        call bcint(vbc22(1,itt),vbc22(2,itt),xx,pe,game,nee,ns,uu)
        call assmb(pe,game,nee,nod)
60      continue
70      return
        end
        subroutine aply3
        level 2, gk,gf,nz
        common/cntrl1/ nopt1,nopt2,nopt3
        common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
        common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
        common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
        common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
     .               ,npt3(10),vpt3(10)
        common/celem/ ne(1000),node(9,1000)
        common/cnode3/ x3(2,1200),u3(1200)
        common/cmatrx/ gk(1200,25),gf(1200),nz
        dimension pe(9,9),game(9),xx(2,9),nod(9),uu(9)
c
c....Apply point loads
        if(npot3.eq.0) go to 20
        do 10 i=1,npot3
          n=npt3(i)
10        gf(n)=gf(n)+vpt3(i)
c....Apply essential boundary conditions
20      if(nbc31.eq.0) go to 40
        big=1.0e100
        do 30 i=1,nbc31
          nn=ndbc31(i)
          gf(nn)=big*vbc31(i)
          gk(nn,1)=big
30      continue
```

```
c....Apply natural boundary conditions
40    if(nbc32.eq.0) go to 70
      do 60 itt=1,nbc32
        nel=neb32(itt)
        ns=nsde32(itt)
        nee=ne(nel)
c....Pick out nodal coordinates
      do 50 j=1,nee
        nod(j)=node(j,nel)
        nj=nod(j)
        xx(1,j)=x3(1,nj)
        xx(2,j)=x3(2,nj)
        uu(j)=u3(nj)
50    continue
      call bcint(vbc32(1,itt),vbc32(2,itt),xx,pe,game,nee,ns,uu)
      call assmb(pe,game,nee,nod)
60    continue
70    return
      end
      subroutine assmb(ek,ef,nee,nodd)
c
c....To assemble the global matrix from every element contribution
c....Valid only symmetry matrix
c
      level 2, gk,gf,nz
      common/cmatrx/ gk(1200,25),gf(1200),nz
      dimension ek(9,9),ef(9),nodd(9)
      do 10 ii=1,nee
        ig=nodd(ii)
        gf(ig)=gf(ig)+ef(ii)
        do 10 jj=1,nee
          jg=nodd(jj)-ig+1
          if(jg.le.0) go to 10
          gk(ig,jg)=gk(ig,jg)+ek(ii,jj)
10      continue
      return
      end
      subroutine assmb4(ek,ef,nee,nodd,gk,gf)
c
c....To assemble the global matrix from every element contribution
c....Valid only symmetry matrix
c
      dimension ek(3,3),ef(3),nodd(9),gk(25,3),gf(25)
      do 10 ii=1,nee
        ig=nodd(ii)
        gf(ig)=gf(ig)+ef(ii)
        do 10 jj=1,nee
          jg=nodd(jj)-ig+1
          if(jg.le.0) go to 10
          gk(ig,jg)=gk(ig,jg)+ek(ii,jj)
10      continue
      return
      end
      subroutine bcint(p,v,x,pe,game,nee,ns,uu)
```

```
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      dimension game(9),pe(9,9),x(2,9),xy(2),dxds(2,2)
      dimension psi(9),dpsi(9,2)
      dimension uu(9),aa(9,9),ab(9),save(9)
      do 10 i=1,nee
       game(i)=0.0
       do 10 j=1,nee
         pe(i,j)=0.0
10    continue
      do 70 loop=1,none
       if(ns.eq.1) then
       xy(2)=-1.0
       xy(1)=xint1(loop)
       else if(ns.eq.2) then
       xy(1)=1.0
       xy(2)=xint1(loop)
       else if(ns.eq.3) then
       xy(2)=1.0
       xy(1)=xint1(loop)
       else if(ns.eq.4) then
       xy(1)=-1.0
       xy(2)=xint1(loop)
       else
       write(6,200)
200    format(2x,#Out of range in bcint.f#,)
       end if
       call shape(xy,nee,psi,dpsi)
       do 20 i=1,2
         do 20 j=1,2
           dxds(i,j)=0.0
           do 20 k=1,nee
             dxds(i,j)=dxds(i,j)+dpsi(k,j)*x(i,k)
20     continue
       if(nopt1.eq.1) then
       rx=0.0
       do 22 i=1,nee
22       rx=rx+x(1,i)*psi(i)
       else
       rx=1.0
       end if
       if(ns.eq.1.or.ns.eq.3) then
          valu=dxds(1,1)**2+dxds(2,1)**2
       else
          valu=dxds(1,2)**2+dxds(2,2)**2
       end if
       vjaco=sqrt(valu)
       fac=vjaco*wint1(loop)*rx
         do 40 ia=1,nee
           ab(ia)=v*psi(ia)
           do 40 ib=1,nee
40           aa(ia,ib)=p*psi(ia)*psi(ib)
         do 45 ic=1,nee
```

```
          save(ic)=0.0
          do 45 id=1,nee
45          save(ic)=save(ic)+(thet-1.0)*aa(ic,id)*uu(id)
        do 46 im=1,nee
          game(im)=game(im)+(ab(im)+save(im))*fac
          do 46 in=1,nee
            pe(im,in)=pe(im,in)+thet*aa(im,in)*fac
46      continue
70    continue
      return
      end
      subroutine comn
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      call oneint(none)
      call twoint(ntwo)
      return
      end
      subroutine data
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
c
c....nopt1=1 is axisymmetry, otherwise plane 2-D
c....nopt2=1 is transient, otherwise steady state
c....nopt3=1 is linear, otherwise nonlinear
c....time   time for transient calculations
c....tmax   max. time in calculation
c....delt   time step increment
c....thet   theta method in time integration scheme
c....    thet=1    Euler Backward Method
c....    thet=2/3  Galerkin Method
c....    thet=1/2  Midpoint or Crank-Nicolson Method
c....    thet=0    Euler Forward Method
c....nprint   no of steps for print out
c....niter    no of iterations for nonlinear case
c....tolen    allowed tolerance in iteration
c
      read(5,*) nopt1,nopt2,nopt3
      if(nopt1.eq.1) then
        write(6,100)
100     format(1h1,/////2x,#THIS IS A 3-D AXISYMMETRY PROBLEM#,/)
      else
        write(6,110)
110     format(/2x,#THIS IS A 2-D PLANE PROBLEM#,/)
      end if
      if(nopt2.eq.1) then
        write(6,120)
120     format(/2x,#THIS IS A TRANSIENT PROBLEM#,/)
      else
        write(6,130)
130     format(/2x,#THIS IS A STEADY STATE PROBLEM#,/)
      end if
      if(nopt3.eq.1) then
```

```
        write(6,140)
140     format(/2x,#THIS IS A LINEAR PROBLEM#,/)
      else
        write(6,150)
150     format(/2x,#THIS IS A NONLINAR PROBLEM#,/)
      end if
      read(5,*) nprint,niter,nwrt
      write(6,160) nprint
160   format(/2x,#THE NO. OF STEPS FOR PRINT OUT RESULTS IS#,i5/)
      write(6,170) niter
170   format(/2x,#THE NO. OF ITERATIONS IS#,i5/)
      read(5,*) delt,tmax,thet,tolen
      write(6,180) delt,tmax,thet
180   format(/2x,#THE TIME INCREMENT IS#,e15.5,/2x,#THE MAX. TIME IS#,
     .e15.5,/2x,#THE THETA IS#,e15.5,/)
      read(5,*) xdim,ydim
      write(6,190) xdim,ydim
190   format(/2x,#THE GLOBAL X-DIMENSION IS#,e15.5,/2x,#THE GLOBAL#,
     .# Y-DIMENSION IS#,e15.5,/)
      read(5,*) none,ntwo
      write(6,200) none,ntwo
200   format(/2x,#THE NO. OF 1-D INTEGRATION PT. IS#,i5,
     ./2x,#THE NO. OF 2-D INTEGRATION PT. IS#,i5,/)
      do 10 i=1,4
      read(5,*) (prop(i,j),j=1,4)
10    write(6,210) i,(prop(i,j),j=1,4)
210   format(/2x,#THE FOLLOWINGS ARE THE DEFAULT COEFFICIENTS IN#,
     .# REGION#,i5,//2x,4e15.5,/)
      return
      end
      subroutine divdi4(x,n,d)
c....To get divided differences for interpolation
c....d and x are vectors with entries f(x(i)) and x(i),
c....i=1,...,n respectively. On exit d(i) will contain
c....f[x(1),...,x(i)]
      dimension x(5),d(5)
      do 20 i=1,n-1
        j=n
10      d(j)=(d(j)-d(j-1))/(x(j)-x(j-i))
        j=j-1
        if(j.ge.(i+1)) go to 10
20    continue
      return
      end
      subroutine elem(x,n,ek,ef,matt,uu)
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      dimension ck(9,9),cc(9,9),xy(2),ek(9,9),ef(9)
      dimension x(2,9),psi(9),dpsi(9,2)
      dimension dpsix(9),dpsiy(9),dxds(2,2),dsdx(2,2)
      dimension ct(9,9),uu(9),save(9)
c
```

```
      do 10 i=1,n
        ef(i)=0.0
        do 10 j=1,n
          ek(i,j)=0.0
10    continue
      call getmat(x,n,uu,aa,ab,ac,ad,matt)
      do 70 loop=1,ntwo
        xy(1)=xint2(loop,1)
        xy(2)=xint2(loop,2)
        call shape(xy,n,psi,dpsi)
        do 20 i=1,2
          do 20 j=1,2
            dxds(i,j)=0.0
            do 20 k=1,n
              dxds(i,j)=dxds(i,j)+dpsi(k,j)*x(i,k)
20      continue
        if(nopt1.eq.1) then
          rx=0.0
          do 25 ii=1,n
25          rx=rx+x(1,ii)*psi(ii)
        else
          rx=1.0
        end if
        detj=dxds(1,1)*dxds(2,2)-dxds(1,2)*dxds(2,1)
        if(detj.le.0.0) go to 99
        dsdx(1,1)=dxds(2,2)/detj
        dsdx(2,2)=dxds(1,1)/detj
        dsdx(1,2)=-dxds(1,2)/detj
        dsdx(2,1)=-dxds(2,1)/detj
        do 30 i=1,n
          dpsix(i)=dpsi(i,1)*dsdx(1,1)+dpsi(i,2)*dsdx(2,1)
          dpsiy(i)=dpsi(i,1)*dsdx(1,2)+dpsi(i,2)*dsdx(2,2)
30      continue
        fac=detj*wint2(loop)*rx
        do 40 i=1,n
          do 40 j=1,n
            ck(i,j)=aa*(dpsix(i)*dpsix(j)+dpsiy(i)*dpsiy(j))
            cc(i,j)=ab/delt*psi(i)*psi(j)
            ct(i,j)=(thet-1.0)*ck(i,j)+cc(i,j)
40      continue
        do 45 i=1,n
          save(i)=0.0
          do 45 k=1,n
45      save(i)=save(i)+ct(i,k)*uu(k)
        do 50 i=1,n
          ef(i)=ef(i)+fac*save(i)
          do 50 j=1,n
            ek(i,j)=ek(i,j)+(thet*ck(i,j)+cc(i,j))*fac
50      continue
70    continue
      return
99    write(6,100)
100   format(2x,#Bad Jacobian Matrx#,/)
      stop
```

```
      end
      subroutine elem4(xx,nee,ek,ef,tmatt,nn,mm,nnode1)
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cnode/ x12(2,1200),u12(1200)
      common/celem/ ne(1000),node(9,1000)
      dimension xx(2,9),save(2),ek(3,3),ef(3),dtdx(2)
      dimension xy(2),psi(9),dpsi(9,2),uu(9),xx12(2,9),nod12(9)
      do 1 ii=1,nee
        ef(ii)=0.0
        do 1 jj=1,nee
1        ek(ii,jj)=0.0
      if(nopt1.eq.1) then
        rx=0.50*(xx(1,1)+xx(1,2))
      else
        rx=1.0
      end if
      a=(xx(1,1)-xx(1,2))**2
      b=(xx(2,1)-xx(2,2))**2
c....Only valid for 4-node element
      detj=0.50*sqrt(a+b)
c....To get two directional cosine
      slop=(xx(2,2)-xx(2,1))/(xx(1,2)-xx(1,1))
      if(slop.le.1.0e-7) then
        theta=asin(1.0)
      else
        theta=atan(-1.0/slop)
      end if
      dxdn=cos(theta)
      dydn=sin(theta)
      fac=rx*detj
      bx=tmatt/delt
      do 40 in=1,none
        xy(2)=1.0
        xy(1)=xint1(in)
        call shape(xy,4,psi,dpsi)
        nee1=ne(nn)
        tmat1=prop(1,1)
        do 5 ja=1,nee1
          nod12(ja)=node(ja,nn)
          nj=nod12(ja)
          xx12(1,ja)=x12(1,nj)
          xx12(2,ja)=x12(2,nj)
          uu(ja)=u12(nj)
5       continue
        call flux4(xx12,nee1,tmat1,uu,dpsi,dtdx)
        do 6 mc=1,2
6       save(mc)=dtdx(mc)
        nee1=ne(mm)
        tmat1=prop(2,1)
        do 10 ja=1,nee1
          nod12(ja)=node(ja,mm)
```

```
          nj=nod12(ja)+nnode1
          xx12(1,ja)=x12(1,nj)
          xx12(2,ja)=x12(2,nj)
          uu(ja)=u12(nj)
10      continue
        call flux4(xx12,nee1,tmat1,uu,dpsi,dtdx)
        do 11 na=1,2
11        save(na)=save(na)-dtdx(na)
        coef=dxdn*save(1)+dydn*save(2)
        do 20 ii=1,nee
c....Node 3 and 4
          j=ii+2
          ef(ii)=ef(ii)+coef*fac*psi(j)*wint1(in)
          do 20 k=1,nee
c....Node 3 and 4
            m=k+2
            ek(ii,k)=ek(ii,k)+bx*fac*psi(j)*psi(m)*wint1(in)
20      continue
40    continue
      return
      end
      subroutine flux4(xx,nee,tmatt,uu,dpsi,dtdx)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      dimension dpsix(9),dpsiy(9),xx(2,9),uu(9),dtdx(2)
      dimension dpsi(9,2),dxds(2,2),dsdx(2,2)
      do 1 if=1,2
1       dtdx(if)=0.0
        do 10 ii=1,2
          do 10 im=1,2
            dxds(ii,im)=0.0
            do 10 if=1,nee
              dxds(ii,im)=dxds(ii,im)+dpsi(if,im)*xx(ii,if)
10      continue
      det=dxds(1,1)*dxds(2,2)-dxds(1,2)*dxds(2,1)
      if(det.le.0.0) go to 1000
      dsdx(1,1)=dxds(2,2)/det
      dsdx(2,2)=dxds(1,1)/det
      dsdx(1,2)=-dxds(1,2)/det
      dsdx(2,1)=-dxds(2,1)/det
      do 20 io=1,nee
        dpsix(io)=dpsi(io,1)*dsdx(1,1)+dpsi(io,2)*dsdx(2,1)
        dpsiy(io)=dpsi(io,1)*dsdx(1,2)+dpsi(io,2)*dsdx(2,2)
20    continue
      do 30 jp=1,nee
        dtdx(1)=dtdx(1)+dpsix(jp)*uu(jp)*tmatt
        dtdx(2)=dtdx(2)+dpsiy(jp)*uu(jp)*tmatt
30    continue
      return
1000  write(6,100)
100   format(/2x,#BAD JACOBIAN MATRIX IN FLUX4.F#,)
      stop
      end
      subroutine form1
      level 2, gk,gf,nz
```

```
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      dimension xx(2,9),uu(9),nodd(9),ek(9,9),ef(9)
c
      do 10 i=1,nnode1
        gf(i)=0.0
        do 10 j=1,25
          gk(i,j)=0.0
10    continue
      do 40 ielm=1,nelem1
        nee=ne(ielm)
        matt=nz
        do 30 j=1,nee
          nodd(j)=node(j,ielm)
          nj=nodd(j)
          xx(1,j)=x12(1,nj)
          xx(2,j)=x12(2,nj)
          uu(j)=u12(nj)
30      continue
        call elem(xx,nee,ek,ef,matt,uu)
        call assmb(ek,ef,nee,nodd)
40    continue
      return
      end
      subroutine form2
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      dimension xx(2,9),uu(9),nodd(9),ek(9,9),ef(9)
c
      do 10 i=1,nnode2
        gf(i)=0.0
        do 10 j=1,25
          gk(i,j)=0.0
10    continue
      do 40 ielm=1,nelem2
        iielm=ielm+nelem1
        nee=ne(iielm)
        matt=nz
        do 30 j=1,nee
          nodd(j)=node(j,iielm)
```

```
          nj=nodd(j)+nnode1
          xx(1,j)=x12(1,nj)
          xx(2,j)=x12(2,nj)
          uu(j)=u12(nj)
30      continue
        call elem(xx,nee,ek,ef,matt,uu)
        call assmb(ek,ef,nee,nodd)
40    continue
      return
      end
      subroutine form3
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/celem/ ne(1000),node(9,1000)
      common/cnode3/ x3(2,1200),u3(1200)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      dimension xx(2,9),uu(9),nodd(9),ek(9,9),ef(9)
c
      do 10 i=1,nnode3
        gf(i)=0.0
        do 10 j=1,25
          gk(i,j)=0.0
10    continue
      do 40 ielm=1,nelem3
        nee=ne(ielm)
        matt=nz
        do 30 j=1,nee
          nodd(j)=node(j,ielm)
          nj=nodd(j)
          xx(1,j)=x3(1,nj)
          xx(2,j)=x3(2,nj)
          uu(j)=u3(nj)
30      continue
        call elem(xx,nee,ek,ef,matt,uu)
        call assmb(ek,ef,nee,nodd)
40    continue
      return
      end
      subroutine form4
      common/cinter/xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .              angl(25),capa(2,25),tin(25),ratio(50)
      common/celem4/ ne4(25),nodes4(3,25)
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cdoman/ xdim,ydim,prop(4,4)
```

```
      dimension nodd(9),xx(2,9),ek(3,3),ef(3)
      do 10 i=1,max124
        gf4(i)=0.0
c....Matrix band width is 2 here,one-D two-node symmetry
         do 10 j=1,2
           gk4(i,j)=0.0
10    continue
c....Get first element No. along the interface in region 1 & 2
      nn=nfirst(1)
      mm=nfirst(2)+nelem1
c....Do the assembly processes
      do 60 ito=1,max124-1
      nee=ne4(ito)
      tmatt=prop(4,3)
      do 40 j=1,nee
        nodd(j)=nodes4(j,ito)
        nj=nodd(j)
        xx(1,j)=xi124(1,nj)
        xx(2,j)=xi124(2,nj)
40    continue
      call elem4(xx,nee,ek,ef,tmatt,nn,mm,nnode1)
      call assmb4(ek,ef,nee,nodd,gk4,gf4)
      nn=nn+1
      mm=mm+1
60    continue
      return
      end
      subroutine getmat(x,n,u,aa,ab,ac,ad,matt)
c....To compute the coefficients in the differential equations
c.... at any specific point and time, in other words, coefficients
c.... can be function of time, temperature(concentration),and
c.... space. Used in nonlinear differential equations
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      dimension x(2,9),u(9)
      aa=prop(matt,1)
      ab=prop(matt,2)
      ac=prop(matt,3)
      ad=prop(matt,4)
      return
      end
      subroutine gett
      common/cnode3/ x3(2,1200),u3(1200)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      dimension cc(2,95)
      cc(1,1)=0.0
      cc(2,1)=0.0
      cc(1,2)=17.0
      cc(2,2)=-0.0620
```

```
cc(1,3)=34.0
cc(2,3)=-0.1210
cc(1,4)=51.0
cc(2,4)=-0.1810
cc(1,5)=69.0
cc(2,5)=-0.240
cc(1,6)=86.0
cc(2,6)=-0.2990
cc(1,7)=103.0
cc(2,7)=-0.3580
cc(1,8)=120.0
cc(2,8)=-0.4170
cc(1,9)=137.0
cc(2,9)=-0.4750
cc(1,10)=155.0
cc(2,10)=-0.5340
cc(1,11)=172.0
cc(2,11)=-0.5930
cc(1,12)=189.0
cc(2,12)=-0.6520
cc(1,13)=207.0
cc(2,13)=-0.7110
cc(1,14)=224.0
cc(2,14)=-0.770
cc(1,15)=241.0
cc(2,15)=-0.8290
cc(1,16)=259.0
cc(2,16)=-0.8880
cc(1,17)=276.0
cc(2,17)=-0.9480
cc(1,18)=294.0
cc(2,18)=-1.0070
cc(1,19)=311.0
cc(2,19)=-1.0670
cc(1,20)=329.0
cc(2,20)=-1.1260
cc(1,21)=346.0
cc(2,21)=-1.1860
cc(1,22)=364.0
cc(2,22)=-1.2460
cc(1,23)=382.0
cc(2,23)=-1.3060
cc(1,24)=399.0
cc(2,24)=-1.3660
cc(1,25)=418.0
cc(2,25)=-1.4260
cc(1,26)=435.0
cc(2,26)=-1.4860
cc(1,27)=452.0
cc(2,27)=-1.5470
cc(1,28)=470.0
cc(2,28)=-1.6070
cc(1,29)=488.0
cc(2,29)=-1.6680
```

```
cc(1,30)=505.0
cc(2,30)=-1.7290
cc(1,31)=523.0
cc(2,31)=-1.790
cc(1,32)=541.0
cc(2,32)=-1.8510
cc(1,33)=559.0
cc(2,33)=-1.9130
cc(1,34)=577.0
cc(2,34)=-1.9740
cc(1,35)=595.0
cc(2,35)=-2.0360
cc(1,36)=613.0
cc(2,36)=-2.0980
cc(1,37)=631.0
cc(2,37)=-2.1600
cc(1,38)=649.0
cc(2,38)=-2.2220
cc(1,39)=667.0
cc(2,39)=-2.2840
cc(1,40)=685.0
cc(2,40)=-2.3470
cc(1,41)=703.0
cc(2,41)=-2.4090
cc(1,42)=721.0
cc(2,42)=-2.4720
cc(1,43)=739.0
cc(2,43)=-2.5350
cc(1,44)=757.0
cc(2,44)=-2.5980
cc(1,45)=775.0
cc(2,45)=-2.6620
cc(1,46)=794.0
cc(2,46)=-2.7250
cc(1,47)=812.0
cc(2,47)=-2.7890
cc(1,48)=830.0
cc(2,48)=-2.8530
cc(1,49)=848.0
cc(2,49)=-2.9170
cc(1,50)=866.0
cc(2,50)=-2.9820
cc(1,51)=865.0
cc(2,51)=-3.0460
cc(1,52)=921.0
cc(2,52)=-3.1760
cc(1,53)=958.0
cc(2,53)=-3.3070
cc(1,54)=995.0
cc(2,54)=-3.4360
cc(1,55)=1032.0
cc(2,55)=-3.5700
cc(1,56)=1069.0
cc(2,56)=-3.7030
```

```
cc(1,57)=1106.0
cc(2,57)=-3.8370
cc(1,58)=1144.0
cc(2,58)=-3.9720
cc(1,59)=1181.0
cc(2,59)=-4.1070
cc(1,60)=1218.0
cc(2,60)=-4.2440
cc(1,61)=1256.0
cc(2,61)=-4.3780
cc(1,62)=1294.0
cc(2,62)=-4.5160
cc(1,63)=1331.0
cc(2,63)=-4.6550
cc(1,64)=1369.0
cc(2,64)=-4.7950
cc(1,65)=1407.0
cc(2,65)=-4.9370
cc(1,66)=1445.0
cc(2,66)=-5.0790
cc(1,67)=1484.0
cc(2,67)=-5.2220
cc(1,68)=1522.0
cc(2,68)=-5.3670
cc(1,69)=1560.0
cc(2,69)=-5.5120
cc(1,70)=1599.0
cc(2,70)=-5.6590
cc(1,71)=1637.0
cc(2,71)=-5.8070
cc(1,72)=1676.0
cc(2,72)=-5.9560
cc(1,73)=1715.0
cc(2,73)=-6.1060
cc(1,74)=1754.0
cc(2,74)=-6.2580
cc(1,75)=1793.0
cc(2,75)=-6.4100
cc(1,76)=1832.0
cc(2,76)=-6.5640
cc(1,77)=1930.0
cc(2,77)=-6.9540
cc(1,78)=2029.0
cc(2,78)=-7.3530
cc(1,79)=2129.0
cc(2,79)=-7.760
cc(1,80)=2229.0
cc(2,80)=-8.1760
cc(1,81)=2330.0
cc(2,81)=-8.6020
cc(1,82)=2432.0
cc(2,82)=-9.0380
cc(1,83)=2534.0
cc(2,83)=-9.4840
```

```
      cc(1,84)=2637.0
      cc(2,84)=-9.940
      cc(1,85)=2741.0
      cc(2,85)=-10.4080
      cc(1,86)=2845.0
      cc(2,86)=-10.8380
      cc(1,87)=3056.0
      cc(2,87)=-11.8850
      cc(1,88)=3270.0
      cc(2,88)=-12.935
      cc(1,89)=3486.0
      cc(2,89)=-14.0440
      cc(1,90)=3706.0
      cc(2,90)=-15.2160
      cc(1,91)=3928.0
      cc(2,91)=-16.4580
      cc(1,92)=4153.0
      cc(2,92)=-17.776
      cc(1,93)=4382.0
      cc(2,93)=-19.1760
      cc(1,94)=4613.0
      cc(2,94)=-20.6670
      cc(1,95)=4800.0
      cc(2,95)=-21.50
      is=nnode3-nmax3+1
c..............................
      factor1=0.1e2
      factor2=0.1e3
      iaa=1000
      iaaa=2000
      ibb=2000
      ibbb=20000
      iperiod=100
c..............................
      pi=acos(-1.0)
      if(icount.ge.iaa.and.icount.le.iaaa) then
        k=1
        do 5 i=is,nnode3
          degree=xi3(1,k)*pi/xi3(1,nmax3)
          u3(i)=u3(i)+factor1*cos(degree)
          k=k+1
5       continue
      end if
      if(icount.ge.ibb.and.icount.le.ibbb) then
        igap=(icount-ibb)/iperiod
        itest=igap/2*2
        ivalue=igap*iperiod+ibb
        if(itest.eq.igap) then
          degree=(icount-ivalue)*pi/iperiod
          do 7 ii=is,nnode3
            u3(ii)=u3(ii)+factor2*cos(degree)
7         continue
        else
          degree=(icount-ivalue)*pi/iperiod
```

```
      do 9 k=is,nnode3
        u3(k)=u3(k)-factor2*cos(degree)
9     continue
    end if
  end if
  j=1
  do 50 i=is,nnode3,2
    n1=1
    n2=95
    if(u3(i).lt.cc(1,1).or.u3(i).gt.cc(1,95)) go to 1000
10  nhalf=(n1+n2)/2
    if(nhalf.eq.n1.or.nhalf.eq.n2) go to 20
    if(u3(i).eq.cc(1,nhalf)) then
      tin(j)=cc(2,nhalf)
      j=j+1
      go to 50
    else if(u3(i).gt.cc(1,nhalf)) then
      n1=nhalf
      go to 10
    else if(u3(i).lt.cc(1,nhalf)) then
      n2=nhalf
      go to 10
    end if
20  rato=(cc(2,n1)-cc(2,n2))/(cc(1,n1)-cc(1,n2))
    tin(j)=cc(2,n1)+rato*(u3(i)-cc(1,n1))
    j=j+1
50 continue
  return
1000 write(6,500)
500 format(2x,#Out of range in gett.f#,/)
  stop
  end
  subroutine init
  common/cnode/ x12(2,1200),u12(1200)
  common/celem/ ne(1000),node(9,1000)
  common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
  common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
 .            ,npt1(10),vpt1(10)
  common/cscal1/ rlen1,rval1
  common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
  common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
 .            ,npt2(10),vpt2(10)
  common/cscal2/ rlen2,rval2
  common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
  common/cnode3/ x3(2,1200),u3(1200)
  common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
 .            ,npt3(10),vpt3(10)
  common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
 .              ,angl(25),capa(2,25),tin(25),ratio(50)
  common/celem4/ ne4(25),nodes4(3,25)
  common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
  common/cooman/ xdim,ydim,prop(4,4)
c
  call init4
```

```
      call init1
      call init2
      call init3
      return
      end
      subroutine init1
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cscal1/ rlen1,rval1
      common/cnode/ x12(2,1200),u12(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
     .             ,npt1(10),vpt1(10)
      common/cdoman/ xdim,ydim,prop(4,4)
c
      write(6,1000)
1000  format(//2x,#**** THE FOLLOWING IS THE INITIAL INFORMATION#,
     .# OF REGION 1 ****#,/)
      read(5,*) rlen1,rval1
      write(6,100) rlen1,rval1
100   format(/2x,#THE REFERENCE LENGTH FOR MESHING REGION 1 IS#,e15.5,
     ./2x,#THE REFERENCE VALUE FOR MESHING REGION 1 IS#,e15.5,)
      read(5,*) npot1,nbc12
      nbc11=2*max124
      read(5,*) t0,t1
      do 10 i=1,max124
        ndbc11(i)=i
        u12(i)=t0
        vbc11(i)=t0
        tin(i)=t1
        j=i+max124
        ndbc11(j)=j
        vbc11(j)=t1
        u12(j)=t1
10    continue
      do 20 i=1,max124
        x12(1,i)=xi124(1,i)
        x12(2,i)=0.0
        j=i+max124
        x12(1,j)=xi124(1,i)
        x12(2,j)=xi124(2,i)
20    continue
      nnode1=max124*2
      nel=max124-1
      mrow=1
      nfirst(1)=1
      nelem1=nel*mrow
      do 30 i=1,nelem1
30      ne(i)=4
      icon=0
      do 70 i=1,mrow
        n1=i*max124
        n2=n1-max124
```

```
        do 60 j=1,nel
          icon=icon+1
          node(1,icon)=n2+j
          node(2,icon)=n2+j+1
          node(3,icon)=n1+j+1
          node(4,icon)=n1+j
60      continue
70    continue
      write(6,200) nnode1,nelem1
      write(6,300) npot1,nbc11,nbc12
      if(npot1.eq.0) go to 110
      write(6,350)
      write(6,400) (i,npt1(i),vpt1(i),i=1,npot1)
110   if(nbc11.eq.0) go to 120
      write(6,450)
      write(6,400) (i,ndbc11(i),vbc11(i),i=1,nbc11)
120   if(nbc12.eq.0) go to 130
      write(6,550)
      write(6,600) (i,neb12(i),nsde12(i),vbc12(1,i),vbc12(2,i),
     .i=1,nbc12)
130   write(6,750)
      write(6,800) (i,x12(1,i),x12(2,i),u12(i),i=1,nnode1)
200   format(/2x,#THE NO OF NODES IS#,i5,
     ./2x,#THE NO OF ELEMENTS IS#,i5,)
300   format(/2x,#THE NO OF POINT SOURCES IS#,i5,
     ./2x,#THE NO OF ESSENTIAL BOUNDARY COND. IS#,i5,
     ./2x,#THE NO OF NATURAL BOUNDARY COND. IS#,i5)
350   format(/5x,#NO#,5x,#NODE#,3x,#POINT SOURCE VALUE#,)
400   format(2x,i5,3x,i5,e15.5)
450   format(/5x,#NO#,5x,#NODE#,3x,#ESSEN BOUND VALUE#,)
550   format(/5x,#NO#,5x,#ELEMENT#,5x,#SIDE#,5x,#P#,5x,#GAMA#,)
600   format(2x,3i6,2e15.5)
750   format(/2x,#NODE NO#,8x,#X#,14x,#Y#,12x,#TEMP#,)
800   format(2x,i5,3e15.5)
      return
      end
      subroutine init2
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cnode/ x12(2,1200),u12(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cscal2/ rlen2,rval2
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .             ,npt2(10),vpt2(10)
c
      write(6,1000)
1000  format(//2x,#**** THE FOLLOWING IS THE INITIAL INFORMATION#,
     .# OF REGION 2 ****#,/)
      read(5,*) rlen2,rval2
      write(6,100) rlen2,rval2
100   format(/2x,#THE REFERENCE LENGTH FOR MESHING REGION 2 IS#,e15.5,
```

```
     ./2x,#THE REFERENCE VALUE FOR MESHING REGION 2 IS#,e15.5,)
      ylen=(ydim-xi124(2,1))/rlen2
      num=int(ylen)
      mp=num+2
      nnode2=mp*max124
      ydis=(ydim-xi124(2,1))/float(num)
      do 50 i=1,max124
        do 40 j=1,mp
          j1=j-1
          nmm=j1*max124+i
          nc=nmm+nnode1
          if(j.eq.(mp-1)) go to 25
          if(j.eq.mp) go to 30
          x12(1,nc)=xi124(1,i)
          x12(2,nc)=ydim-float(j1)*ydis
          go to 40
25        x12(1,nc)=xi124(1,i)
          x12(2,nc)=ydim-(float(j1)-rval2)*ydis
          go to 40
30        x12(1,nc)=xi124(1,i)
          x12(2,nc)=xi124(2,i)
40      continue
50    continue
      nel=max124-1
      mrow=num+1
      nfirst(2)=nel*num+1
      nelem2=nel*mrow
      icon=nelem1
      do 70 i=1,mrow
        n1=i*max124
        n2=n1-max124
        do 60 j=1,nel
          icon=icon+1
          node(1,icon)=n2+j+1
          node(2,icon)=n2+j
          node(3,icon)=n1+j
          node(4,icon)=n1+j+1
60      continue
70    continue
      do 80 i=1,nelem2
        ii=i+nelem1
80      ne(ii)=4
      read(5,*) npot2,nbc22
      nbc21=2*max124
      read(5,*) ti,tj
      k=nnode2-max124+1
      do 90 i=1,max124
        ndbc21(i)=i
        vbc21(i)=ti
        j=max124+i
        ndbc21(j)=k
        vbc21(j)=tj
        k=k+1
90    continue
```

```
      np=nnode2-max124
      do 95 i=1,np
        j=i+nnode1
95      u12(j)=ti
      do 96 i=np+1,nnode2
        j=i+nnode1
96      u12(j)=tj
      write(6,200) nnode2,nelem2
      write(6,300) npot2,nbc21,nbc22
      if(npot2.eq.0) go to 110
      write(6,350)
      write(6,400) (i,npt2(i),vpt2(i),i=1,npot2)
110   if(nbc21.eq.0) go to 120
      write(6,450)
      write(6,400) (i,ndbc21(i),vbc21(i),i=1,nbc21)
120   if(nbc22.eq.0) go to 130
      write(6,550)
      write(6,600) (i,neb22(i),nsde22(i),vbc22(1,i),vbc22(2,i),
     .i=1,nbc22)
130   ii=nnode1+1
      if=nnode1+nnode2
      j=1
      write(6,750)
      do 150 i=ii,if
        id=j/11
        ic=11*id
        if(j.eq.ic.or.j.eq.1) then
          write(6,800) j,x12(1,i),x12(2,i),u12(i)
        end if
150   j=j+1
200   format(/2x,#THE NO OF NODES IS#,i5,/2x,
     .#THE NO OF ELEMENT IS#,i5)
300   format(/2x,#THE NO OF POINT SOURCES IS#,i5,
     ./2x,#THE NO OF ESSENTIAL BOUNDARY COND. IS#,i5,
     ./2x,#THE NO OF NATURAL BOUNDARY COND. IS#,i5)
350   format(/5x,#NO#,5x,#NODE#,5x,#POINT SOURCE VALUE#,)
400   format(2x,i5,3x,i5,e15.5)
450   format(/5x,#NO#,5x,#NODE#,3x,#ESSEN BOUND VALUE#,)
550   format(/5x,#NO#,5x,#ELEMENT#,4x,#SIDE#,5x,#P#,5x,#GAMA#,)
600   format(2x,3i6,2e15.5)
750   format(/x,#NODE NO#,8x,#X#,14x,#Y#,12x,#TEMP#,)
800   format(2x,i5,3e15.5)
      return
      end
      subroutine init3
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
      common/cdoman/ xdim,ydim,prop(4,4)
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cnode3/ x3(2,1200),u3(1200)
      common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
     .             ,npt3(10),vpt3(10)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      dimension a(55)
```

```
c
      read(5,*) ny
      nnode3=ny*nmax3
      npot3=0
      nbc31=nmax3
      nbc32=nmax3-1
      read(5,*) cexp
      b=cexp/float(ny-2)
      c=0.0
      do 10 i=1,ny-1
        m=i-1
        a(i)=exp(-float(m)*b)
        c=c+a(i)
10    continue
      d1=(ydim-xi3(2,1))/c
      do 30 j=1,nmax3
        aa=0.0
        kk=0
        do 20 k=j,nnode3,nmax3
          x3(2,k)=ydim-d1*aa
          x3(1,k)=xi3(1,j)
          kk=kk+1
          if(kk.lt.ny) aa=aa+a(kk)
          ki=k
20      continue
        x3(2,ki)=xi3(2,j)
30    continue
      nel=nmax3-1
      mrow=ny-1
      nelem3=nel*mrow
      read(5,*) ci
      do 70 i=1,nnode3
70      u3(i)=ci
      do 80 i=1,nbc31
        ndbc31(i)=i
80      vbc31(i)=ci
      nn=nel*(mrow-1)+1
      do 100 i=1,nbc32
        neb32(i)=nn
        nsde32(i)=3
        vbc32(1,i)=0.0
        vbc32(2,i)=0.0
        nn=nn+1
100   continue
      write(6,1000)
1000  format(//2x,#**** THE FOLLOWING IS THE INITIAL INFORMATION#,
     .# OF REGION 3 ****#,/)
      write(6,200) nnode3,nelem3
      write(6,300) npot3,nbc31,nbc32
      if(npot3.eq.0) go to 110
      write(6,350)
      write(6,400) (i,npt3(i),vpt3(i),i=1,npot3)
110   if(nbc31.eq.0) go to 120
      write(6,450)
```

```
      write(6,400) (i,ndbc31(i),vbc31(i),i=1,nbc31)
120   if(nbc32.eq.0) go to 130
      write(6,550)
      write(6,600) (i,neb32(i),nsde32(i),vbc32(1,i),vbc32(2,i),
     .i=1,nbc32)
130   write(6,750)
      do 150 i=1,nnode3
        id=i/21
        ic=21*id
        if(i.eq.ic.or.i.eq.1) then
          write(6,800) i,x3(1,i),x3(2,i),u3(i)
        end if
150   continue
200   format(/2x,#THE NO OF NODES IS#,i5,
     ./2x,#THE NO OF ELEMENTS IS#,i5)
300   format(/2x,#THE NO OF POINT SOURCES IS#,i5,
     ./2x,#THE NO OF ESSENTIAL BOUNDARY COND. IS#,i5,
     ./2x,#THE NO OF NATURAL BOUNDARY COND. IS#,i5)
350   format(/5x,#NO#,5x,#NODE#,3x,#POINT SOURCE VALUE#,)
400   format(2x,i5,3x,i5,e15.5)
450   format(/5x,#NO#,5x,#NODE#,3x,#ESSEN BOUND VALUE#,)
550   format(/4x,#NO#,3x,#ELEMENT#,4x,#SIDE#,9x,#P#,11x,
     .#GAMA#,)
600   format(2x,i4,4x,i4,5x,i4,2e15.5)
750   format(/2x,#NODE NO#,8x,#X#,14x,#Y#,12x,#CONCEN#,)
800   format(2x,i5,3e15.5)
      return
      end
      subroutine init4
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .              ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/celem4/ ne4(25),nodes4(3,25)
      common/cdoman/ xdim,ydim,prop(4,4)
c
      write(6,1000)
1000  format(//2x,#***** THE FOLLOWING IS THE INITIAL INFORMATION#,
     .# OF INTERFACE, REGION 4 ***** #,/)
      read(5,*) max124,nmax3
      xunit=xdim/float(max124-1)
      do 1 i=1,max124-1
        xi124(1,i)=(i-1)*xunit
1     continue
      xi124(1,max124)=xdim
      read(5,*) si
      do 10 i=1,max124
10      xi124(2,i)=si
      do 20 i=1,nmax3
20      xi3(2,i)=si
      xi3(1,1)=xi124(1,1)
      do 30 i=1,max124-1
        a=xi124(1,i)
        b=xi124(1,i+1)
        j=2*i
```

```
          xi3(1,j)=(a+b)/2.0
          xi3(1,j+1)=b
30      continue
        pi=asin(1.0)
        do 40 i=1,max124
40        angl(i)=pi
        do 50 i=1,max124-1
          ne4(i)=2
          nodes4(1,i)=i
          nodes4(2,i)=i+1
50      continue
        write(6,200) max124,max124-1
        write(6,250)
        write(6,300) (i,xi124(1,i),xi124(2,i),i=1,max124)
        write(6,350)
        write(6,400) (i,ne4(i),nodes4(1,i),nodes4(2,i),i=1,max124-1)
200     format(/2x,#THE NO OF NODES IS#,i5,
       ./2x,#THE NO OF ELEMENTS IS#,i5,)
250     format(/6x,#NO#,10x,#X#,14x,#Y#,)
300     format(2x,i6,2e15.5)
350     format(/2x,#ELEM NO#,4x,#NO OF NODES#,6x,#NODE 1#,
       .6x,#NODE 2#,)
400     format(2x,i5,5x,i8,5x,i8,5x,i8)
        return
        end
        subroutine int4(x,n,d,t,p)
c....On entrance, d and x are vectors containing f[x(1),...,x(i)]
c....and x(i), i=1,...,n, respectively. On exit p will contain
c....the value p(t) of the (n-1)-th degree polynomial interpolating
c....to f on x
        dimension x(5),d(5)
        p=d(n)
        i=n-1
10      p=d(i)+(t-x(i))*p
        i=i-1
        if(i.ge.1) go to 10
        return
        end
        subroutine inter4(x,n,d,t,p,angle,curv)
c....Given vectors d(i)=f[x(1),...,x(i)] and x(i), i=1,...,n
c....Obtain p(t), angle of normal direction and curvature at t
        dimension x(5),d(5)
        p=d(n)
        i=n-1
10      p=d(i)+(t-x(i))*p
        i=i-1
        if(i.ge.1) go to 10
c....Compute the coefficients of the first and second derivatives
c....of Newton Divided Difference Formula
        pder1=d(2)
        do 50 i=3,n
          cc=0.0
          do 40 im=1,i-1
            c=1.0
```

```
          do 30 in=1,i-1
            if(im.eq.in) go to 30
            c=c*(t-x(in))
30        continue
          cc=cc+c
40      continue
        pder1=pder1+d(i)*cc
50    continue
      pder2=2.0*d(3)
      if(n.eq.3) go to 1000
      c=0.0
      do 60 i=1,3
60      c=c+(t-x(i))
      pder2=pder2+d(4)*c*2.0
      if(n.eq.4) go to 1000
      c=0.0
      do 70 i=1,4
        k=i+1
65      if(k.gt.4) go to 70
        c=c+(t-x(i))*(t-x(k))
        k=k+1
        go to 65
70    continue
      pder2=pder2+d(5)*c*2.0
      if(n.eq.5) go to 1000
      write(6,200)
200   format(/2x,#N OUT OF RANGE IN INTER4.F#,)
      stop
c....Get angle of normal direction
1000  if(pder1.le.0.1e-7) then
        angle=asin(1.0)
      else
        slop=-1.0/pder1
        angle=atan(slop)
      end if
c....Compute curvature by formula P"/(1+P'**2)**1.5
      dumny=(1.0+pder1**2)**1.5
      if(abs(pder2).le.0.1e-8) then
        curv=0.0
      else
        curv=abs(pder2)/dumny
      end if
      return
      end
      subroutine mesh1(nnodec,nelemc)
c....To remesh the temperature domain of region 1 in every
c.... time step
c....1. Given interface nodal coordinates xi124(2,25)
c....2. Find Max. y-distance ymax of xi124(2,25)
c....3. Obtain the nodal coordinates along each y-dir
c....4. Finer mesh is constructed at both end-sides
c
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
```

```
     .                ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cscal1/ rlen1,rval1
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
c
c....To find the max. of y coordinate along the interface
      ymax=xi124(2,1)
      do 10 i=2,max124
        if(xi124(2,i).gt.ymax) ymax=xi124(2,i)
10    continue
c....To find the no. of nodes along each y-dir
      ylen=ymax/rlen1+0.5
      num=int(ylen)
      if(num.lt.1) stop
      if(num.eq.1) then
        mp=num+2
      else
        mp=num+3
      end if
      nfirst(1)=(max124-1)*(mp-2)+1
      nnodec=max124*mp
c....For each y-dir obtain the nodal coordinates
c....Special consideration for end side nodes
      do 50 i=1,max124
        ydis=xi124(2,i)/num
        do 40 j=1,mp
          j1=j-1
          nmm=j1*max124+i
          if(j.eq.1) go to 20
          if(j.eq.2) go to 25
          if(j.eq.(mp-1).and.j.ne.mp) go to 26
          if (j.eq.mp) go to 30
          xc(1,nmm)=xi124(1,i)
          xc(2,nmm)=(j1-1)*ydis
          go to 40
20        xc(1,nmm)=xi124(1,i)
          xc(2,nmm)=j1*ydis
          go to 40
25        xc(1,nmm)=xi124(1,i)
          xc(2,nmm)=(j1-rval1)*ydis
          go to 40
26        xc(1,nmm)=xi124(1,i)
          xc(2,nmm)=(j1-rval1-1)*ydis
          go to 40
30        xc(1,nmm)=xi124(1,i)
          xc(2,nmm)=xi124(2,i)
40      continue
50    continue
c....Idetify the element data
c....Only consider 4-node element here
      nel=max124-1
      mrow=num+2
      nelemc=nel*mrow
      icon=0
      do 70 i=1,mrow
```

```
          n1=i*max124
          n2=n1-max124
          do 60 j=1,nel
            icon=icon+1
            nodec(1,icon)=n2+j
            nodec(2,icon)=n2+j+1
            nodec(3,icon)=n1+j+1
            nodec(4,icon)=n1+j
60        continue
70      continue
        return
        end
        subroutine mesh2(nnodec,nelemc)
c....To remesh the temperature domain of region 2 in every.
c.... time step
c....1. Given interface nodal coordinates xi124(2,25)
c....2. Find min. y-distance ymin of xi124(2,25)
c....3. Obtain the nodal coordinates along each y-dir
c....4. Finer mesh is constructed at interface
c
        common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
       .               ,angl(25),capa(2,25),tin(25),ratio(50)
        common/cscal2/ rlen2,rval2
        common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
        common/cdoman/ xdim,ydim,prop(4,4)
        common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
c....rval1=0.5
c....To find the min. of y coordinates along the interface
        ymin=xi124(2,1)
        do 10 i=2,max124
          if(xi124(2,i).lt.ymin) ymin=xi124(2,i)
10      continue
c....To find the no. of nodes along each y-dir
        ylen=(ydim-ymin)/rlen2+0.5
        num=int(ylen)
        mp=num+2
        nnodec=max124*mp
c....For each y-dir obtain the nodal coordinates
        do 50 i=1,max124
          ydis=(ydim-xi124(2,i))/num
          do 40 j=1,mp
            j1=j-1
            nmm=j1*max124+i
            if(j.eq.(mp-1)) go to 25
            if(j.eq.mp) go to 30
            xc(1,nmm)=xi124(1,i)
            xc(2,nmm)=ydim-j1*ydis
            go to 40
25          xc(1,nmm)=xi124(1,i)
            xc(2,nmm)=ydim-(j1-rval2)*ydis
            go to 40
30          xc(1,nmm)=xi124(1,i)
            xc(2,nmm)=xi124(2,i)
```

```
40        continue
50      continue
c....Identify yhe element data
c....Only consider 4-node element here
        nel=max124-1
        nfirst(2)=num*nel+1
        mrow=num+1
        nelemc=nel*mrow
        icon=0
        do 70 i=1,mrow
          n1=i*max124
          n2=n1-max124
          do 60 j=1,nel
            icon=icon+1
            nodec(1,icon)=n2+j+1
            nodec(2,icon)=n2+j
            nodec(3,icon)=n1+j
            nodec(4,icon)=n1+j+1
60        continue
70      continue
        return
        end
        subroutine mesh3
c....To remesh the concentration domain in every time step
c....Given interface nodal coordinates xi3(2,50)
c....Keep the same meshing system except squeeze each y-dir
c.... proportional to ratio(i) respectively
c
        common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .                angl(25),capa(2,25),tin(25),ratio(50)
        common/cdoman/ xdim,ydim,prop(4,4)
        common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
        common/cnode3/ x3(2,1200),u3(1200)
        common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
c....For each y-direction obtain the nodal coordinates
        do 10 i=1,nmax3
          xc(1,i)=xi3(1,i)
          xc(2,i)=ydim
10      continue
        do 50 i=1,nmax3
          do 40 j=i+nmax3,nnode3,nmax3
            xc(1,j)=xi3(1,i)
            xc(2,j)=ydim-(ydim-x3(2,j))*ratio(i)
            jj=j
40       continue
         xc(2,jj)=xi3(2,i)
50      continue
        return
        end
        subroutine mod1(nnodec)
c....To modify the boundary conditions of the temperature region 1
        common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
        common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
```

```
     .              ,npt1(10),vpt1(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .                ,angl(25),capa(2,25),tin(25),ratio(50)
c
      n1=max124+1
      n2=max124*2
      k=nnodec-max124+1
      kk=1
      do 10 i=n1,n2
        ndbc11(i)=k
        vbc11(i)=tin(kk)
        k=k+1
        kk=kk+1
10    continue
c....................................
      iaa=10000
      iaaa=20000
      tvalue=-0.3
      factor1=-0.01
c....................................
      if (icount.ge.iaa.and.icount.le.iaaa) then
        pi=acos(-1.0)
        do 30 i=1,max124
          degree=xi124(1,i)*pi/xi124(1,max124)
          vbc11(i)=tvalue+factor1*cos(degree)
30      continue
      else
        do 40 i=1,max124
          vbc11(i)=tvalue
40      continue
      end if
      return
      end
      subroutine mod2(nnodec)
c....To modify the boundary conditions of the temperature region 2
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .             ,npt2(10),vpt2(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .                ,angl(25),capa(2,25),tin(25),ratio(50)
c
      n1=max124+1
      n2=max124*2
      k=nnodec-max124+1
      kk=1
      do 10 i=n1,n2
        ndbc21(i)=k
        vbc21(i)=tin(kk)
        k=k+1
        kk=kk+1
10    continue
      return
      end
      subroutine mod3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
```

```
      common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
     .              ,npt3(10),vpt3(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      dimension tmp(50)
c
      dc=0.0
      const=1.0-dc
      j=1
      do 10 i=1,max124-1
        a=-vnor(i)
        b=-vnor(i+1)
        c=(a+b)/2.0
        vbc32(1,j)=(a+c)/2.0*const
        j=j+1
        vbc32(1,j)=(b+c)/2.0*const
        j=j+1
10    continue
c..................................
      factor1=0.5e-5
      factor2=-0.5e-4
      iaa=201
      iaaa=10000
      ibb=2100
      ibbb=3000
      iperiod=10
c..................................
      pi=acos(-1.0)
      if(icount.ge.iaa.and.icount.le.iaaa) then
        do 20 k=1,nmax3
          tmp(k)=xi3(1,k)*pi/xi3(1,nmax3)
20      continue
        do 30 k=1,nmax3-1
          degree=0.5*(tmp(k)+tmp(k+1))
          vbc32(1,k)=vbc32(1,k)+factor1*cos(degree)
30      continue
      end if
      if(icount.ge.ibb.and.icount.le.ibbb) then
        igap=(icount-ibb)/iperiod
        itest=igap/2*2
        ivalue=iperiod*igap+ibb
        if(itest.eq.igap) then
          degree=(icount-ivalue)*pi/iperiod
          do 40 k=1,nmax3-1
            vbc32(1,k)=vbc32(1,k)+factor2*cos(degree)
40        continue
        else
          degree=(icount-ivalue)*pi/iperiod
          do 50 k=1,nmax3-1
            vbc32(1,k)=vbc32(1,k)-factor2*cos(degree)
50        continue
        end if
      end if
```

```
      return
      end
      subroutine move
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .              ,angl(25),capa(2,25),tin(25),ratio(50)
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/celem4/ ne4(25),nodes4(3,25)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
      dimension xim(2,25)
c
      call proc4
      call velo4(xim)
      call adst4(xim)
      call post4
      return
      end
      subroutine new1(nnodec,nelemc)
c....To obtain the nodal temp. in the mesh system of domain 1
c....Update all node and element information
c
      common/cnode/ x12(2,1200),u12(1200)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .              ,angl(25),capa(2,25),tin(25),ratio(50)
      common/celem/ ne(1000),node(9,1000)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
      nc=nnodec-(max124*2)+1
      n1=nnode1-max124+1
      do 10 i=1,max124
        if(xc(2,nc).gt.x12(2,n1)) then
          write(6,1000)
1000      format(2x,#Too large of time step ,out of range in new1.f#,)
          stop
        end if
        nc=nc+1
        n1=n1+1
10    continue
      nf=nnodec-max124
      do 50 i=1,max124
        do 40 j=i,nf,max124
          do 30 k=i,nnode1,max124
            if(x12(2,k).lt.xc(2,j)) go to 30
            if(x12(2,k).eq.xc(2,j)) then
              uc(j)=u12(k)
              go to 40
            end if
            kk=k-max124
```

```
            slp=(u12(k)-u12(kk))/(x12(2,k)-x12(2,kk))
            uc(j)=u12(k)+slp*(xc(2,j)-x12(2,k))
            go to 40
30        continue
40      continue
50    continue
      do 55 i=1,max124
        nf=nf+1
        uc(nf)=tin(i)
55    continue
      nnode1=nnodec
      do 70 i=1,nnode1
        u12(i)=uc(i)
        do 60 j=1,2
          x12(j,i)=xc(j,i)
60      continue
70    continue
      nelem1=nelemc
      do 90 i=1,nelem1
        ne(i)=4
        do 80 j=1,4
          node(j,i)=nodec(j,i)
80      continue
90    continue
      return
      end
      subroutine new2(nnodec,nelemc)
c....To obtain the nodal temp. in domain 2
c....Update all node and element information
c
      common/cnode/ x12(2,1200),u12(1200)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/celem/ ne(1000),node(9,1000)
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
      nsum=nnode1+nnode2
      do 50 i=1,max124
        do 40 j=i,nnodec,max124
          l=i+nnode1
          do 30 k=l,nsum,max124
            if(x12(2,k).gt.xc(2,j)) go to 30
            if(x12(2,k).eq.xc(2,j)) then
              uc(j)=u12(k)
              go to 40
            end if
            kk=k-max124
            slp=(u12(k)-u12(kk))/(x12(2,k)-x12(2,kk))
            uc(j)=u12(k)+slp*(xc(2,j)-x12(2,k))
            go to 40
30        continue
40      continue
```

```
50    continue
      nnode2=nnodec
      do 70 i=1,nnode2
        k1=i+nnode1
        u12(k1)=uc(i)
        do 60 j=1,2
          x12(j,k1)=xc(j,i)
60      continue
70    continue
      nelem2=nelemc
      do 90 i=1,nelem2
        k1=i+nelem1
        ne(k1)=4
        do 80 j=1,4
          node(j,k1)=nodec(j,i)
80      continue
90    continue
      return
      end
      subroutine new3
c....To obtain the nodal concentration in the new mesh system
c.... by using old mesh system and old nodal values
c....Update all node and element information
c
      common/cnode3/ x3(2,1200),u3(1200)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/celem/ ne(1000),node(9,1000)
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
      do 10 i=1,nelem3
10      ne(i)=4
      nel=nmax3-1
      mrow=nelem3/nel
      icon=0
      do 20 i=1,mrow
        n1=i*nmax3
        n2=n1-nmax3
        do 15 j=1,nel
          icon=icon+1
          node(1,icon)=n2+j+1
          node(2,icon)=n2+j
          node(3,icon)=n1+j
          node(4,icon)=n1+j+1
15      continue
20    continue
      do 50 i=1,nmax3
         do 40 j=i,nnode3,nmax3
           do 30 k=i,nnode3,nmax3
             if(x3(2,k).gt.xc(2,j)) go to 30
             if(x3(2,k).eq.xc(2,j)) then
               uc(j)=u3(k)
               go to 40
```

```
            end if
            kk=k-nmax3
            slp=(u3(k)-u3(kk))/(x3(2,k)-x3(2,kk))
            uc(j)=u3(k)+slp*(xc(2,j)-x3(2,k))
            go to 40
30        continue
40      continue
50    continue
      do 70 i=1,nnode3
        u3(i)=uc(i)
        do 60 j=1,2
          x3(j,i)=xc(j,i)
60      continue
70    continue
      return
      end
      subroutine oneint(n)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      if(n.eq.1) then
        xint1(1)=0.0
        wint1(1)=2.0
       return
      else if(n.eq.2) then
        xint1(1)=-1.0/sqrt(3.0)
        xint1(2)=-xint1(1)
        wint1(1)=1.0
        wint1(2)=wint1(1)
      return
      else if(n.eq.3) then
        xint1(1)=-sqrt(3.0/5.0)
        xint1(2)=0.0
        xint1(3)=-xint1(1)
        wint1(1)=5.0/9.0
        wint1(2)=8.0/9.0
        wint1(3)=wint1(1)
       return
      else if(n.eq.4) then
        xint1(1)=-0.861136311594053
        xint1(2)=-0.339981043584856
        xint1(3)=-xint1(2)
        xint1(4)=-xint1(1)
        wint1(1)=0.347854845137454
        wint1(2)=0.652145154862546
        wint1(3)=wint1(2)
        wint1(4)=wint1(1)
      return
      else
        write(6,100)
100     format(2x,#Choose the improper value in oneint.f#)
      stop
      end if
      end
      subroutine post1(npass)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
```

```
      common/cnode/ x12(2,1200),u12(1200)
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
c
      write(6,100) time,npass
100   format(//2x,#THE RESULTS OF REGION 1 IS PRINTED AS FOLLOWING #,
     .#AT TIME#,e15.5,/2x,#THE ITERATION NO IS#,i5,)
      write(6,200)
200   format(/2x,#NODE#,10x,#X#,14x,#Y#,14x,#TEMP#,)
      do 5 i=1,nnode1
        if(abs(u12(i)).le.1.0e-15) u12(i)=0.0
5     continue
      do 10 i=1,nnode1
10    write(6,300) i,x12(1,i),x12(2,i),u12(i)
300   format(i6,3e15.5)
      return
      end
      subroutine post2(npass)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cnode/ x12(2,1200),u12(1200)
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
c
      write(6,100) time,npass
100   format(//2x,#THE RESULTS OF REGION 2 IS PRINTED AS FOLLOWING #,
     .#AT TIME#,e15.5,/2x,#THE ITERATION NO IS#,i5,)
      write(6,200)
200   format(/2x,#NODE#,10x,#X#,14x,#Y#,14x,#TEMP#,)
      do 5 i=1,nnode2
        kk=i+nnode1
        if(abs(u12(kk)).le.1.0e-15) u12(kk)=0.0
5     continue
      do 10 i=1,nnode2
        k=i+nnode1
10    write(6,300) i,x12(1,k),x12(2,k),u12(k)
300   format(i6,3e15.5)
      return
      end
      subroutine post3(npass)
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cnode3/ x3(2,1200),u3(1200)
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
c
      write(6,100) time,npass
100   format(//2x,#THE RESULTS OF REGION 3 IS PRINTED AS FOLLOWING #,
     .#AT TIME#,e15.5,/2x,#THE ITERATION NO IS#,i5,)
      write(6,200)
200   format(/2x,#NODE#,10x,#X#,14x,#Y#,14x,#CONC#,)
      do 5 i=1,nnode3
        if(abs(u3(i)).le.1.0e-15) u3(i)=0.0
5     continue
      do 10 i=1,nnode3
10    write(6,300) i,x3(1,i),x3(2,i),u3(i)
300   format(i6,3e15.5)
      return
```

```
      end
      subroutine post4
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      if(icount/nprint*nprint.eq.icount) then
        write(6,10) time
10      format(/2x,#THE FOLLOWING IS PRINTED OUT AT TIME#,e15.5,)
        write(6,20)
20      format(//5x,#NO#,10x,#X#,14x,#Y#,13x,#ANGL#,10x,
     .  #CAPA1#,10x,#CAPA2#,)
        do 30 i=1,max124
30      write(6,40) i,xi124(1,i),xi124(2,i),angl(i),capa(1,i),
     .  capa(2,i)
40      format(2x,i5,5e15.5)
        write(6,50)
50      format(//2x,#NO(CON)#,7x,#X#,14x,#Y#,12x,#RATIO#,)
        do 60 i=1,nmax3
60      write(6,70) i,xi3(1,i),xi3(2,i),ratio(i)
70      format(i6,3e15.5)
      end if
      return
      end
      subroutine prep1
c....To get the nodal coordinates, element information, nodal
c.... values, and boundary conditions
c
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cnode1/ x12(2,1200),u12(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),
     .             vbc12(2,20),npt1(10),vpt1(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
      call mesh1(nnodec,nelemc)
      call new1(nnodec,nelemc)
      call mod1(nnodec)
      call set3(nnode1,nelem1,npot1)
      return
      end
      subroutine prep2
c....To get the nodal coordinates, element information, nodal
c.... values, and boundary conditions
c
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/cnode1/ x12(2,1200),u12(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),
     .             vbc22(2,20),npt2(10),vpt2(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
```

```
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cscal2/ rlen2,rval2
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      mm=nnode1+nnode2
      nn=nelem1+nelem2
      nm=npot1+npot2
c
      call mesh2(nnodec,nelemc)
      call new2(nnodec,nelemc)
      call mod2(nnodec)
      call set3(mm,nn,nm)
      return
      end
      subroutine prep3
c....To get the nodal coordinates, element information, nodal
c.... values, and boundary conditions
c
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cnode3/ x3(2,1200),u3(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),
     .            vbc32(2,40),npt3(10),vpt3(10)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .              angl(25),capa(2,25),tin(25),ratio(50)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
c
      call mesh3
      call new3
      call mod3
      call set3(nnode3,nelem3,npot3)
      return
      end
      subroutine proc1
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
c
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
     .            ,npt1(10),vpt1(10)
      common/cscal1/ rlen1,rval1
      common/cmatrx/ gk(1200,25),gf(1200),nz
      call form1
      call aply1
      return
      end
```

```
      subroutine proc2
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
c
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/celem/ ne(1000),node(9,1000)
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .            ,npt2(10),vpt2(10)
      common/cscal2/ rlen2,rval2
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cnode/ x12(2,1200),u12(1200)
      call form2
      call aply2
      return
      end
      subroutine proc3
      level 2, gk,gf,nz
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/cdoman/ xdim,ydim,prop(4,4)
c
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/cnode3/ x3(2,1200),u3(1200)
      common/celem/ ne(1000),node(9,1000)
      common/cbc3/ ndbc31(50),vbc31(50),ned32(40),nsde32(40),vbc32(2,40)
     .            ,npt3(10),vpt3(10)
      common/cmatrx/  gk(1200,25),gf(1200),nz
      call form3
      call aply3
      return
      end
      subroutine proc4
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .              angl(25),capa(2,25),tin(25),ratio(50)
      common/celem4/ ne4(25),nodes4(3,25)
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cdoman/ xdim,ydim,prop(4,4)
      call form4
      call solve4
      return
      end
      subroutine rhsb(n,ib)
      level 2, gk,gf,nz
```

```
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      np1=n+1
      do 20 i=2,n
        sum=0.0
        k1=min0(ib-1,i-1)
        do 10 k=1,k1
10        sum=sum+gk(i-k,k+1)/gk(i-k,1)*gf(i-k)
20    gf(i)=gf(i)-sum
      uc(n)=gf(n)/gk(n,1)
      do 40 k=2,n
        i=np1-k
        j1=i+1
        j2=min0(n,i+ib-1)
        sum=0.0
        do 30 j=j1,j2
          mm=j-j1+2
30      sum=sum+uc(j)*gk(i,mm)
40      uc(i)=(gf(i)-sum)/gk(i,1)
        return
        end
      subroutine rhsb4(gk,dn,gf,n,ib)
      dimension gk(25,3),gf(25),dn(25)
      np1=n+1
      do 20 i=2,n
        sum=0.0
        k1=min0(ib-1,i-1)
        do 10 k=1,k1
10        sum=sum+gk(i-k,k+1)/gk(i-k,1)*gf(i-k)
20    gf(i)=gf(i)-sum
      dn(n)=gf(n)/gk(n,1)
      do 40 k=2,n
        i=np1-k
        j1=i+1
        j2=min0(n,i+ib-1)
        sum=0.0
        do 30 j=j1,j2
          mm=j-j1+2
30      sum=sum+dn(j)*gk(i,mm)
40      dn(i)=(gf(i)-sum)/gk(i,1)
        return
        end
      subroutine set3(nn,ne,np)
      if(nn.lt.2.or.nn.gt.1200) go to 1000
      if(ne.lt.1.or.ne.gt.1000) go to 1000
      if(np.lt.0.or.np.gt.10) go to 1000
      return
1000  write(6,100)
100   format(/2x,#Out of dimensional range in set3.f#,)
      stop
      end
      subroutine shape(x,n,psi,dpsi)
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                              c
```

```
c    This subroutine is to calculate the values of the shape    c
c    functions and their derivatives with respect to the        c
c    master element coordinate at the spcified point x          c
c    input parameters x,n                                       c
c    output parameters psi dpsi                                 c
c    4-, 8-, and 9-node quadrilateral element are considered    c
c    n.......the number of nodes (and shape functions) in the   c
c            element                                            c
c    x(1),x(2)...coordinates of point in the master element     c
c                coordinate system                              c
c    psi.....shape functions                                    c
c    dpsi....derivatives of shape functions                     c
c                                                               c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      dimension x(2),psi(9),dpsi(9,2)
      if(n.eq.4) then
        p=0.250*(1.0-x(2))
        q=0.250*(1.0+x(2))
        r=1.0-x(1)
        s=1.0+x(1)
        psi(1)=p*r
        psi(2)=p*s
        psi(3)=q*s
        psi(4)=q*r
        dpsi(1,1)=-p
        dpsi(1,2)=-0.250*r
        dpsi(2,1)=p
        dpsi(2,2)=-0.250*s
        dpsi(3,1)=q
        dpsi(3,2)=-dpsi(2,2)
        dpsi(4,1)=-q
        dpsi(4,2)=-dpsi(1,2)
      return
      else if(n.eq.8) then
        psi(1)=0.250*(1.0-x(1))*(1.0-x(2))*(-1.0-x(1)-x(2))
        psi(2)=0.250*(1.0+x(1))*(1.0-x(2))*(-1.0+x(1)-x(2))
        psi(3)=0.250*(1.0+x(1))*(1.0+x(2))*(-1.0+x(1)+x(2))
        psi(4)=0.250*(1.0-x(1))*(1.0+x(2))*(-1.0-x(1)+x(2))
        psi(5)=0.50*(1.0-x(1)**2)*(1.0-x(2))
        psi(6)=0.50*(1.0+x(1))*(1.0-x(2)**2)
        psi(7)=0.50*(1.0-x(1)**2)*(1.0+x(2))
        psi(8)=0.50*(1.0-x(1))*(1.0-x(2)**2)
        dpsi(1,1)=0.250*(1.0-x(2))*(2.0*x(1)+x(2))
        dpsi(1,2)=0.250*(1.0-x(1))*(x(1)+2.0*x(2))
        dpsi(2,1)=0.250*(1.0-x(2))*(2.0*x(1)-x(2))
        dpsi(2,2)=0.250*(1.0+x(1))*(-x(1)+2.0*x(2))
        dpsi(3,1)=0.250*(1.0+x(2))*(2.0*x(1)+x(2))
        dpsi(3,2)=0.250*(1.0+x(1))*(x(1)+2.0*x(2))
        dpsi(4,1)=0.250*(1.0+x(2))*(2.0*x(1)-x(2))
        dpsi(4,2)=0.250*(1.0-x(1))*(-x(1)+2.0*x(2))
        dpsi(5,1)=-x(1)*(1.0-x(2))
        dpsi(5,2)=-0.50*(1.0-x(1)**2)
        dpsi(6,1)=0.50*(1.0-x(2)**2)
```

```
         dpsi(6,2)=-x(2)*(1.0+x(1))
         dpsi(7,1)=-x(1)*(1.0+x(2))
         dpsi(7,2)=-dpsi(5,2)
         dpsi(8,1)=-dpsi(6,1)
         dpsi(8,2)=-x(2)*(1.0-x(1))
      return
      else if(n.eq.9) then
         fact1=x(1)**2-x(1)
         fact2=x(2)**2-x(2)
         fact3=x(1)**2+x(1)
         fact4=x(2)**2+x(2)
         fact5=1.0-x(2)**2
         fact6=1.0-x(1)**2
         psi(1)=0.250*fact1*fact2
         psi(2)=0.250*fact3*fact2
         psi(3)=0.250*fact3*fact4
         psi(4)=0.250*fact1*fact4
         psi(5)=0.50*fact6*fact2
         psi(6)=0.50*fact3*fact5
         psi(7)=0.50*fact6*fact4
         psi(8)=0.50*fact1*fact5
         psi(9)=fact5*fact6
         dpsi(1,1)=0.250*(2.0*x(1)-1.0)*fact2
         dpsi(1,2)=0.250*fact1*(2.0*x(2)-1.0)
         dpsi(2,1)=0.250*(2.0*x(1)+1.0)*fact4
         dpsi(2,2)=0.250*fact3*(2.0*x(2)+1.0)
         dpsi(3,1)=dpsi(2,1)
         dpsi(3,2)=dpsi(2,2)
         dpsi(4,1)=0.250*(2.0*x(1)-1.0)*fact4
         dpsi(4,2)=0.250*fact1*(2.0*x(2)+1.0)
         dpsi(5,1)=-x(1)*fact2
         dpsi(5,2)=0.50*fact6*(2.0*x(2)-1.0)
         dpsi(6,1)=0.50*(2.0*x(1)+1.0)*fact5
         dpsi(6,2)=-x(2)*fact3
         dpsi(7,1)=-x(1)*fact4
         dpsi(7,2)=0.50*fact6*(2.0*x(2)+1.0)
         dpsi(8,1)=0.50*(2.0*x(1)-1.0)*fact5
         dpsi(8,2)=-x(2)*fact1
         dpsi(9,1)=-2.0*x(1)*fact5
         dpsi(9,2)=-2.0*x(2)*fact6
      return
      else
      write(6,100)
100   format(2x,#Choose the wrong no. of shape functions #,/)
      stop
      end if
      end
      subroutine solve(nnode)
      level 2, gk,gf,nz
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      if(nz.eq.3) ib=23
      if(nz.eq.1.or.nz.eq.2) ib=13
      call trib(nnode,ib)
```

```
      call rhsb(nnode,ib)
      return
      end
      subroutine solve4
      common/cmatr4/gk4(25,3),gf4(25),vnor(25)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .              ,angl(25),capa(2,25),tin(25),ratio(50)
      ib=2
      call trib4(gk4,max124,ib)
      call rhsb4(gk4,vnor,gf4,max124,ib)
      return
      end
      subroutine sov1
      level 2, gk,gf,nz
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/cbc1/ ndbc11(25),vbc11(25),neb12(20),nsde12(20),vbc12(2,20)
     .            ,npt1(10),vpt1(10)
      common/cscal1/ rlen1,rval1
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .              angl(25),capa(2,25),tin(25),ratio(50)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      nz=1
      npass=0
      error=0.0
      call prep1
10    call proc1
      call solve(nnode1)
      if(nopt3.eq.1) go to 40
      npass=npass+1
      do 20 i=1,nnode1
        diff=abs(uc(i)-u12(i))
        if(diff.gt.error) error=diff
        u12(i)=uc(i)
20    continue
      if(error.le.tolen) go to 50
      if(npass.lt. niter) go to 10
      write(6,100)
100   format(2x,#Dont Converge in sov1.f#,)
      stop
40    do 45 i=1,nnode1
45      u12(i)=uc(i)
50    if(icount/nprint*nprint.eq.icount) call post1(npass)
      return
      end
      subroutine sov2
      level 2, gk,gf,nz
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
```

```
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/ccon2/ nnode2,nelem2,npot2,nbc21,nbc22
      common/celem/ ne(1000),node(9,1000)
      common/cnode/ x12(2,1200),u12(1200)
      common/ccon1/ nnode1,nelem1,npot1,nbc11,nbc12
      common/cbc2/ ndbc21(25),vbc21(25),neb22(20),nsde22(20),vbc22(2,20)
     .            ,npt2(10),vpt2(10)
      common/cscal2/ rlen2,rval2
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      nz=2
      npass=0
      error=0.0
      call prep2
10    call proc2
      call solve(nnode2)
      if(nopt3.eq.1) go to 40
      npass=npass+1
      do 20 i=1,nnode2
        k=i+nnode1
        diff=abs(uc(i)-u12(k))
        if(diff.gt.error) error=diff
        u12(k)=uc(i)
20    continue
      if(error.le.tolen) go to 50
      if(npass.lt. niter) go to 10
      write(6,100)
100   format(2x,#Dont Converge in sov2.f#,)
      stop
40    do 45 i=1,nnode2
        k=i+nnode1
45      u12(k)=uc(i)
50    if(icount/nprint*nprint.eq.icount) call post2(npass)
      return
      end
      subroutine sov3
      level 2, gk,gf,nz
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      common/cntrl1/ nopt1,nopt2,nopt3
      common/cdoman/ xdim,ydim,prop(4,4)
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      common/ccon3/ nnode3,nelem3,npot3,nbc31,nbc32
      common/celem/ ne(1000),node(9,1000)
      common/cnode3/ x3(2,1200),u3(1200)
      common/cbc3/ ndbc31(50),vbc31(50),neb32(40),nsde32(40),vbc32(2,40)
     .            ,npt3(10),vpt3(10)
      common/cmatrx/ gk(1200,25),gf(1200),nz
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2),
     .               angl(25),capa(2,25),tin(25),ratio(50)
```

```
      common/cwork/ xc(2,1200),nodec(9,1000),uc(1200)
      nz=3
      npass=0
      error=0.0
      call prep3
10    call proc3
      call solve(nnode3)
      if(nopt3.eq.1) go to 40
      npass=npass+1
      do 20 i=1,nnode3
        diff=abs(uc(i)-u3(i))
        if(diff.gt.error) error=diff
        u3(i)=uc(i)
20    continue
      if(error.le.tolen) go to 50
      if(npass.lt. niter) go to 10
      write(6,100)
100   format(2x,#Dont Converge in sov3.f#,)
      stop
40    do 45 i=1,nnode3
45      u3(i)=uc(i)
50    if(icount/nprint*nprint.eq.icount) call post3(npass)
      return
      end
      subroutine trib(n,ib)
      level 2, gk,gf,nz
      common/cmatrx/ gk(1200,25),gf(1200),nz
      do 20 i=2,n
        m1=min0(ib-1,n-i+1)
        do 20 j=1,m1
          sum=0.0
          k1=min0(i-1,ib-j)
          do 10 k=1,k1
10          sum=sum+gk(i-k,k+1)*gk(i-k,j+k)/gk(i-k,1)
20    gk(i,j)=gk(i,j)-sum
      return
      end
      subroutine trib4(gk,n,ib)
      dimension gk(25,3)
      do 20 i=2,n
        m1=min0(ib-1,n-i+1)
        do 20 j=1,m1
          sum=0.0
          k1=min0(i-1,ib-j)
          do 10 k=1,k1
10          sum=sum+gk(i-k,k+1)*gk(i-k,j+k)/gk(i-k,1)
20    gk(i,j)=gk(i,j)-sum
      return
      end
      subroutine twoint(m)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                             c
c    This subroutine is to generate the integration points    c
c    and weights for Gaussian Quadrature integration with     c
```

```
c     either four-point or nine-point or sixteen-point for       c
c     the square element                                         c
c     m=4.....four-point                                         c
c     m=9.....nine-point                                         c
c     m=16....sixteen-point                                      c
c                                                                c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
      common/cint/ xint1(4),wint1(4),xint2(16,2),wint2(16),none,ntwo
      if(m.eq.4) then
        do 5 i=1,4
5         wint2(i)=1.0
        a=-1.0/sqrt(3.0)
        xint2(1,1)=a
        xint2(1,2)=a
        xint2(2,1)=-a
        xint2(2,2)=a
        xint2(3,1)=a
        xint2(3,2)=-a
        xint2(4,1)=-a
        xint2(4,2)=-a
        return
      else if(m.eq.9) then
        wint2(1)=25.0/81.0
        wint2(2)=40.0/81.0
        wint2(3)=wint2(1)
        wint2(4)=wint2(2)
        wint2(5)=64.0/81.0
        wint2(6)=wint2(2)
        wint2(7)=wint2(1)
        wint2(8)=wint2(2)
        wint2(9)=wint2(1)
        aa=-sqrt(3.0/5.0)
        do 10 i=1,7,3
10        xint2(i,1)=aa
        do 20 i=2,8,3
20        xint2(i,1)=0.0
        do 30 i=3,9,3
30        xint2(i,1)=-aa
        do 40 i=1,3
40        xint2(i,2)=aa
        do 50 i=4,6
50        xint2(i,2)=0.0
        do 60 i=7,9
60        xint2(i,2)=-aa
        return
      else if (m.eq.16) then
        a1=0.347854845137454
        a2=0.652145154862546d0
        wint2(1)=a1*a1
        wint2(2)=a1*a2
        wint2(3)=wint2(2)
        wint2(4)=wint2(1)
        wint2(5)=wint2(2)
```

```
          wint2(6)=a2*a2
          wint2(7)=wint2(6)
          wint2(8)=wint2(2)
          wint2(9)=wint2(2)
          wint2(10)=wint2(6)
          wint2(11)=wint2(6)
          wint2(12)=wint2(2)
          wint2(13)=wint2(1)
          wint2(14)=wint2(2)
          wint2(15)=wint2(2)
          wint2(16)=wint2(1)
          a=-0.861136311594053
          b=-0.339981043584856
          do 110 i=1,13,4
110         xint2(i,1)=a
          do 120 i=2,14,4
120         xint2(i,1)=b
          do 130 i=3,15,4
130         xint2(i,1)=-b
          do 140 i=4,16,4
140         xint2(i,1)=-a
          do 150 i=1,4
150         xint2(i,2)=a
          do 160 i=5,8
160         xint2(i,2)=b
          do 170 i=9,12
170         xint2(i,2)=-b
          do 180 i=13,16
180         xint2(i,2)=-a
          return
          else
          write(6,100)
100       format(2x,#Choose the improper value in twoint.f#,)
          stop
          end if
          end
      subroutine velo4(xim)
c.... To obtain the normal nodal velocity along interface
c.... To obtain the nodal coordinates along interface
      dimension xim(2,25)
      common/cinter/ xi3(2,50),nmax3,xi124(2,25),max124,nfirst(2)
     .               ,angl(25),capa(2,25),tin(25),ratio(50)
      common/cmatr4/ gk4(25,3),gf4(25),vnor(25)
      common/cntrl2/ time,tmax,delt,thet,nprint,niter,tolen,icount,nwrt
      xim(1,1)=xi124(1,1)
      xim(2,1)=xi124(2,1)+vnor(1)
      xim(1,max124)=xi124(1,max124)
      xim(2,max124)=xi124(2,max124)+vnor(max124)
      do 10 i=2,max124-1
        thita=angl(i)
        xim(1,i)=xi124(1,i)+vnor(i)*cos(thita)
        xim(2,i)=xi124(2,i)+vnor(i)*sin(thita)
10    continue
      do 20 i=1,max124
```

```
        vnor(i)=vnor(i)/delt
20    continue
      ibefore=icount-1
      if(ibefore/nwrt*nwrt.eq.ibefore) then
        tbefore=ibefore*delt
        write(6,300) tbefore
300     format(//2x,#THE INTERFACIAL NODE LOCATIONS, NORMAL VELOCITIES#,
     .  #, MOVING DIRECTIONS, AND TEMPERATURES #,
     .  /2x,#AT TIME #,e15.5,#    ARE#,)
        write(6,1000)
1000    format(//3x,#NO#,13x,#X#,18x,#Y#,18x,
     .         #VN#,16x,#ANGL#,15x,#TEMP#,)
        do 30 i=1,max124
        write(6,100) i,xi124(1,i),xi124(2,i),vnor(i),angl(i),tin(i)
30      continue
100     format(/i5,5e19.5)
      end if
      return
      end
```

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720