To the Graduate Council:

I am submitting herewith a thesis written by Michael J. Gilbert entitled "A Two-Phase Enthalpy Method." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science with a major in Mathematics.

Vasilios Alexiades, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor
and Dean of The Graduate School

## STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Michael J. Gilbert

Date 5-1-92

# A TWO-PHASE ENTHALPY METHOD

A Thesis

Presented for the

Master of Science Degree

The University of Tennessee, Knoxville

Michael J. Gilbert

May 1992

## DEDICATION

This thesis is dedicated to Dr. Bernard A. Kliefoth III and Dr. Stephen R. Krauss. Without the excellent medical skills of these doctors and The University of Tennessee medical community, this thesis would not have been possible.

## ACKNOWLEDGEMENTS

I thank Dr. Vasilios Alexiades for suggesting the topic, a two-phase enthalpy method, and his guidance in this effort. I also thank Michael J. Stevens, the Sun operating system manager for the mathematics department, and William J. Mckeehan, the department's technical writer, for their helpful suggestions in compiling this thesis.

## ABSTRACT

A "compact" finite difference scheme to solve two-phase Stefan problems in one-dimension is described. This type of scheme was proposed by Milton E. Rose for the one-phase problem. Integration over control volumes and backward and forward differences are used to develop an enthalpy scheme that converges to the weak solution of the two-phase Stefan problem. Numerical results are compared to the exact Neumann (similarity) solution and to numerical solutions from an explicit and an SOR enthalpy scheme.

# TABLE OF CONTENTS


| CHAPTER | | PAGE |
|---|---|---|
| 1. | INTRODUCTION | 1 |
| 2. | THE ENTHALPY METHOD | 4 |
| | 2.1 Enthalpy formulation | 4 |
| | 2.2 The classical Stefan problem | 8 |
| 3. | A COMPACT ENTHALPY SCHEME | 12 |
| 4. | CONVERGENCE OF THE SCHEME | 17 |
| | 4.1 Weak formulation | 17 |
| | 4.2 Convergence of the compact scheme | 21 |
| | 4.3 Numerical convergence | 27 |
| 5. | NUMERICAL EXPERIMENTS | 29 |
| | 5.1 Slow melting case | 30 |
| | 5.2 Moderately fast melting case | 31 |
| | 5.3 Fast melting case | 32 |
| | 5.4 Accuracy question | 33 |
| | 5.5 Unequal Thermophysical properties | 36 |
| 6. | CONCLUSION | 37 |

**BIBLIOGRAPHY** . . . . . **39**

**APPENDIX** . . . . . **41**

A. Tables . . . . . 42

B. Figures . . . . . 47

C. Source Code . . . . . 59

**VITA** . . . . . **77**

# LIST OF TABLES


| TABLE | | PAGE |
|---|---|---|
| 1 | Numerical convergence of the compact scheme. . . . . | 42 |
| 2 | Comparisons of schemes, $St_L = St_S = 0.1$, $M = 40$ . . . . . | 43 |
| 3 | Comparisons of schemes, $St_L = St_S = 0.1$, $M = 80$ . . . . . | 44 |
| 4 | Comparisons of schemes, $M = 80$, $St_L = St_S = 1.0$ . . . . . | 45 |
| 5 | Comparisons of schemes, $M = 80$, $St_L = St_S = 5.0$ . . . . . | 46 |

# LIST OF FIGURES


| FIGURE | | PAGE |
|---|---|---|
| 1 | Slow melting case. | 47 |
| 2 | Temperature histories when $M = 40$. | 48 |
| 3 | Temperature histories when $M = 80$. | 49 |
| 4 | Front position when $St_S = St_L = 0.1$. | 50 |
| 5 | Moderately fast melting case. | 51 |
| 6 | Front position when $St_S = St_L = 1.0$. | 52 |
| 7 | Fast melting case. | 53 |
| 8 | Front position when $St_S = St_L = 5.0$. | 54 |
| 9 | Melting case when $St_L = 1.0, St_S = 0.1, c_L = 10$. | 55 |
| 10 | Front position when $St_L = 1.0, St_S = 0.1, c_L = 10$. | 56 |
| 11 | Melting case when $St_L = St_S = 0.1, \nu = 2$. | 57 |
| 12 | Front position when $St_L = St_S = 0.1, \nu = 2$. | 58 |

# CHAPTER 1

# INTRODUCTION

The two-phase Stefan problem describes the melting process for a slab of solid material at an initial temperature below the melt temperature when only conduction heat transfer is taken into account. The temperature distribution in the solid and in the liquid, as well as the location of the interface between solid and liquid as it evolves in time is sought. The problem is *explicitly* solvable when it is assumed that the slab is of semi-infinite length, and a constant temperature is imposed at the left face. The semi-infinite case is the classical two-phase Stefan problem. It is a reasonable approximation to that of the finite interval case, for short times [1]. The melting process is modeled mathematically by the heat conduction equation, but the presence of the unknown interface renders the problem nonlinear. In a weak formulation, the energy conservation law is imposed globally, irrespectively of phase, in terms of enthalpy and temperature. The phase of the material at a point in the slab depends on the value of the enthalpy at that point. The energy conservation equation updates the enthalpy, from which we can find the phase and the temperature via the equation of state of the material. The enthalpy scheme results when the slab is discretized into a finite number of control volumes and the discretized energy equation is used to find the enthalpy

within each volume irrespective of phase. The great advantage of such a scheme is that explicit tracking of the interface is avoided.

Milton E. Rose developed an implicit enthalpy scheme for one-phase Stefan problems. This scheme was introduced in three articles submitted to *The Journal of Scientific Computing* in 1989, 1990, and 1991 [4],[5],[6].

This thesis uses Rose's [4] "compact" finite difference scheme to develop an enthalpy scheme for two-phase Stefan problems in one space dimension. We find that the method works for the two-phase problem as well as it does for the one-phase problem. The energy conservation equation is discretized by integration over each control volume. Using forward and backward finite differences for the flux, we obtain a nonlinear tridiagonal system to be solved at each time step by iteration. We introduce a discrete tridiagonal operator that can be solved on a fixed solution domain with relatively few iterations. The nonlinear tridiagonal system is solved by a two-step method. The system is first solved implicitly at the first half time-step. The system is then solved explicitly at the second half time-step. The overall effect is that of an implicit scheme. The numerical results indicate that the position of the melting front can be determined with first order accuracy. The implicit compact scheme appears to have no stability restriction. However, for temperature outputs to be accurate the front should move no more than one cell per time step. There is a jump discontinuity involved when the front passes through the cell and latent heat is absorbed as solid material becomes

liquid. The accuracy restriction is found to be an improvement over the stability restriction, but it is dependent on physical properties of the phase change material.

We first review the enthalpy method. Next we construct a compact finite difference enthalpy scheme for solving a two-phase Stefan problem. Convergence of the scheme is shown. Then we compare the compact scheme with the exact Neumann (similarity) solution for the classical two-phase Stefan problem. Finally, we compare the performance of the compact scheme against an explicit scheme and an implicit SOR scheme which are described in [1]. Efficiency of the methods with respect to computer processing unit (CPU) time is also be discussed.

# CHAPTER 2

# THE ENTHALPY METHOD

## 2.1 Enthalpy formulation

The thermodynamic quantity *enthalpy* is defined to be the heat absorbed by a material under constant pressure. The enthalpy (thermal energy) per unit mass is denoted by $E$. The enthalpy represents the total energy for a pure material heated under constant pressure when there are no volume changes [1].

The heat accompanying a rise in temperature is referred to as *sensible heat.* The *latent heat* of phase change, denoted $L$, is released at the *melt temperature*, $T_m$. Heat flux, $q$, is a vector pointing in the direction of heat flow that represents the amount of energy entering or leaving a unit cross sectional area in unit time.

The First Law of Thermodynamics states the fact that

$$heat\ increase = heat\ in\ -\ heat\ out.$$

From this fundamental law of heat transfer, we get the Energy Conservation Law

$$(\rho E)_t + \vec{\nabla}\vec{q} = 0$$

where $\rho$ is the density of the material. In one dimension, denoting $\rho E$ as the

enthalpy per unit volume by $e$, we have

$$e_t + q_x = 0. \tag{2.1}$$

When heat transfer is governed by conduction only (convective and radiative modes assumed negligible), heat flux is given by Fourier's Law

$$q = -kT_x, \tag{2.2}$$

in which $k$ is the thermal conductivity of the material [1].

For moderate temperature variations, the thermophysical properties of the material may be considered constants. We will assume that $\rho, c_S, c_L, k_S, k_L$ are positive constants, where $c_S, c_L$ are the specific heats, and $k_S, k_L$ are the conductivities of the solid and liquid respectively.

Taking liquid at the melt temperature, $T_m$, as the reference state of zero energy, the enthalpy (per unit volume) of the material is related to the temperature by the following equation of state:

$$e(x,t) = \begin{cases} \rho c_S(T(x,t) - T_m) - \rho L, & T(x,t) \leq T_m \quad \text{(solid)} \\ \\ \rho c_L(T(x,t) - T_m), & T(x,t) \geq T_m \quad \text{(liquid)} \end{cases} \tag{2.3a}$$

At the melt temperature, $T = T_m$, the material undergoes the change of phase. Its energy increases from $-\rho L$ to 0 as it absorbs heat. The material may be thought of as a mixture of solid and liquid, referred to as "mushy". The solid fraction present is given by $\sigma = \dfrac{e}{-\rho L}$ where $0 \leq \sigma \leq 1$. Solving (2.3a) for $T$ in terms of

$e$, we can express the equation of state as

$$T(e) = T_m + \begin{cases} (e+\rho L)/c_S & e \le -\rho L \quad \text{(solid)} \\ 0 & -\rho L < e < 0 \quad \text{(mushy)} \\ e/\rho c_L & e \ge 0 \quad \text{(liquid)} \end{cases} \tag{2.3b}$$

The idea of the enthalpy approach is direct and physical. The volume occupied by the phase change material is partitioned into a finite number of control volumes. Energy conservation, (2.1), is applied to each control volume to obtain a discrete heat balance. The scheme updates the enthalpy of each control volume at each time-step. Using the equation of state, we update the phase (i.e. $\sigma$ ) and the temperature and proceed to the next time step. We can locate the front by observing the mushy cell at each time step. The location of the interface within the mushy cell may be found easily from the solid fraction of the cell at each time-step, since $\sigma\Delta x$ is the solid portion of the cell of width $\Delta x$.

The simplest way of deriving a discrete heat balance is a time-explicit discretation, namely a forward Euler discretation of the time derivative $e_t$ in (2.1). The disadvantage of the *explicit enthalpy scheme* is that the time-step size is restricted by the Courant-Friederichs-Lewy (CFL) condition for stability [1]:

$$\frac{k\Delta t}{\rho c \Delta x^2} \le \frac{1}{2}$$

This forces the time step to be small. Thus, the number of time steps required to reach a desired time becomes large.

To overcome the time-step restriction, one may use a backward Euler disc-

retation of $e_t$. The result is an unconditionally stable *implicit enthalpy scheme.* Then a system of nonlinear equations must be solved [1], which may be achieved by a Gauss-Seidel or an SOR iteration. Both explicit and implicit schemes are described in detail in [1].

As an alternative, Milton Rose proposed a "compact" enthalpy scheme, [4], [5], [6], to overcome the time-step restriction and to achieve second order discretization in space and time. The time derivative $e_t$ uses a backward discretization for every half time-step and a forward one for the other half. The result appears to be an unconditionally stable scheme. Then a tridiagonal system of nonlinear equations must be solved. Rose [4] constructed a finite volume scheme which was consistent with (2.1) and also satisfied a discrete version of energy conservation across the interface. As a result the scheme could be shown to converge.

The model developed in Chapter 3 can be used to solve the physical problem of a slab $0 \leq x \leq l$ of material with initial enthalpy $e_{init}(x) \leq -L$ and temperature $T_{init}(x) \leq T_m$ that is melted by constant temperature $T_0 > T_m$ applied to the left face at $x = 0$. The interface between solid and liquid phase, or melt front, $x = X(t)$, will be determined. At the melt front there is a jump discontinuity. Therefore, the solution obtained is a weak solution that is piecewise smooth. The precise sense of the weak formulation will be presented in Chapter 4.

## 2.2 The classical Stefan problem

In the classical formulation of the Stefan problem, energy conservation is imposed separately inside each phase. Then, conservation of energy across the interface must be imposed explicitly as an additional interface condition, the so called Stefan condition. We describe this formulation and then present the well-known exact solution for the semi-infinite slab case.

Consider a slab of solid material, $0 \leq x \leq l$, of constant cross-sectional area $A$. Heat is input at $x = 0$. This creates a liquid phase in $0 \leq x < X(t)$ and solid phase in $X(t) < x \leq l$, with $x = X(t)$ the location of the melt front at time $t$. Within each phase, the energy conservation law (2.1) reduces to the standard heat equation

$$\begin{aligned} T_t &= \alpha_L T_{xx}, \qquad 0 < x < X(t), \qquad t > 0, \\ T_t &= \alpha_S T_{xx} \qquad X(t) \leq x \leq l, \qquad t > 0 \end{aligned} \tag{2.4}$$

where $\alpha_i = k_i/\rho c_i$ denotes the thermal diffusivity of phase $i = L, S$. At the interface, the temperature is the melt temperature:

$$T(X(t), t) = T_m \tag{2.5}$$

We shall derive the Stefan condition from global energy conservation. The total enthalpy in the slab at time $t > 0$, according to (2.3a) is

$$E(t) = A\Big\{\int_0^{X(t)} \rho c_L [T(x,t) - T_m]\, dx + \int_{X(t)}^{l} \{\rho c_S [T(x,t) - T_m] - \rho L\}\, dx\Big\}.$$

Global heat balance demands

$$\frac{dE}{dt} = \text{net heat flow into the slab} = A\{q(0,t) - q(l,t)\}$$

where $q(0,t)$ is the flux into the slab at $x = 0$, and $q(l,t)$ is the flux at $x = l$.

Thus,

$$\frac{dE}{dt} = A\{-k_L T_x(0,t) + k_S T_x(l,t)\}. \tag{2.6}$$

We compute $\frac{dE}{dt}$:

$$\frac{1}{A}\frac{dE(t)}{dt} = \frac{d}{dt}\Big\{\int_0^{X(t)} \rho c_L[T(x,t) - T_m]\,dx + \int_{X(t)}^{l} \{\rho c_S[T(x,t) - T_m] - \rho L\}\,dx\Big\}.$$

By using Leibniz' Rule, we get

$$\frac{1}{A}\frac{dE(t)}{dt} = \rho c_L[T(X(t),t) - T_m]X'(t) + \rho L X'(t) + \int_0^{X(t)} \rho c_L T_t(x,t)dx$$

$$-\rho c_S[T(X(t),t) - T_m]X'(t) + \int_{X(t)}^{l} \rho c_S T_t(x,t)dx.$$

Then equations in (2.4) along with substitutions for $\alpha_L$ and $\alpha_S$, leave us with

$$\frac{1}{A}\frac{dE(t)}{dt} = \rho c_L[T(X(t),t) - T_m]X'(t) + \rho L X'(t) + \int_0^{X(t)} k_L T_{xx}(x,t)dx$$

$$-\rho c_S[T(X(t),t) - T_m]X'(t) + \int_{X(t)}^{l} k_S T_{xx}(x,t)dx.$$

Noting that at $T(X(t),t) = T_m$ and integrating the $T_{xx}$ terms gives

$$\frac{1}{A}\frac{dE(t)}{dt} = \rho L X'(t) + k_L T_x(X(t)^-,t) - k_L T_x(0,t) + k_S T_x(l,t) - k_S T_x(X(t)^+,t)$$

Thus, from (2.6), we obtain the *Stefan condition*:

$$\rho L X'(t) = k_S T_x(X(t)^+,t) - k_L T_x(X(t)^-,t). \tag{2.7a}$$

The Stefan condition says that the rate of change in latent heat, $\rho L X'(t)$, equals the amount by which the heat flux jumps across the interface, $[q]_-^+$. Since $\rho L = [e]_-^+$ is equivalent to the jump in enthalpy, (2.7a) can be rewritten as

$$[e]_-^+ X'(t) = [q]_-^+. \tag{2.7b}$$

In particular, the heat flux can be continuous across the interface if and only if either $L = 0$ or the interface does not move.

Imposing the temperature at $x = 0$ and at $x = l$, we are led to the following *Classical two-phase Stefan problem* in a finite slab: Find $T(x,t)$ and $X(t)$ such that:

$$\begin{aligned} T_t &= \alpha_L T_{xx}, \qquad 0 < x < X(t), \qquad t > 0, \\ T_t &= \alpha_S T_{xx} \qquad X(t) \le x \le l, \qquad t > 0 \end{aligned} \tag{2.8}$$

$$T(X(t),t) = T_m \tag{2.9}$$

$$\rho L X'(t) = k_S T_x(X(t)^+,t) - k_L T_x(X(t)^-,t). \tag{2.10}$$

$$X(0) = 0, \qquad T(x,0) = T_{init}(x), \qquad 0 \le x \le l \tag{2.11}$$

$$T(0,t) = T_0(t), \qquad T(l,t) = T_l(t), \qquad t > 0 \tag{2.12}$$

Due to the nonlinearity apparent in (2.7), no exact analytical solutions are known, even for constant initial and boundary data.

For a semi-infinite slab $0 \le x \le \infty$, a similarity transformation [1] leads to an exact solution known as

**the Neumann solution of the 2-phase Stefan problem:**

*Interface location,*

$$X(t) = 2\lambda\sqrt{\alpha_L t}, \qquad t > 0 \tag{2.13}$$

*Temperature in the liquid region,* $\quad 0 < x < X(t), \quad t > 0$

$$T(x,t) = T_L - (T_L - T_m)\ \mathrm{erf}(x/2\sqrt{\alpha_L t})/\ \mathrm{erf}(\lambda), \tag{2.14}$$

*Temperature in the solid region,* $\quad X(t) < x < \infty, \quad t > 0$

$$T(x,t) = T_{init} + (T_m - T_{init})\ \mathrm{erfc}(x/2\sqrt{\alpha_S t})/\ \mathrm{erfc}(\lambda\sqrt{\alpha_L/\alpha_S}), \tag{2.15}$$

where $\lambda$ is the unique solution to the trancendental equation

$$\frac{St_L}{\exp(\lambda^2)\ \mathrm{erf}(\lambda)} - \frac{St_S}{\nu \exp(\nu^2\lambda^2)\ \mathrm{erfc}(\nu\lambda)} = \lambda\sqrt{\pi}, \tag{2.16}$$

with

$$St_L = \frac{c_L(T_L - T_m)}{L}, \qquad St_S = \frac{c_S(T_m - T_S)}{L}, \qquad \nu = \sqrt{\alpha_L/\alpha_S} \tag{2.17}$$

Here $\mathrm{erf}(x)$ and $\mathrm{erfc}(x)$ denote the error function and the complementary error function, respectively. Forcing the temperature at $x = l$ to be the Neumann temperature gives boundary conditions for the right face used in the compact finite difference scheme.

# CHAPTER 3

# A COMPACT ENTHALPY SCHEME

Consider the two-dimensional diffusion equation

$$u_t = u_{xx} + u_{yy} \tag{3.1}$$

and let $F, G$ denote standard centered finite difference approximations to $u_{xx}$ and $u_{yy}$ respectively, involving spatial parameters $\Delta x, \Delta y$. Using standard finite difference notations, an ADI solution technique to solve (3.1) is

$$\frac{u^{n+1/2} - u^n}{\tau} = F^{n+1/2} + G^n \tag{3.2a}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\tau} = F^{n+1/2} + G^{n+1} \tag{3.2b}$$

where $\tau = \Delta t/2$. This scheme, which is formally second order accurate in the time parameter $\Delta t$ and in the spatial parameters $\Delta x, \Delta y$, is unconditionally stable since both (3.2a) and (3.2b) are implicit. The effectiveness of the scheme is found in the fact that each step involves the solution of a tridiagonal system whose solution can be obtained efficiently.

In one dimension this scheme reduces to the form

$$\frac{u^{n+1/2} - u^n}{\tau} = F^{n+1/2} \tag{3.3a}$$

$$\frac{u^{n+1} - u^{n+1/2}}{\tau} = F^{n+1/2} \tag{3.3b}$$

so that in this case only the first half-step (3.3a) requires the solution of the tridiagonal system to be found implicitly. The second half-step is found explicitly. The overall scheme turns out to be implicit and it is not bound by the CFL limitation, and it is formally of second order accuracy in both space and time.

To develop the compact scheme we partition the length of phase change material with a fixed choice of disjoint subintervals. We propose to describe the physics within each subinterval by solving (2.1) in isolation from its neighboring intervals (compactly) and then, separately, to extend the solution in the global sense by continuity conditions for $q, T$ across the boundaries of adjoining subintervals [5].

In order to implement this scheme for the one-dimensional problem, we consider a slab of material of length $l$ divided into $M$ intervals

$$I_i = x_i - \Delta x \leq x \leq x_i + \Delta x \quad \text{for } i = 1, 3, ..., 2M-1$$

as suggested by the stencil

$$| \quad \dot{x}_{i-2} \quad | \quad \dot{x}_i \quad | \quad \dot{x}_{i+2} \quad |$$

where the nodes of the control volume are indicated by $\cdot$ and the faces by $|$. First, we integrate the energy conservation law (2.1) over each interval $I_i$ and over $n\Delta t \leq t \leq (n+1/2)\Delta t$ and $(n+1/2)\Delta t \leq t \leq (n+1)\Delta t$ to obtain:

$$\frac{e_i^{n+1/2} - e_i^n}{\tau} + \frac{q_{i+1}^{(n+1/2)-} - q_{i-1}^{(n+1/2)+}}{2\Delta x} = 0 \tag{3.4a}$$

$$\frac{e_i^{n+1} - e_i^{n+1/2}}{\tau} + \frac{q_{i+1}^{(n+1/2)-} - q_{i-1}^{(n+1/2)+}}{2\Delta x} = 0 \tag{3.4b}$$

$$\text{for } i = 1, 3, ..., 2M - 1$$

involving the right and left limits $q_-^+$ of the fluxes at the endpoints of the interval $I_i$. Next, express Fourier's Law for these flux values by backward and forward difference equations

$$q_{i+1}^- = \frac{-k_{i+1}^-(T_{i+1}^- - T(e_i)}{\Delta x}$$

$$q_{i-1}^+ = \frac{-k_{i+1}^+(T(e_i) - T_{i-1}^+)}{\Delta x}$$

in which $T(e_i)$ is given by the equation of state (2.3b) and $k_{i+1}^- = k_{i-1}^+ = k_i$. Since each interval is fixed in time, the faces do not move. Therefore, the left hand side of (2.7b) is zero. Thus, (2.7b) imposes the following continuity conditions across the faces of the intervals.

$$[q_i] = [T_i] = 0, \qquad \text{for } i = 2, 4, ..., 2M - 2. \tag{3.5}$$

Thus, we have the compact finite difference scheme to solve (2.1) developed by Rose. The variables which occur are $T_-^+, q_-^+$ at the cell faces, and $e$ at the cell nodes. These can be reduced to $T = T^+ = T^-, q = q^+ = q^-$ by continuity so that $[q] = [T] = 0$ is satisfied. Therefore, the variables $T, q$ are found at the faces and $e$ is found at the nodes.

Let

$$F_i = \begin{cases} -(q_{i+1} - q_{i-1})/2\Delta x & i \text{ odd} \\ -[q_i]_-^+ & i \text{ even} \end{cases} \tag{3.6a}$$

Then

$$F_i = \begin{cases} k_{i+1}^-(T_{i+1} - T(e_i)) - k_{i-1}^+(T(e_i) - T_{i-1})/(2\Delta x)^2 & i \text{ odd} \\ k_i^+(T(e_{i+1}) - T_i) + k_i^-(T_i - T(e_{i-1}))/\Delta x & i \text{ even} \end{cases} \tag{3.6b}$$

where $k^+$ indicates conductivity of the cell to the right of the face and $k^-$ the conductivity of the cell to the left. Thus (3.5a) and (3.5b) lead to

$$F_i^{n+1/2} = \begin{cases} (e_i^{n+1/2} - e_i^n)/\tau & i = 1, 3, ..., 2M-1 \\ 0 & i = 2, 4, ..., 2M-2 \end{cases} \tag{3.7a}$$

and

$$F_i^{n+1/2} = \begin{cases} (e_i^{n+1} - e_i^{n+1/2})/\tau & i = 1, 3, ..., 2M-1 \\ 0 & i = 2, 4, ..., 2M-2 \end{cases} \tag{3.7b}$$

in which the variable $q$ has been eliminated. The solution of the tridiagonal system (3.7a) yields the values $(T_0, e_1, T_2, e_3, ..., e_{2M-1}, T_{2M})^{n+1/2}$. The odd equations of (3.7b) are solved explicitly for $e_i^{n+1}$ at the odd nodes. The nonlinearity of the system arises through the term $T(e_i)$ in $F_i$.

Rose [5] calls (3.7a) and (3.7b) the potential form of the compact scheme. It is evident that (3.6b) gives rise to a symmetric operator. Rose [4] further suggests that (3.6b) leads to a positive definite, dissipative discrete operator on the basis of energy arguments related to (2.1) multiplied by $T$ then integrated by parts to represent the total energy on the interval [5].

To solve the scheme (3.7a) for a fixed $n$, we let $U$ indicate the unknowns $(T, e)$. The symbolic form of the algebraic system is

$$A(U)U + C(U) = f^n \tag{3.8}$$

where $C(U)$ is a matrix of constants that consist of $L/c_S$ as dictated by the equation of state (2.3b). $A(U)$ and $C(U)$ are updated for each iteration. We can solve (3.8) by the iteration scheme: $U^{(0)} = U^n$, let $U^{k+1}$ solve

$$A(U^k)U^{k+1} = f^n - C(U^k), \qquad k = 0, 1, 2, ... \tag{3.9}$$

If convergence occurs, then $U = (T^{n+1/2}, e^{n+1/2})$. We have observed that an iteration error of $O(\Delta x^4)$ is sufficient for an accuracy of at most $O(\Delta x^2)$.

This model can be used to solve the physical problem mentioned in Section 2.1. The next chapter shows that the compact scheme converges to the weak solution of the problem.

# CHAPTER 4

# CONVERGENCE OF THE SCHEME

## 4.1 Weak formulation

As mentioned in Chapter 2, the enthalpy formulation of the Stefan problem must be viewed in a weak (distributional) sense, since it involves derivatives of the quantities $e$ and $q$ which experience jumps across the interface. Here we make precise the meaning of weak solution and proceed to show that the discrete solutions of the compact numerical scheme (3.7a) and (3.7b) converge to this weak solution. We interpret (2.1) in distributional sense over the region $G \equiv (0,l) \times (0,t^*), t^* > 0$. Also, set $G^- = \{(x,t) : 0 < x < X(t), 0 < t < t^*\}$ and $G^+ = \{(x,t) : X(t) < x < l, 0 < t < t^*\}$. To obtain a precise concept of weak solution of the problem, choose test function $\phi(x,t)$ in the space

$$\phi \in \Phi \equiv \{\phi \in C^2(\overline{G}) : \phi|_{t=t^*} = \phi|_{x=0} = \phi|_{x=l} = 0\},$$

multiply (2.1) by $\phi$ and integrate over $G$ to get

$$0 = \langle e_t + q_x, \phi \rangle = \iint_G \{e_t + q_x\}\phi dx dt$$

$$= \int_0^{t^*} \int_0^l \{e_t\phi - (kT_x)_x\phi\} dx dt$$

$$= \int_0^{t^*} \int_0^l \{(e\phi)_t - e\phi_t - (kT_x\phi)_x + (k[T-T_m]\phi_x)_x - k[T-T_m]\phi_{xx}\}dxdt$$

$$= -\int_0^{t^*} \int_0^l \{e\phi_t + k[T-T_m]\phi_{xx}\}dxdt$$

$$+ \int_0^{t^*} \int_0^{X(t)^-} \{(e\phi)_t + (k[T-T_m]\phi_x - kT_x\phi)_x\}dxdt$$

$$+ \int_0^{t^*} \int_{X(t)^+}^l \{(e\phi)_t + (k[T-T_m]\phi_x - kT_x\phi)_x\}dxdt$$

where $X(t)$ is the front position. Therefore, by the Divergence Theorem applied in each phase we get

$$\begin{aligned} &\int_0^{t^*} \int_0^l \{e\phi_t + k[T-T_m]\phi_{xx}\}dxdt \\ &= \iint_{\partial G^+ \cup \partial G^-} \left\{e\phi N_t + (k[T-T_m]\phi_x - kT_x\phi)\cdot \vec{N}_x\right\} dS \end{aligned} \tag{4.1}$$

with $N_t$ and $\vec{N}_x$ the time and space components of the outward unit normal to the boundaries. Integrating the first term in the right hand side of (4.1) we get

$$\iint_{\partial G^+ \cup \partial G^-} e\phi N_t dS = -\int_0^{t^*} e(X(t)^-, t)\phi X'(t)dt + \int_{X(t^*)^-}^0 e(x,t^*)\phi(x,t^*)dx$$

$$-\int_0^l e(x,0)\phi(x,0)dx + \int_l^{X(t)^+} e(x,t^*)\phi(x,t^*)dx + \int_0^{t^*} e(X(t)^+,t)\phi X'(t)dt$$

Since, $\phi \in \Phi$ implies $\phi(x,t^*) = 0$, we have

$$\iint_{\partial G^+ \cup \partial G^-} e\phi N_t dS = -\int_0^l e(x,0)\phi(x,0) - \int_0^{t^*} [e]^{\pm}_{-} X'(t)\phi dt. \tag{4.2}$$

Integrating the last term of (4.1) and using $\phi \in \Phi$ gives

$$\iint_{\partial G^+ \cup \partial G^-} (k[T - T_m]\phi_x - kT_x\phi) \cdot \vec{N}_x dS$$

$$= \int_0^{t^*} \left\{ k_L[T_m - T_m]\phi_x - k_L T_x(X(t)^-, t)\phi \right\} dt$$

$$- \int_0^{t^*} \{k_L[T(0,t) - T_m]\phi_x - k_S[T_m - T_m]\phi_x\} \, dt$$

$$- \int_0^{t^*} \left\{ k_S T_x(X(t)^+, t)\phi \right\} dt + \int_0^{t^*} k_S[T(l,t) - T_m]\phi_x(l,t) dt$$

Therefore,

$$\iint_{\partial G^+ \cup \partial G^-} (k[T - T_m]\phi_x - kT_x\phi) \cdot \vec{N}_x dS = -\int_0^{t^*} [kT_x]_S^L \phi dt$$

$$+ \int_0^{t^*} \{k_S[T_l(t) - T_m]\phi_x(l,t) - k_L[T_0(t) - T_m]\phi_x(0,t)\} \, dt \tag{4.3}$$

Combining (4.2) and (4.3) in (4.1) gives

$$\int_0^{t^*} \int_0^l \{e\phi_t + k[T - T_m]\phi_{xx}\} dx dt = -\int_0^l e_{init}(x)\phi(x,0) dx$$

$$- \int_0^{t^*} \left\{ [e]_S^L X'(t) - [q]_S^L \right\} \phi dt$$

$$+ \int_0^{t^*} \{k_S[T_l(t) - T_m]\phi_x(l,t) - k_L[T_0(t) - T_m]\phi_x(0,t)\} dt$$

The second integral on the right hand side vanishes by the Stefan condition (2.7b). This leaves

$$\int_0^{t^*} \int_0^l \{e\phi_t + k[T - T_m]\phi_{xx}\} dx dt = - \int_0^l e_{init}(x)\phi(x,0) dx$$

$$+ \int_0^{t^*} \{k_S[T_l(t) - T_m]\phi_x(l,t) - k_L[T_0(t) - T_m]\phi_x(0,t)\} dt.$$

This brings us to the following definition of a weak solution.

**Definition 4.1** *$T(x,t)$ and $e(x,t)$ constitute a weak solution of the two-phase Stefan problem if*

*(i) $e$ and $T$ are bounded (square) integrable functions in $G \equiv (0,l) \times (0,t^*)$ and $T_x \in L^2(G)$.*

*(ii) (2.3b) holds almost everywhere in $\overline{G}$,*

*(iii) the relation*

$$\int_0^{t^*} \int_0^l \{e\phi_t + [T - T_m]\phi_{xx}\} dx dt = - \int_0^l e_{init}(x)\phi(x,0) dx \tag{4.4}$$

$$+ \int_0^{t^*} \{k_S[T_l(t) - T_m]\phi_x(l,t) - k_L[T_0(t) - T_m]\phi_x(0,t)\} dt$$

*holds for every test function* $\phi \in \Phi$ *where* $e_{init}(x) \in (2.3b)$.

The well-posedness of this weak formulation of the Stefan problem has been established by Kamenomostkaya [3], Oleinek [7]. A detailed proof may be found in [1]. They have shown, in fact, that the explicit enthalpy scheme converges to the weak solution as the mesh becomes finer. The convergence of the SOR scheme has been proved by Elliot [2]. We shall show that the same is true for the compact scheme of Chapter 3.

## 4.2 Convergence of the compact scheme

Using the methods described in [1], one can derive a priori estimates on the discrete solution of the compact scheme and thus show that the discrete solutions converge weakly to some $\hat{e}, \hat{T}$ as the mesh becomes finer. Now we shall show that this weak limit $(\hat{e}, \hat{T})$ is in fact a weak solution according to Definition 4.1.

**Theorem 4.1** *The weak limit* $(\hat{e}, \hat{T})$ *of the discrete solutions of the compact scheme is a weak solution of the Stefan Problem, i.e. it satisfies (4.4).*

**proof:** Let $\phi \in \Phi$ and define $\tilde{e}, \tilde{\phi}, \tilde{\phi}_t, \tilde{\phi}_x$, and $\tilde{T}$ to be bounded sequences in $L^2(G)$ such that

$$\tilde{\phi}(x,t) = \phi(x_j, t_n), \quad \text{where} \quad j = 1, 2, ..., \quad n = 0, 1, ...,$$

$$\tilde{e}(x,t) = e_{2k-1}^{\frac{n+1}{2}}, \quad \tilde{T}(x,t) = T_{2k-1}^{\frac{n+1}{2}} \quad \text{and} \quad \tilde{\phi}_t(x,t) = \phi_t(x_{2k-1}, t_{\frac{n+1}{2}}),$$

$$\tilde{\phi}_x(x,t) = \phi_x(x_{2k-1}, t_{\frac{n+1}{2}}) \quad \text{and} \quad \tilde{\phi}_{xx}(x,t) = \phi_{xx}(x_{2k-1}, t_{\frac{n+1}{2}})$$

over

$$(2k-2)\Delta x \le x \le 2k\Delta x, \qquad k = 1, ..., M,$$

and

$$n\tau \le t \le (n+1)\tau, \qquad n = 0, 1, ..., 2N-1,$$

for $(x,t) \in \overline{G}$. By assumption, $\tilde{e}, \tilde{T}$ converges to $\hat{e}, \hat{T}$ weakly in $L^2(G)$. Take the discrete version of (2.1) found in (3.7a) and multiply it by the test function $\phi_j^n$ where each $\phi_j^n$ is an element of the sequence $\tilde{\phi}$. Summing $j$ over the odd nodes by letting $j = 2k-1$ for $k = 1, ..., M$, also summing in half time-steps over $n$ for $\tau = \frac{\Delta t}{2}$, we get,

$$\sum_{k=1}^{M} \sum_{n=0}^{2N-1} \frac{e_{2k-1}^{\frac{n+1}{2}} - e_{2k-1}^{\frac{n}{2}}}{\tau} \phi_{2k-1}^{\frac{n}{2}}$$

$$= \sum_{k=1}^{M} \sum_{n=0}^{2N-1} k_{2k-1} \frac{T_{2k-2}^{\frac{n+1}{2}} - 2T(e_{2k-1}^{\frac{n+1}{2}}) + T_{2k}^{\frac{n+1}{2}}}{2(\Delta x)^2} \phi_{2k-1}^{\frac{n}{2}} \tag{4.5}$$

We apply summation by parts on the left-hand side as follows. Using the identity

$$\frac{e_j^{n+1/2} - e_j^n}{\tau} \phi_j^n = \frac{(e\phi)_j^{n+1/2} - (e\phi)_j^n}{\tau} - e_j^{n+1/2} \frac{\phi_j^{n+1/2} - \phi_j^n}{\tau} \qquad j = 1, 3, ..., 2M-1$$

we have

$$\sum_{k=1}^{M} \sum_{n=0}^{2N-1} \frac{e_{2k-1}^{\frac{n+1}{2}} - e_{2k-1}^{\frac{n}{2}}}{\tau} \phi_{2k-1}^{\frac{n}{2}}$$

$$= \sum_{k=1}^{M} \sum_{n=0}^{2N-1} \frac{(e\phi)_{2k-1}^{\frac{n+1}{2}} - (e\phi)_{2k-1}^{\frac{n}{2}}}{\tau} - \sum_{k=1}^{M} \sum_{n=0}^{2N-1} e_{2k-1}^{\frac{n+1}{2}} \frac{\phi_{2k-1}^{\frac{n+1}{2}} - \phi_{2k-1}^{\frac{n}{2}}}{\tau}.$$

The first double sum is a telescoping series when summed over $n$. Since $\phi \in \Phi$ implies that $\phi(x_{2k-1}, t_N) = \phi(x_{2k-1}, t^*) = 0$ we are left with

$$-\frac{1}{\tau}\sum_{k=1}^{M} e(x_{2k-1},0)\phi(x_{2k-1},0) - \sum_{k=1}^{M}\sum_{n=0}^{2N-1} e_{2k-1}^{\frac{n+1}{2}} \frac{\phi_{2k-1}^{\frac{n+1}{2}} - \phi_{2k-1}^{\frac{n}{2}}}{\tau}.$$

Now, $e(x_{2k-1}, 0) = e_{init}(x_{2k-1})$ implies

$$-\frac{1}{2\tau\Delta x}\sum_{k=1}^{M}\int_{(2k-2)\Delta x}^{2k\Delta x} e_{init}(x_{2k-1})\phi(x_{2k-1},0)dx = -\frac{1}{2\tau\Delta x}\int_0^l \tilde{e}_{init}(x)\tilde{\phi}(x,0)dx$$

where

$$\tilde{e}_{init}(x) = e_{init}(x_{2k-1}) \text{ and } \tilde{\phi}(x,0) = \phi(x_{2k-1},0)$$

over

$$(2k-2)\Delta x \le x \le 2k\Delta x \qquad k = 1, ..., M.$$

Also, the definition

$$\tilde{\phi}_t(x, t+\tau) = \frac{\phi_{2k-1}^{\frac{n+1}{2}} - \phi_{2k-1}^{\frac{n}{2}}}{\tau}$$

implies

$$-\frac{1}{2\tau\Delta x}\sum_{k=1}^{M}\sum_{n=0}^{2N-1}\int_{n\tau}^{(n+1)\tau}\int_{(2k-2)\Delta x}^{2k\Delta x} \tilde{e}(x_{2k-1}, t+\tau)\tilde{\phi}_t(x_i, t+\tau)dxdt$$

$$= -\frac{1}{2\tau\Delta x}\int_0^{t^*}\int_0^l \tilde{e}(x,t)\tilde{\phi}_t(x,t)dxdt$$

Therefore, the left-hand side of (4.5) is

$$\sum_{k=1}^{M}\sum_{n=0}^{2N-1}\frac{e_{2k-1}^{\frac{n+1}{2}}-e_{2k-1}^{\frac{n}{2}}}{\tau}\phi_{2k-1}^{\frac{n}{2}} \tag{4.6}$$

$$=-\frac{1}{\Delta x\Delta t}\Big\{\int_0^{t^*}\int_0^{l}\tilde{e}(x,t)\tilde{\phi}_t(x,t)dxdt+\int_0^{l}\tilde{e}_{init}(x)\tilde{\phi}(x,0)dx\Big\}.$$

Now, consider the right-hand side of (4.5). Using the summation by parts identity (leaving out the subscripts $n+1/2$)

$$k_j\frac{T_{j-1}-2T(e_j)+T_{j+1}}{2(\Delta x)^2}\phi_j=\frac{k_j\dfrac{T_{j+1}-T(e_j)}{\Delta x}\phi_{j+1}-k_j\dfrac{T(e_j)-T_{j-1}}{\Delta x}\phi_{j-1}}{2\Delta x}$$

$$+\frac{k_j[T_{j+1}-T_m]\dfrac{\phi_{j+1}-\phi_j}{\Delta x}-k_j[T_{j-1}-T_m]\dfrac{\phi_j-\phi_{j-1}}{\Delta x}}{2\Delta x} \tag{4.7}$$

$$+k_j[T(e_j)-T_m]\frac{\phi_{j-1}-2\phi_j+\phi_{j+1}}{2(\Delta x)^2}\qquad j=1,3,...,2M-1$$

and looking at the first term summed over $j=2k-1,\quad k=1,2,...,M$ we get

$$\sum_{k=1}^{M}\frac{k_{2k-1}\dfrac{T_{2k}-T(e_{2k-1})}{\Delta x}\phi_{2k}-k_{2k-1}\dfrac{T(e_{2k-1})-T_{2k-2}}{\Delta x}\phi_{2k-2}}{2\Delta x}=$$

$$\frac{1}{2\Delta x}\Big[k_1\frac{T_2-T(e_1)}{\Delta x}\phi_2-k_1\frac{T(e_1)-T_0}{\Delta x}\phi_0+...+k_j\frac{T_{j+1}-T(e_j)}{\Delta x}\phi_{j+1}$$

$$-k_j\frac{T(e_j)-T_{j-1}}{\Delta x}\phi_{j-1}+k_{j+2}\frac{T_{j+3}-T(e_{j+2})}{\Delta x}\phi_{j+3}-k_{j+2}\frac{T(e_{j+2})-T_{j+1}}{\Delta x}\phi_{j+1}$$

$$+...+k_{2M-1}\frac{T_{2M}-T(e_{2M-1})}{\Delta x}\phi_{2M}-k_{2M-1}\frac{T(e_{2M-1})-T_{2M-2}}{\Delta x}\phi_{2M-1}\Big].$$

Regrouping terms in the following manner

$$\left[k_j\frac{T_{j+1}-T(e_j)}{\Delta x}-k_{j+2}\frac{T(e_{j+2})-T_{j+1}}{\Delta x}\right]\phi_{j+1}\quad\text{for}\quad j=1,3,...,2M-1$$

gives backward and forward difference quotients representing the discrete fluxes. But

$$[q_{j+1}]_-^+=0\quad\text{for}\quad j=1,3,...,2M-3$$

by energy conservation condition across the faces, (3.5). This leaves

$$\frac{1}{2\Delta x}\left[k_{2M-1}\frac{T_{2M}-T(e_{2M-1})}{\Delta x}\phi_{2M}-k_1\frac{T(e_1)-T_0}{\Delta x}\phi_0\right]=0$$

because $\phi\in\Phi$ implies

$$\phi_{2M}=\tilde{\phi}(x_l,t)=\tilde{\phi}(x_0,t)=\phi_0=0.$$

Next, we consider the second term of the summation by parts identity, (4.7) summed over $j=2k-1,\quad k=1,2,...,M$. We get

$$\sum_{k=1}^{M}k_{2k-1}\left[[T_{2k}-T_m]\frac{\phi_{2k}-\phi_{2k-1}}{\Delta x}-[T_{2k-2}-T_m]\frac{\phi_{2k-1}-\phi_{2k-2}}{\Delta x}\right]$$

$$=\left[k_S[\tilde{T}_l(t)-T_m]\tilde{\phi}_x(l,t)-k_L[\tilde{T}_0(t)-T_m]\tilde{\phi}_x(0,t)\right]$$

by regrouping as in the previous sum and noticing that we have backward and forward difference quotients for $\tilde{\phi}_x(x_i,t)$ where $i=2,4,...,2M-2$. That is, $\phi\in C^2(\overline{G})$ implies $\phi_x\in C(\overline{G})$, so we have

$$[T_{j+1}-T_m]\left[\frac{\phi_{j+1}-\phi_j}{\Delta x}-\frac{\phi_{j+2}-\phi_{j+1}}{\Delta x}\right]=0\quad\text{for } j=1,3,...,2M-3$$

by continuity. Therefore, the second term of (4.7) gives

$$\sum_{k=1}^{M}\sum_{n=0}^{2N-1}\int_{n\tau}^{(n+1)\tau}\int_{(2k-2)\Delta x}^{2k\Delta x} k_{2k-1}\Big\{[T_{2k}^{\frac{n+1}{2}}-T_m]\frac{\phi_{2k}^{\frac{n+1}{2}}-\phi_{2k-1}^{\frac{n+1}{2}}}{\Delta x}$$

$$-[T_{2k-2}^{\frac{n+1}{2}}-T_m]\frac{\phi_{2k-1}^{\frac{n+1}{2}}-\phi_{2k-2}^{\frac{n+1}{2}}}{\Delta x}\Big\}dxdt$$

$$=\int_0^{t^*}\Big\{\tilde{k}_S[\tilde{T}_l(t)-T_m]\tilde{\phi}_x(l,t)-\tilde{k}_L[\tilde{T}_0(t)-T_m]\tilde{\phi}_x(0,t)\Big\}dxdt$$

where

$$\begin{aligned}\tilde{T}_0(t) &= T(x_0,t_{\frac{n+1}{2}}), & \tilde{T}_l(t) &= T(x_{2M},t_{\frac{n+1}{2}})\\ \tilde{\phi}_x(0,t) &= \phi_x(x_0,t_{\frac{n+1}{2}}), & \tilde{\phi}_x(l,t) &= \phi_x(x_{2M},t_{\frac{n+1}{2}})\end{aligned}$$

for

$$n\tau\le t\le(n+1)\tau \qquad n=0,1,...,2N-1$$

Finally, the last term in the summation by parts identity, (4.7) gives

$$\sum_{k=1}^{M}\sum_{n=0}^{2N-1}\int_{n\tau}^{(n+1)\tau}\int_{(2k-2)\Delta x}^{2k\Delta x} k_{2k-1}[T(e_{2k-1}^{\frac{n+1}{2}})-T_m]\frac{\phi_{2k-2}^{\frac{n+1}{2}}-2\phi_{2k-1}^{\frac{n+1}{2}}+\phi_{2k}^{\frac{n+1}{2}}}{2(\Delta x)^2}dxdt$$

$$=-\frac{1}{2\tau\Delta x}\int_0^{t^*}\int_0^{l}[T(\tilde{e})-T_m](\tilde{\phi}_x)_x dxdt$$

Thus, (4.5) becomes

$$\int_0^{t^*}\int_0^l \left\{\tilde{e}\tilde{\phi}_t + \tilde{k}[T(\tilde{e}) - T_m](\tilde{\phi}_x)_x\right\} dxdt =$$

$$\int_0^{t^*} \{k_S[\tilde{T}_l(t) - T_m]\tilde{\phi}_x(l,t) - k_L[\tilde{T}_0(t) - T_m]\tilde{\phi}_x(0,t)\}dt \qquad (4.8)$$

$$-\int_0^l \tilde{e}_{init}(x)\tilde{\phi}(x,0)dx$$

Now, we take $M \to \infty$ through subsequence $M_i$ for which $\left\{\tilde{e}, \tilde{T}\right\} \rightharpoonup \left\{\hat{e}, \hat{T}\right\}$ and observe the convergence of each term. For example, the first term of (4.8) becomes

$$\iint_G \tilde{e}\tilde{\phi}_t dxdt \equiv \iint_G \left[\tilde{e} - \hat{e}\right]\phi_t dxdt + \iint_G \tilde{e}\left[\tilde{\phi}_t - \phi_t\right]dxdt + \iint_G \hat{e}\phi_t dxdt$$

$$\to 0 + 0 + \iint_G \hat{e}\phi_t dxdt \quad \text{as} \quad M_i \to \infty$$

as suggested in Alexiades [1]. Similarly, the other terms converge so that (4.8) $\to$ (4.4) with $\hat{e}, \hat{T}$ in place of $e, T$. Thus, $\hat{e}, \hat{T}$ is a weak solution of the two-phase Stefan problem.

### 4.3 Numerical convergence

Table 1 [1] was generated by the source code found in the Appendix C for the two-phase Stefan problem when $T_0 = 1$,$T_{init} = -1$, $e_{init} = -11$, $L = 10$, for which the trancendental root of (2.16) is $\lambda = 0.189134$. All other constants

[1]Tables are found in Appendix A.

have been normalized to 1. $M$ is the number of nodes. Max error is the absolute value of the difference between the numerical temperature and the exact Neumann temperature found at output nodes .1,.2,...,.9. This error is calculated at each time step $\Delta t = .12(\Delta x)^2$. All experiments were run until the front $X(t)$ reached $x = 0.5$. Since $\Delta x = 1/2M$, the maximum error is less than $4\Delta x$. Therefore, Table 1 indicates the error is $O(\Delta x)$. The maximum of the $L^1$ errors in temperatures at these nodes is also shown.

Table 1 shows that as $M$ increases the error decreases. Thus, we see convergence. Note that these errors are calculated at a distance from the left face $x = 0$. During the first few time steps, the instantaneous application of $T_0$ at $x = 0$ causes much greater error at points near $x = 0$, but after a few time steps this smoothes out to give reasonable results.

# CHAPTER 5

## NUMERICAL EXPERIMENTS

The data in Table 1 results from solving the problem of finding a temperature distribution $T(x,t)$ and an interface function $X(t)$ satisfying the conditions of equations (2.4) when a constant temperature $T_0 \geq T_m$ is applied to the left face of a solid slab of length $0 \leq x \leq l$ at initial temperature $T_{init}(x) \leq T_m$ for time $t > 0$. The interface temperature is $T(X(t),t) = T_m$ and the temperature of the liquid region is $T_L > T_m$. Equation (2.7a) gives the Stefan condition for this melting problem.

By forcing the temperature at the right boundary to be the Neumann temperature, we can use the equation of state, (2.3), along with equations (3.7a) and (3.7b) to obtain temperatures at output nodes that can be compared for accuracy with the Neumann solution. This is the equivalent of a physical experiment where a thermocouple is placed at a particular position in the slab in order to measure temperature change at that position during the melting process.

The Stefan number, being the ratio of sensible to latent heat, (see (2.17)), is an indicator of the speed of the melting process. The melting process is slow when $St_S = St_L = 0.1$. The melting process is fast when $St_S = St_L = 5.0$. These melting processes are presented along with the melting process when $St_S = St_L =$

1.0.

A slab of length $0 \leq x \leq l$ at initial temperature $T_{init} = -1$ is discretized into $M$ control volumes of length $\Delta x = 1/2M$. Heat, at temperature $T_0 = 1$, is applied at $x = 0$. The time step used in solving equations (3.7a) and (3.7b) is chosen to satisfy the explicit scheme CFL condition (i.e. $\Delta t \leq .3\Delta x^2$ where $\Delta x = 1/M$). The following figures are the result of modeling the different melting processes.

### 5.1 Slow melting case

Figure 1 [1] gives the temperature history of location $x = 0.3$ during the slow melting process, with $St_L = St_S = 0.1$.

$St_L = 0.1$ implies that there is a large latent heat to overcome relative to sensible heat. Therefore, the melting process is slow. The largest errors in temperature result when the numerical scheme dictates that the node is still mushy while the Neumann temperature is that of an already melted node. To compare with the Neumann temperature history, we plot them both in Figure 2. When we compare the Neumann temperature and the numerical temperature during this process at the node $x = 0.3$, the maximum difference in temperature is $2.95 \times 10^{-2}$ and of the order $2.4\Delta x$ where $\Delta x = 1/2M = 1.25 \times 10^{-2}$.

By solving the system numerically we observe step-like behavior of the numerical temperature. This behavior occurs at the mushy node during the phase

[1] Figures are found in Appendix B.

change process because at a mushy node the temperature equals zero. Figure 1 shows that this behavior first occurs when the temperature of the output node, $x = 0.3$, reaches the melt temperature. As the front moves to the control volumes further to the right, the temperature of a current mushy node is kept at the melt temperature for as long as the front lies within it. The temperature throughout the slab quickly reaches a steady-state and remains at the state until the front moves to the next cell.

At that moment the temperature readjusts toward a new steady-state and we see a new step in the figure. The width of each step is directly proportional to the time the front remains inside each control volume, and therefore proportional to the mesh size. Indeed, if we make the mesh finer we see less of the effect, as shown in Figure 3. For the same reason, the faster the melting process, the less time that the front spends in each cell, and the less pronounced this step-like behavior becomes as we see in Figure 5 and Figure 7.

The numerical and exact fronts are shown in Figure 4 The maximum front tracking error is $7.87 \times 10^{-3}$ or approximately of the order $0.63\Delta x$. So, the curves are indistinguishable at the scale of the figure.

### 5.2 Moderately fast melting case

If we repeat the experiment by letting $St_L = St_S = 1.0$ and applying heat until the front reaches .75, we get a temperature history as given in Figure 5. We

have a larger Stefan number with latent heat equivalent to the sensible heat. The melting process will be somewhat faster than when we used the smaller Stefan number. The maximum error in temperatures at the node for this experiment $2.24 \times 10^{-2}$ occurred when time $t = 0.16$, while the node was in the process of going from mushy to liquid and is of the order $1.8\Delta x$. It is precisely the point were the nonlinearity of the system that solves (3.7a) contributes the greatest effect. The step-like behavior is still present but considerably less pronounced than in Figure 1.

The front position is shown in Figure 6. The maximum front tracking error for this experiment is $1.6 \times 10^{-2}$ and is of the order $\Delta x$. The maximum error occurs during the very first time step. The first node's left face instantaneously goes from the initial temperature, $T_{init} = -1$ to the temperature $T_0 = 1$, and the Neumann front starts off with infinite speed. After this initial error, the next highest front tracking error is $3.1 \times 10^{-3}$. This is approximately of the order $0.25\Delta x$. The error smoothes out after a few time steps and has a maximum error of $1.5 \times 10^{-3}$. The front tracking error for the first experiment exhibits similar behavior.

### 5.3 Fast melting case

Next, we wish to model an experiment in which conduction dominates the phase change. In order to do this we apply the phase change process to a material with large Stefan number such that the sensible heat is larger than the latent heat.

Figure 7 shows the temperature history of such an experiment. The maximum error in temperature at this node is $7.9 \times 10^{-3}$. It occurs right before the phase change from mushy to liquid takes place and is of the order $0.64\Delta x$.

This error is not as large as the errors in the previous case, since the latent heat jump is much smaller here. The front moves much faster, the time interval during which each node is mushy is much shorter, and so the step-like nature of the history curve is hardly visible in Figure 7. Thus, the nonlinearity of the problem has less impact when the phase change actually occurs. Figure 8 gives the front position for the large Stefan number experiment.

The maximum front tracking $1.78 \times 10^{-2}$, of the order $1.4\Delta x$, occurs again at the start of the experiment. The next largest error, $8.5 \times 10^{-3}$, is of the order $0.68\Delta x$.

### 5.4 Accuracy question

Since the scheme is unconditionally stable we can increase the time-step and see how the temperature histories are affected. We have allowed the CFL condition for the explicit solution to govern the size of the time-step. Increasing the time-step size we find that the maximum difference between Neumann and numerical temperatures is $2.97 \times 10^{-2}$ at the output nodes $0.1, 0.2, ..., 0.9$ when $\Delta t = 5\Delta t_{CFL}$. The maximum front tracking error is $9.1 \times 10^{-3}$. We have gained in computing as far as execution time is concerned without a great loss in accuracy. When $\Delta t =$

$10\Delta t_{CFL}$ the maximum difference between Neumann and numerical temperatures is $5.25 \times 10^{-2}$, and the maximum front tracking error is $8 \times 10^{-3}$. So, when measuring temperature at a specific node inside the volume we see that the CFL condition can be violated without a great loss in accuracy.

Finally, we want to explore exactly how far we can increase the time-step. Another point to examine is what loss in accuracy occurs when the time-step is increased? Furthermore, how does the performance compare in terms of accuracy and CPU time with an implicit SOR scheme that solves the same problem? Do the implicit schemes perform better than the explicit scheme? Table 2 addresses these questions when $St_L = St_S = 0.1$, $L = 10, M = 40$, and the explicit time-step $\Delta t_{CFL} = 1.875 \times 10^{-4}$ is multiplied by a constant factor in order to increase the time-step size. The difference between the Neumann and numerical temperatures at output nodes $.1, .2, ..., .9$ are measured at each time step. $L^1$ errors have also been calculated along with the front tracking errors. The maximum of these errors for each experiment is recorded in the Table. These are compared to results that Alexiades [1] has obtained for the explicit and the SOR schemes. The errors incurred by the explicit scheme are displayed in row one of Table 2. All calculations were done on a SUN4/110 workstation, so the CPU times (user time) were obtained on the same machine with the same Fortran compiler. The results shown from the SOR scheme are the best found by varying the over-relaxation parameter $\omega$ (typically $1.4 \leq \omega \leq 1.7$).

It is clear that the compact scheme does not perform well when we increase $\Delta t_{CFL}$ to 20 times the value given by the CFL condition. Table 3 is the result of repeating the above experiments with $M = 80$ and $\Delta t_{CFL} = 4.68750 \times 10^{-5}$.

The compact scheme performs well compared to the SOR scheme when $\Delta t_{CFL}$ is multiplied by a factor of 40, but the CPU time is twice as long. The errors of both schemes are comparable to that of the explicit solution. While the SOR scheme continues to provide decent results when $\Delta t_{CFL}$ is multiplied by a factor of 80, the accuracy of the compact scheme degrades to a 31% error on temperature.

Table 4 shows similar results for the moderately fast melting case ($St_L = St_S = 1.0$) with $M = 80$, $\Delta t = 4.68750 \times 10^{-5}$. The compact scheme improves when the latent heat and the sensible heat are equivalent. When $\Delta t_{CFL}$ is multiplied by a factor of 40, its accuracy is comparable to the explicit method, but there is still significant cost in terms of CPU time. Table 5 results when $M = 80, St_L = St_S = 5.0, \Delta t_{CFL} = 4.68750 \times 10^{-5}$.

It appears that the compact scheme is best for the conduction dominated problem. When the latent heat is smaller than the sensible heat, the scheme allows $\Delta t$ to be larger. Multiplying $\Delta t_{CFL}$ by a factor of 40 yields reasonable accuracy at one third the cost in CPU time as the explicit scheme, but the SOR scheme yields similar results at slightly less cost. At a factor of 60 however, the front error of the SOR scheme degrades to 12% while the compact scheme retains reasonable accuracy. This was the only case found in which the compact scheme

outperforms the SOR scheme.

### 5.5 Unequal Thermophysical properties

So far, we have used equal thermophysical properties for the phases: $St_L = St_S$ and $\alpha_L = \alpha_S$ in the numerical experiments. Now we examine the performance when they are different. Figure 9 shows the case $St_L = 1.0$ and $St_S = 0.1, \lambda = 0.594510$. That is, the problem of melting a slab with $c_L = 10$ at initial temperature $T_{init} = -1$ by applying $T_0 = 1$ at the face $x = 0$. The compact scheme's results are similar to those of the SOR scheme. $M$ must be large in order to get decent results in temperature outputs at the initial application of $T_0$. After a few times steps the error is reasonable. The front position is given in Figure 10.

Another case of unequal thermophysical properties is when $St_L = St_S$, but $\alpha_L \neq \alpha_S$. The temperature history appears in Figure 11 when we use unequal conductivities $k_L = 4, k_S = 1, \lambda = 0.200574$. The front position is shown in Figure 12 .

# CHAPTER 6

# CONCLUSION

The one-dimensional, two-phase Stefan problem with Neumann solution can be treated by the enthalpy method using an unconditionally stable numerical method requiring the iterative solution of a symmetric tridiagonal system of equations (3.7a) at each time step. The number of iterations per time step is most often less than 4. This suggests that we have a computationally efficient method. Table 2 shows that when $M = 40, \Delta t$ can be multiplied by a factor of 10 and still yield good results in output error. The CPU time is decreased by 36 percent when compared to the explicit method. $\Delta t$ can be multiplied by a factor of 40 and still yield acceptable error when $M = 80$ as shown in Table 3. The CPU time is cut by one third. On the other hand, the SOR method produces better results in each of these cases. The small Stefan numbers imply larger latent heat. The nonlinearity of the system caused by the equation of state (2.3b) incorporates this jump. When the Stefan numbers are 1.0 we see an improvement in the efficiency. Furthermore, Table 5 shows that we can multiply $\Delta t$ by a factor of 60 with error output comparable to the explicit error output. This suggests that the method is most efficient for large Stefan numbers.

It is apparent that there is a CFL-like condition restricting the time-step for

accuracy rather than stability. An obvious such condition is that the front can move no more than one control volume during a time step, which means $|X'|\Delta t \leq 2\Delta x$.

In the compact scheme, the implicit step is over half a time-step, so we restrict $\Delta t$ so that $|X'|\Delta t/2 \leq 2\Delta x$. For the explicitly solvable problems we are examining, the exact front is given by $X(t) = 2\lambda\sqrt{\alpha_L t}$, so the condition is

$$\Delta t \leq \frac{16\Delta x^2}{\lambda^2 \alpha_L}. \tag{6.1}$$

For the case $St_L = St_S = 0.1, \lambda = 0.189134$ and (6.1) restricts $\Delta t \leq 7.0 \times 10^{-2} = 373\Delta t_{CFL}$. Table 2 indicates that the error has broken down at $20\Delta t = 3.75 \times 10^{-3}$. This implies that (6.1) is not restrictive enough.

For the case $St_L = St_S = 1.0, \lambda = 0.377760$ and (6.1) restricts $\Delta t \leq 4.4 \times 10^{-3} = 94\Delta t_{CFL}$. Table 3 indicates that the error has broken down at $100\Delta t_{CFL}$.

For the case $St_L = St_S = 5.0, \lambda = 0.450161$ and (6.1) restricts $\Delta t \leq 3.1 \times 10^{-3} = 66\Delta t_{CFL}$. Table 4 indicates that the error has broken down at $100\Delta t_{CFL}$. On the other hand, $60\Delta t_{CFL}$ does not violate (6.1), and the error is reasonable when compared to the explicit scheme results.

Therefore, it is not clear that limitations on the accuracy can be found using the front position and physical properties of the materials used.

# BIBLIOGRAPHY

# BIBLIOGRAPHY


[1] Vasilios Alexiades and Alan D. Solomon, *Mathematical Modeling of Melting and Freezing Processes*, Hemisphere Publishing Co., Washington, D.C., 1992.

[2] C. M. Elliot and J. R. Ockendon, *Weak and Variational Methods for Moving Boundary Problems*, Pitman Press, Boston (1982)

[3] S. L. Kamenomostkaya, "On the Stefan Problem (in Russian)", *Mat. Sbornik*, Vol. 53(95), 1961, pp. 489-514.

[4] Milton E. Rose, "Compact Finite Volume Methods for the Diffusion Equation", *Journal of Scientific Computing*, Vol.4, No. 3, Septmeber 1989, pp. 261-290.

[5] Milton E. Rose, "An Implicit Enthalpy Scheme for One-Phase Stefan Problems", v.1.7 preprint , 3/8/90.

[6] Milton E. Rose, "An Enthalpy Scheme for Stefan Problems in Several Dimensions", preprint, 12/18/90.

[7] O. Oleinik, "A method of solution of the general Stefan problem", *Soviet Math. Doklady*, Vol. 1, 1960, 1350-1354.

# APPENDIX

# APPENDIX A

## Tables

Table 1: Numerical convergence of the compact scheme.

| M | max error | max L1 error |
|---|---|---|
| 10 | 0.162905 | 8.13138E-02 |
| 20 | 8.83111E-02 | 4.26744E-02 |
| 40 | 4.89184E-02 | 2.09496E-02 |
| 80 | 2.72332E-02 | 1.16360E-02 |
| 160 | 1.39365E-02 | 5.27048E-03 |

Table 2: Comparisons of schemes, $St_L = St_S = 0.1$, $M = 40$

| Code | Front Error | Temperature Error | L1 Error | number of iterations | CPU |
|---|---|---|---|---|---|
| dtfactor=1 | | | | no. of time steps: 9321 | |
| EXP | 7.17079E-03 | 9.00053E-02 | 1.23843E-02 | | 55.2u |
| SOR | 8.21967E-03 | 8.67650E-02 | 1.36787E-02 | 39236 | 88.0u |
| ROSE | 7.86534E-03 | 4.86858E-02 | 2.04615E-02 | 18656 | 353.9u |
| dtfactor=10 | | | | no. of time steps: 933 | |
| SOR | 7.13642E-03 | 8.38407E-02 | 1.85262E-02 | 13969 | 22.4u |
| ROSE | 7.99387E-03 | 5.25302E-02 | 2.00665E-02 | 1887 | 35.3u |
| dtfactor=20 | | | | no. of time steps: 467 | |
| SOR | 7.23558E-03 | 8.38410E-02 | 2.36442E-02 | 10579 | 13.6u |
| ROSE | 2.74566E-02 | 0.115489 | 3.52641E-02 | 971 | 15.3u |
| dtfactor=40 | | | | no. of time steps: 235 | |
| SOR | 7.21842E-03 | 6.98302E-02 | 1.85981E-02 | 6787 | 7.7u |
| ROSE | 7.74220E-02 | 0.167801 | 7.38325E-02 | 546 | 11.0u |
| dtfactor=80 | | | | no. of time steps: 118 | |
| SOR | 1.52936E-02 | 0.110401 | 3.14047E-02 | 3753 | 4.4u |
| ROSE | 7.08258E-02 | 0.545349 | 9.66813E-02 | 369 | 6.7u |

Table 3: Comparisons of schemes, $St_L = St_S = 0.1$, $M = 80$

| Code | Front Error | Temperature Error | L1 Error | number of iterations | CPU |
|---|---|---|---|---|---|
| dtfactor=1 | | | | no. of time steps: 37286 | |
| EXP | 3.58540E-03 | 4.86872E-02 | 6.19215E-03 | | 391.3u |
| SOR | 4.10984E-03 | 4.89212E-02 | 6.83945E-03 | 142019 | 985.2u |
| ROSE | 3.93267E-03 | 2.66847E-02 | 9.66876E-03 | 74622 | 4360.1u |
| dtfactor=20 | | | | no. of time steps: 1865 | |
| SOR | 3.61777E-03 | 4.84602E-02 | 1.18220E-02 | 35805 | 84.0u |
| ROSE | 1.37283E-02 | 4.23717E-02 | 1.27442E-02 | 3808 | 181.9u |
| dtfactor=40 | | | | no. of time steps: 934 | |
| SOR | 3.60923E-03 | 4.84620E-02 | 9.30791E-03 | 23365 | 52.8u |
| ROSE | 3.87111E-02 | 5.49001E-02 | 2.37985E-02 | 2019 | 118.4u |
| dtfactor=80 | | | | no. of time steps: 467 | |
| SOR | 7.64644E-03 | 4.93000E-02 | 1.57500E-02 | 17123 | 92.0u |
| ROSE | 3.54129E-02 | 0.316225 | 6.84829E-02 | 1239 | 59.1u |

Table 4: Comparisons of schemes, $M = 80$, $St_L = St_S = 1.0$

| Code | Front Error | Temperature Error | L1 Error | number of iterations | CPU |
|---|---|---|---|---|---|
| dtfactor=1 | | | | no. of time steps: 9370 | |
| EXP | 4.39306E-03 | 3.50725E-02 | 3.38105E-03 | | 91.8u |
| SOR | 6.87000E-03 | 3.60000E-02 | 5.31000E-03 | 40848 | 160.6u |
| ROSE | 8.01015E-03 | 2.08850E-02 | 4.93034E-03 | 18724 | 980.2u |
| dtfactor=5 | | | | no. of time steps: 1876 | |
| SOR | 1.64000E-02 | 3.70000E-02 | 7.61000E-03 | 18397 | 50.2u |
| ROSE | 4.90981E-03 | 2.08892E-02 | 5.09413E-03 | 3776 | 197.6u |
| dtfactor=10 | | | | no. of time steps: 939 | |
| SOR | 1.64000E-02 | 3.33000E-02 | 7.79000E-03 | 13953 | 36.8u |
| ROSE | 1.63575E-02 | 1.96155E-02 | 5.05720E-03 | 964 | 50.6u |
| dtfactor=40 | | | | no. of time steps: 235 | |
| SOR | 6.54000E-02 | 5.16000E-02 | 1.41000E-02 | 7931 | 18.4u |
| ROSE | 2.77043E-02 | 3.65476E-02 | 9.05121E-03 | 546 | 38.7u |
| dtfactor=100 | | | | no. of time steps: 95 | |
| SOR | 7.32000E-02 | 7.91000E-02 | 2.14000E-02 | 4934 | 11.4u |
| ROSE | 0.112199 | 0.121769 | 3.73680E-02 | 353 | 17.8u |

Table 5: Comparisons of schemes, $M = 80$, $St_L = St_S = 5.0$

| Code | Front Error | Temperature Error | L1 Error | number of iterations | CPU |
|---|---|---|---|---|---|
| dtfactor=1 | | | | no. of time steps: 6669 | |
| EXP | 1.74347E-02 | 1.88444E-02 | 1.39182E-03 | | 66.3u |
| SOR | 1.63000E-02 | 1.95000E-02 | 4.00000E-03 | 29535 | 107.6u |
| ROSE | 8.88606E-03 | 1.25842E-0 | 2.62586E-03 | 13195 | 697.8u |
| dtfactor=20 | | | | no. of time steps: 335 | |
| SOR | 3.89851E-02 | 3.97314E-02 | 1.26709E-02 | 7567 | 17.8u |
| ROSE | 4.39548E-02 | 1.09922E-02 | 3.14599E-03 | 694 | 36.8u |
| dtfactor=40 | | | | no. of time steps: 168 | |
| SOR | 7.80000E-02 | 6.57000E-02 | 1.68000E-02 | 5694 | 14.0u |
| ROSE | 7.55941E-02 | 3.46652E-02 | 5.47880E-03 | 388 | 20.5u |
| dtfactor=60 | | | | no. of time steps: 113 | |
| SOR | 0.12600E+00 | 9.21000E-02 | 2.09000E-02 | 4804 | 11.0u |
| ROSE | 4.77468E-02 | 4.12328E-02 | 8.37058E-03 | 283 | 17.4u |
| dtfactor=100 | | | | no. of time steps: 68 | |
| SOR | 0.137833 | 0.123152 | 2.69033E-02 | 3654 | 7.9u |
| ROSE | 6.16408E-02 | 9.35826E-02 | 1.40380E-02 | 244 | 17.7u |

# APPENDIX B

## Figures

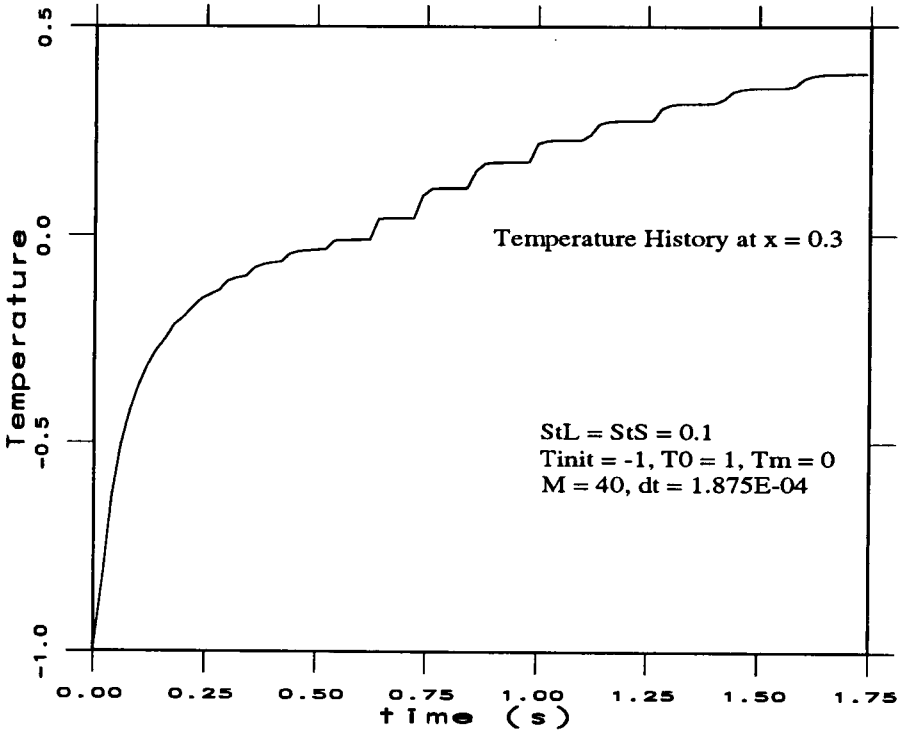


Figure 1: Slow melting case.

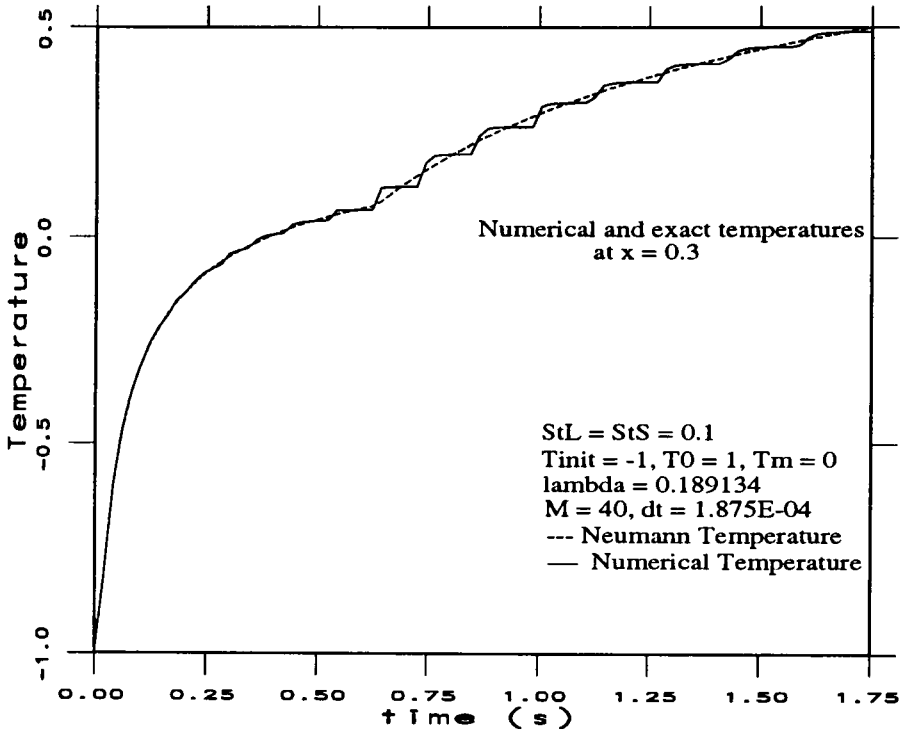


Figure 2: Temperature histories when $M = 40$.

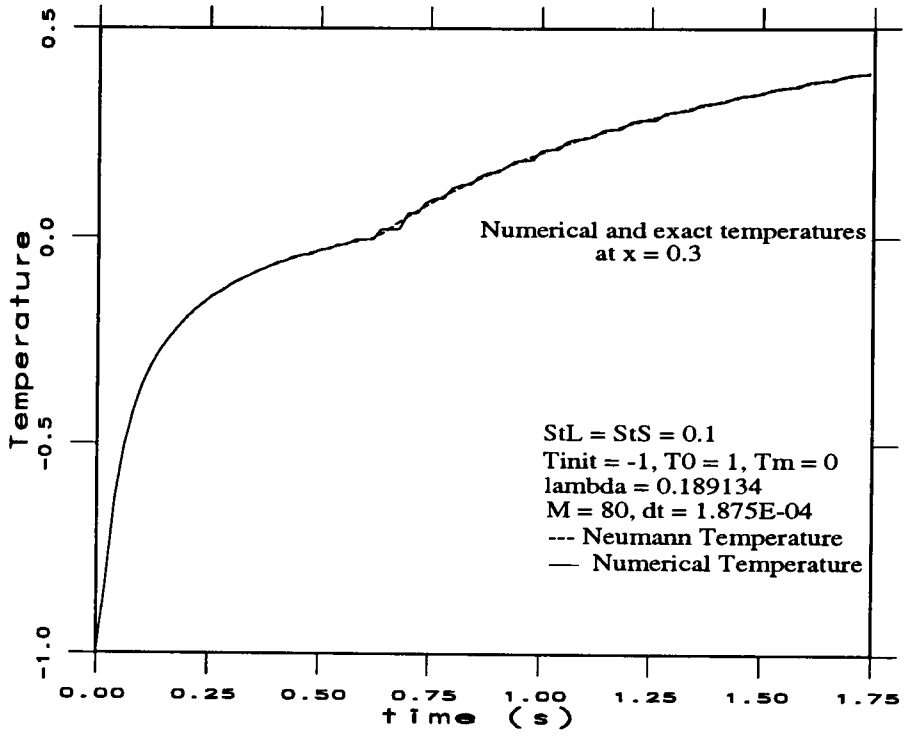


Figure 3: Temperature histories when $M = 80$.

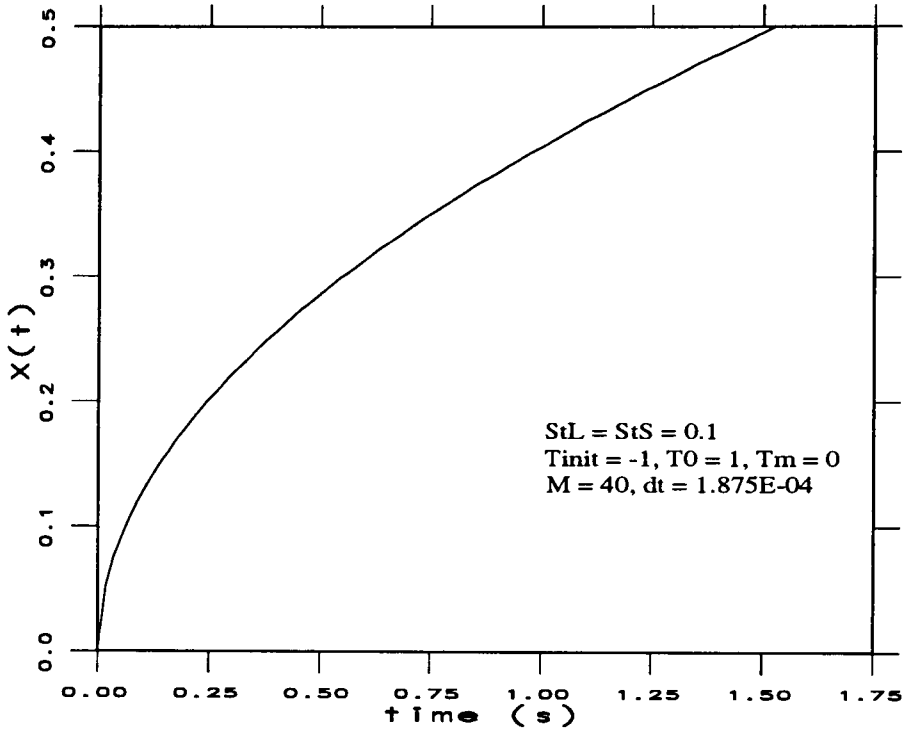


Figure 4: Front position when $St_S = St_L = 0.1$.

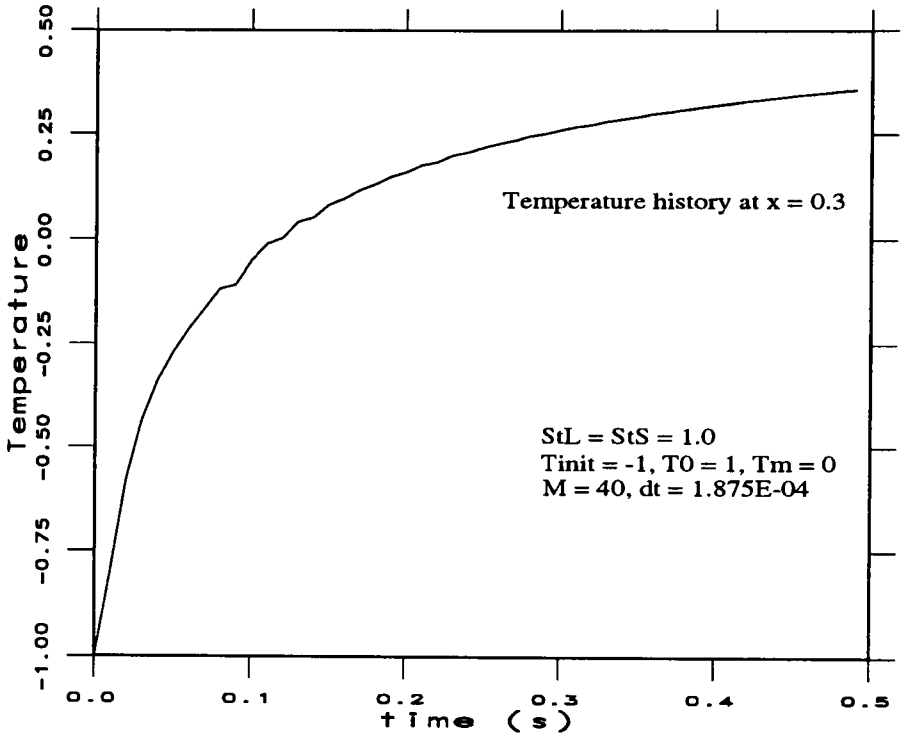


Figure 5: Moderately fast melting case.

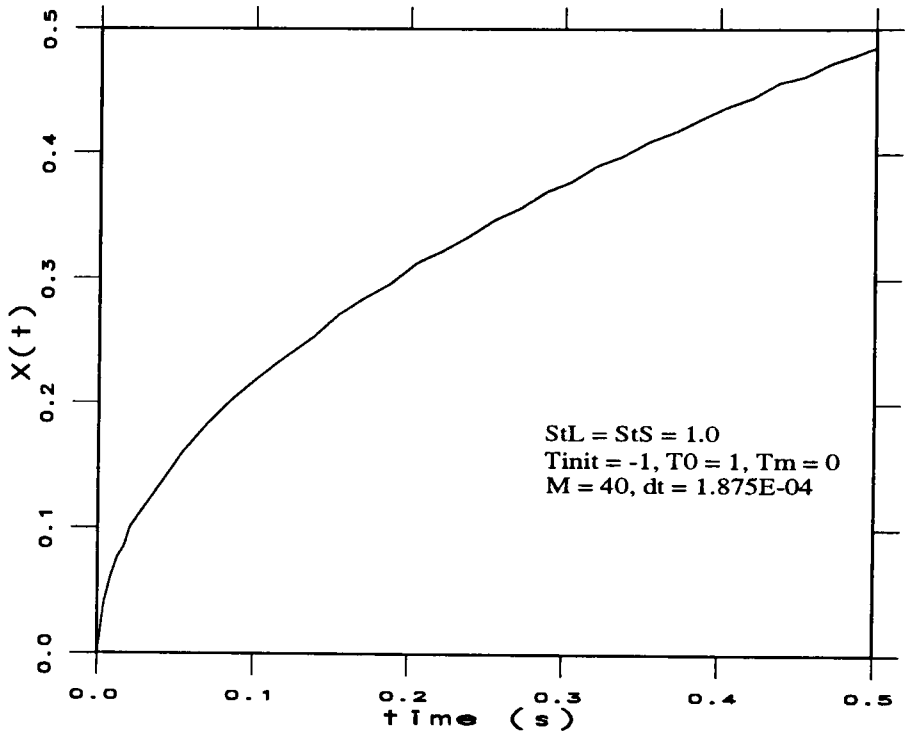


Figure 6: Front position when $St_S = St_L = 1.0$.

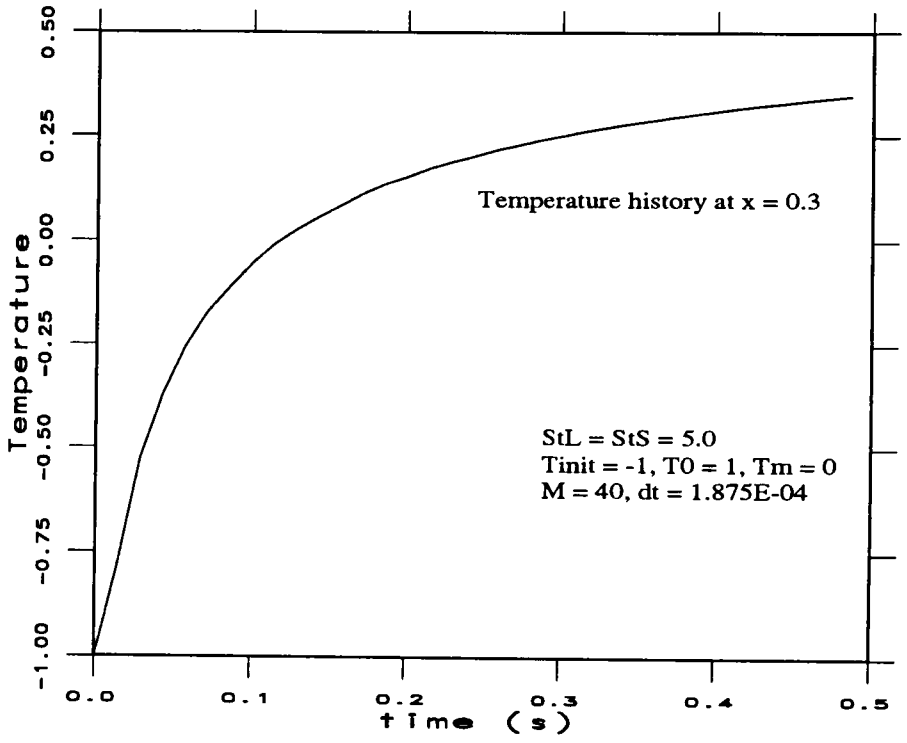


Figure 7: Fast melting case.

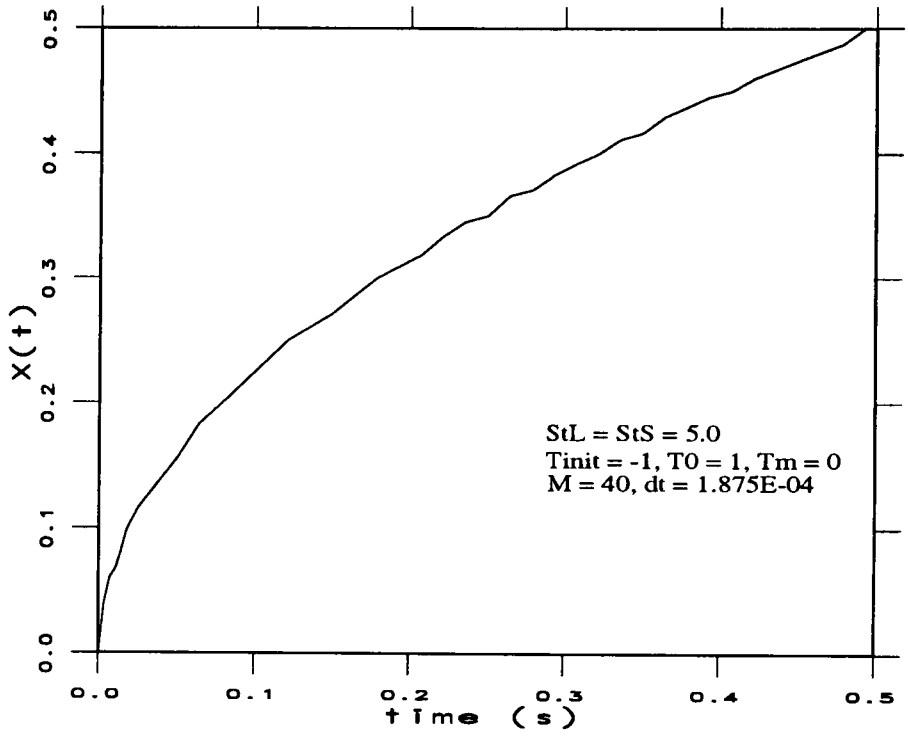


Figure 8: Front position when $St_S = St_L = 5.0$.

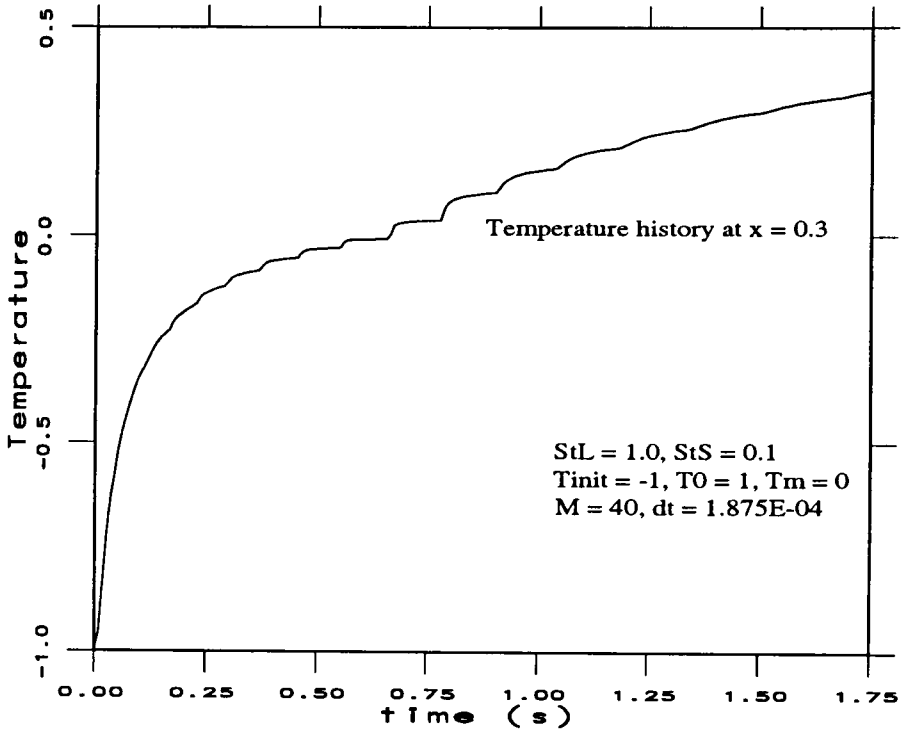


Figure 9: Melting case when $St_L = 1.0, St_S = 0.1, c_L = 10$.

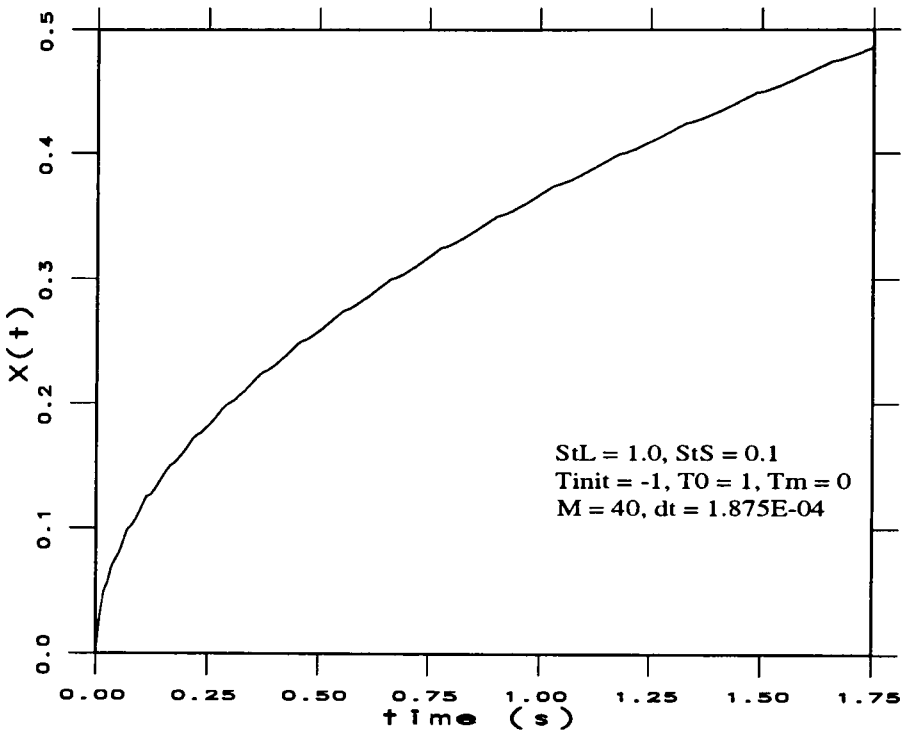


Figure 10: Front position when $St_L = 1.0, St_S = 0.1, c_L = 10$.

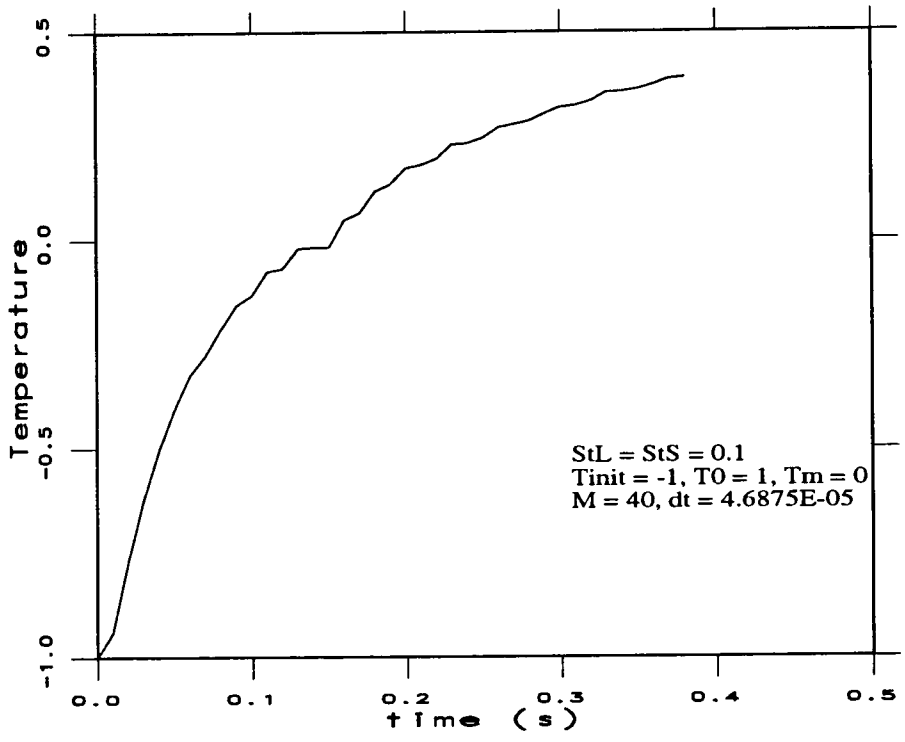


Figure 11: Melting case when $St_L = St_S = 0.1, \nu = 2$.

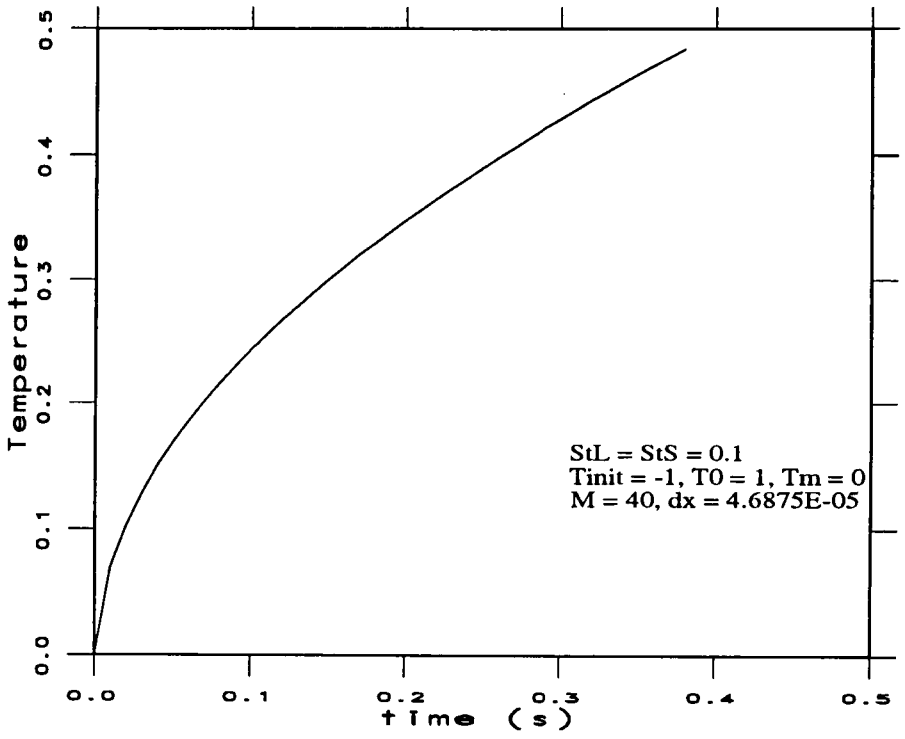


Figure 12: Front position when $St_L = St_S = 0.1, \nu = 2$.

# APPENDIX C

## Source Code

```
c  Rose2p.f   Source code in fortran for Thesis 2-phase problem.
c
c ..............................................
c
c  ..................... input variables .........
c  rho = density, cp = specific heats, zk = conductivities
c  zltht=latent heat, Tm=melt temperature,
c  einit = initial enthalpy,
c  Tinit = initial temperature, T0 = imposed temperature at x=0,
c  zlam = transc. root for Neumann solution (found separately)
c  zl= slab length, M= # of nodes, sigma= time factor multiplier
c  tmax= final computation time, dtout = output time-step,
c  numout = no. of output locations xout(k), k=1,numout.
c  maxits=max # of tridiagonal iterations,
c  maxsteps=max # of time steps
c
c  ..............................................
c
c  StL = stefan # due to t0, StS = stefan # due to Tinit
c  alpha = diffusivities, zniw = SQRT( alphaL/alphaS )
c
c  dx= space mesh = zl/2M, dt = time increment=sigma*.12dx*dx
c  del = dt/[4*(dx**2)], tol=iteration error=dx**2
c  xi(i)=grid points , dtperm=permanent time step
c  dtout=desired time interval for output, tout=output time
c  nsteps=# of steps thus far
c  gamma(i)= equation of state variable
c
c  ro(i)=initial vector=f(n)
c  rx(i)= vector U(k)
c  x(i)=vector solved by tridiagonal system
c  tn(i)=numerical temperatures
c  ta(i)=analytical temperatures
c  pdiff(i)=% difference between temperatures
```

```
c  ifrntn(i)=numerical front position
c  ifrnta(i)=analytical front position
c
c
c     The subroutines:
c        input-reads in data values and calculates variables
c        mesh-creates uniform mesh
c        init-initializes rx(i) as dictated by gamma(i)
c        conditions-calculates gamma(i), liquid frac.from rx(i)
c        tridiagsolv-builds matrix system from gamma(i) & rx(i)
c                    and then solves the system to get x(i)
c        supdiff-checks for SUP(ABS(rx(i)-x(i)))<tol
c        update-if SUP(ABS(rx(i)-x(i)))>tol,then rx(i)=x(i)
c        failed-prints output and method failed message
c        rosedata-calculates analytical solution,calculates
c                  temps. and outputs:numerical enthalpies,
c                  analytical enthalpies. Also calculates
                   Neumann & numerical temps,energy balance
c        compare-calculates errors
c        output-prints output at the desired time
c        finalout-gives final output values
c
c
c
c
      common rho,cpl,zkl,zk(400),tm,zltht,stl,cpl1,alphl,zlam
      common cps,zks,alphs,zniw,sts,cps1,frntn,front,zlthtm
      common einit,tinit,t0,m,dx,sigma,maxits,m2,n,dt,del,tol
      common time,dtout,tmax,maxsteps,xi(0:400)
      common ro(0:400),rx(0:400),its
      common gamma(400),x(400),c(400),rsupdiff,tn(400),ta(400)
      common tdiff(400),terr,zlamda(400),nsteps,itsttl,aveits
      common erfzlam,erfczlam,cfac1,cfac2,cfac3,jphase(400)
      common flxoldin,flxoldout,totEold,totE,totsenht,totLtht
      common engyBal,Egain,heatgain,heatBal,diffT(20)
      common numout,xout(20),ixout(20,2),errL1,errL1max
      common errX,errXmax,Tempout(20),TNeumout(20),errT,errTmax
c
c
      call input
      call mesh
c
```

```
      dtperm=dt
      tout=MAX(dtout,dt)
      time=0.0
c
      call init
 100  continue
      time=time+dt
      nsteps=nsteps+1
      if((time.gt.tmax).or.(nsteps.gt.maxsteps))goto 300
      its=1
 200  if(its.le.maxits)then
        call conditions
        call tridiagsolv
        call supdiff
        if(rsupdiff.le.tol)then
          call conditions
          call rosedata
          call compare
          tpdt=time+dt
          call outhist
          if(time.eq.tout)then
            call output
            dt=dtperm
            tout=tout+MAX(dtout,dt)
          elseif(tpdt.gt.tout)then
            dt=tout-time
          endif
          itsttl=itsttl+its
          goto 100
        else
          its=its+1
          goto 200
        endif
      else
        call failed
        stop
      endif
      goto 100
 300  continue
      call finalout
      end
c
```

```
c
c
      subroutine input
c
c     <<<<<common block>>>>>
c
c      read(5,*) rho,cpl,cps,zks,zkl,tm,zltht
c      read(5,*) einit,t0,m,zl,maxits
c      read(5,*) sigma,dtout,tmax,maxsteps
c      read(5,*) zlam
c
      read(5,*) rho,cpl,cps,zks,zkl,tm,zltht
      read(5,*) einit,t0,maxits,maxsteps,sigma
      read(5,*) zl,M,zlam,tmax,dtout,numout
      read(5,*) (xout(k),k=1,numout)
      read(5,*)
      read(5,*)
c
      tinit=(einit+zltht)/cps
      alphl=zkl/(rho*cpl)
      alphs=zks/(rho*cps)
      zniw=SQRT(alphl/alphs)
      stl=cpl*(t0-tm)/zltht
      sts=cps*(tm-tinit)/zltht
      m2=2*m
      n=m2-1
      dx=zl/m2
c      dt=dx/alphl/stl*sigma*dx
      dt = 1.2*dx*dx*sigma
      del=dt/4.0/dx/dx
c      tol=MAX(1.e-6,dx**4)
      tol=1.e-5
c
      zlthtm=-1.0*zltht
      cpl1=1.0/cpl
      cps1=1.0/cps
      cfac1=2.0*zks*del*zltht/cps
      cfac2=2.0*zks*zltht/cps
      cfac3=zks*zltht/cps
      erfzlam=ERF(zlam)
      erfczlam=1.0-ERF(zlam*zniw)
c
```

```
c     ...Neumann melt time of midpoint 1/2m...
      tmeltNeu=zl**2/(16.0*zlam*zlam*alphl)
c     ...Neumann melt time of .75 1/2m...
c      tmeltNeu=9.*zl**2/(4.*16.0*zlam*zlam*alphl)
      tmax=tmeltNeu
c
      print*,'physical props: cpl,cps,zks,zkl,tm,zltht:'
      print*,cpl,cps,zkl,zks,tm,zltht
      print*,'input:einit,tinit,t0,m,l:',einit,tinit,t0,m,zl
      print*,'input:dtout,tmax:',dtout,tmax
      print*,'sigma,dx,dt:',sigma,dx,dt
      print*,'sts,stl,zlam:',sts,stl,zlam
      return
      end
c
c
c
      subroutine mesh
c
c     <<<<<common block>>>>>
c
      xi(0)=0.0
      do 2,i=1,m2
        xi(i)=xi(i-1)+dx
 2    continue
c      print*,'nodes'
c      print*,(xi(i),i=1,n,2)
      print*,'output nodes'
      print*,(xout(i),i=1,numout)
c     ...find neighboring nodes for output nodes...
      errDx=1.e-3*dx + dx
      do 5, k=1,numout
         do 4, i=1,n,2
            val=ABS(xi(i)-xout(k))
            if( val .lt. errDx )then
              ixout(k,1) =  i
              ixout(k,2) = i+2
              goto 5
            endif
 4       continue
 5    continue
c      print*,'neighboring nodes to output nodes'
```

```
c      do 6, k=1,numout
c        print*,(ixout(k,i),i=1,2)
c   6   continue
c      print*,'*****************************'
       return
       end
c
c
c
       subroutine init
c
c      <<<<<common block>>>>>
c
       ro(0)=t0
       rx(0)=ro(0)
       do 3,i=1,m2
         if(mod(i,2).eq.0.0)then
           ro(i)=0.0
           rx(i)=tinit
         else
           ro(i)=einit
           rx(i)=ro(i)
         endif
 3     continue
       ro(m2)=rx(m2)
       return
       end
c
c
c
       subroutine conditions
c
c      <<<<<common block>>>>>
c
c      ...calculate conditions for odd nodes...
c
       do 20,i=1,n,2
           if(rx(i).le.zlthtm)then
             gamma(i)=cps1
             zk(i)=zks
             zlamda(i)=0.0
             c(i)=cfac1
```

```
            elseif(rx(i).ge.0.0)then
              gamma(i)=cpl1
              zk(i)=zkl
              zlamda(i)=1.0
              c(i)=0.0
            else
              gamma(i)=0.0
              zlamda(i)=1.0+(rx(i)/zltht)
c              zk(i)=zkl*zks/(zks*zlamda(i)+zkl*(1.0-zlamda(i)))
              zk(i)=zks*zlamda(i)+zkl*(1.0-zlamda(i))
              c(i)=0.0
            endif
 20     continue
c
c       ...calculate conditions for even nodes...
c
        do 30,i=2,n+1,2
          if(rx(i-1).le.zlthtm)then
              c(i)=cfac2
          elseif((rx(i-1).gt.zlthtm).and.(rx(i+1).le.zlthtm))then
              c(i)=cfac3
          else
              c(i)=0.0
          endif
 30     continue
c
c       ...calculate rx(m2) at half time...
        htime=time+dt/2.0
        htfac=2.0*SQRT(alphs*htime)
        erfczlam=1.0-ERF(zlam*zniw)
        rx(m2)=tinit+(tm-tinit)*ERFC(1.0/htfac)/erfczlam
        ro(m2)=rx(m2)
        return
        end
c
c
c
      subroutine tridiagsolv
c
c       <<<<<common block>>>>>
c
      dimension a(400,400),z(400),zl(400,400),u(400,400)
```

```
c
c     ...construct b vector...
c
      a(1,n+1)=ro(1)+zk(1)*del*ro(0)-c(1)
      do 30,i=2,n-1
         a(i,n+1)=ro(i)-c(i)
 30   continue
      a(n,n+1)=ro(n)+zk(n)*del*ro(n+1)-c(i)
c      print*,'b vector'
c      print*,(a(i,n+1),i=1,n)
c
c     ...construct A...
c
      a(1,1)=1.0+2.0*zk(1)*del*gamma(1)
      a(1,2)=-1.0*del*zk(1)
      do 1,i=2,n-1
        if(mod(i,2).eq.0.0)then
c     ...even row of A calcutates eq.**...
          do 2,j=2,n-1
            if(i.eq.j)then
              a(i,j)=-1.0*(zk(j-1)+zk(j+1))
              a(i,j-1)=zk(j-1)*gamma(j-1)
              a(i,j+1)=zk(j+1)*gamma(j+1)
            endif
 2        continue
        else
c     ...odd row of A calculates eq *...
          do 3,j=2,n-1
            if(i.eq.j)then
              a(i,j)=1.0+2.0*zk(j)*del*gamma(j)
                if(a(i,j).eq.0.0)then
                  print*,'Matrix A is singular'
                  stop
                endif
              a(i,j-1)=-1.0*zk(j)*del
              a(i,j+1)=-1.0*zk(j)*del
            endif
 3        continue
        endif
 1    continue
      a(n,n-1)=-1.0*zk(n)*del
      a(n,n)=1.0+2.0*zk(n)*del*gamma(n)
```

```
c       print*,'A matrix:'
c       do 11,i=1,n
c         print*,(a(i,j),j=1,n)
c 11    continue
c
c       ...factor A into L & U...
c
        zl(1,1)=a(1,1)
        u(1,1)=1.0
        u(1,2)=a(1,2)/zl(1,1)
c       ..............................
        do 4,i=2,n-1
           zl(i,i-1)=a(i,i-1)
           zl(i,i)=a(i,i)-(zl(i,i-1)*u(i-1,i))
           u(i,i+1)=a(i,i+1)/zl(i,i)
           u(i,i)=1.0
 4      continue
c       ..............................
        zl(n,n-1)=a(n,n-1)
        zl(n,n)=a(n,n)-zl(n,n-1)*u(n-1,n)
        u(n,n)=1.0
c
c       ...solve Lz=b...
c
        z(1)=a(1,n+1)/zl(1,1)
        do 5,i=2,n
          z(i)=(a(i,n+1)-zl(i,i-1)*z(i-1))/zl(i,i)
 5      continue
c
c       ...solve Ux=z...
c
        x(n)=z(n)
        do 6,i=1,n-1
          j=n-i
          x(j)=z(j)-(u(j,j+1)*x(j+1))
 6      continue
c       print*,'x vector:'
c       print*,(x(i),i=1,n)
        return
        end
c
c
```

```
c
      subroutine supdiff
c
c     <<<<<common block>>>>>
c
      rdiffl=0.0
      rsupdiff=0.0
      do 7 l=1,n
        rdiffl=ABS(rx(l)-x(l))
        rsupdiff=MAX(rdiffl,rsupdiff)
 7    continue
      do 9,i=1,n
         rx(i)=x(i)
 9    continue
c     ...claculate flux at .5dt...
c      if(rsupdiff.le.tol)then
c         if(rx(1).le.zthhtm)then
c            tn(1)=(rx(1)+zltht)/cps
c            zk(1)=zks
c         elseif(rx(1).ge.0.0)then
c            tn(1)=rx(1)
c            zk(1)=zkl
c         else
c            tn(1)=0.0
c            zlamda(1)=1.0+(rx(1)/zltht)
c            zk(1)=zkl*zks/(zks*zlamda(1)+zkl*(1.0-zlamda(1)))
c            zk(i)=zkl*zks/(zks*zlamda(i)+zkl*(1.0-zlamda(i)))
c         endif
c         if(rx(n).le.zlthtm)then
c            tn(n)=(rx(n)+zltht)/cps
c         elseif(rx(n).ge.0.0)then
c            tn(n)=rx(n)/cpl
c         else
c            tn(n)=0.0
c            zlamda(n)=1.0+(rx(n)/zltht)
c            zk(n)=zkl*zks/(zks*zlamda(n)+zkl*(1.0-zlamda(n)))
c         endif
c         flxoldin=zk(1)*(t0-tn(1))/dx
c         flxoldout=zk(n)*(tn(n)-ro(m2))/dx
c      endif
      return
      end
```

```
c
c
c
      subroutine rosedata
c
c     <<<<<common block>>>>>
c
c
c     ...update b vector...
      do 7,i=1,n,2
        if(rx(i).le.zlthtm)then
          ro(i)=rx(i)+zk(i)*del*
     &          (rx(i-1)-2.0*gamma(i)*(rx(i)+zltht)+rx(i+1))
        else
          ro(i)=rx(i)+zk(i)*del*(rx(i-1)
     &                -2.0*gamma(i)*rx(i)+rx(i+1))
        endif
 7    continue
c     ...set rx(i) = ro(i), i odd...
      do 9,i=1,n,2
        rx(i)=ro(i)
 9    continue
c
c     ...find the temps & energy balance...
c     ...calculate Neumann solution...
c
      tfac1=2.0*SQRT(alphl*time)
      tfac2=2.0*SQRT(alphs*time)
      frntn=zlam*tfac1
      do 11,i=1,n,2
         if(xi(i).lt.frntn)then
           ta(i)=t0-(t0-tm)*ERF(xi(i)/tfac1)/erfzlam
         else
           ta(i)=tinit+(tm-tinit)*ERFC(xi(i)/tfac2)/erfczlam
         endif
 11   continue
      rx(m2)=tinit+(tm-tinit)*ERFC(1.0/tfac2)/erfczlam
c     ...calculate numerical temps,assign phase & front...
      do 13,i=1,n,2
        if(ro(i).le.zlthtm)then
          tn(i)=tm+(ro(i)+zltht)/cps
          zlamda(i)=0.0
```

```
            zk(i)=zks
            jphase(i)=0
          elseif(ro(i).ge.0.0)then
            tn(i)=tm+ro(i)/cpl
            zlamda(i)=1.0
            zk(i)=zkl
            jphase(i)=2
          else
            tn(i)=tm
            zlamda(i)=1.0+(ro(i)/zltht)
c            zk(i)=zkl*zks/(zks*zlamda(i)+zkl*(1.0-zlamda(i)))
            zk(i)=zkl*zks/(zks*zlamda(i)+zkl*(1.0-zlamda(i)))
            jphase(i)=1
            front=(i-1+2.0*zlamda(i))*dx
          endif
 13     continue
c       ...sum the energies...
c
c       totEold=totE
c       totE=0.0
c       totsenht=0.0
c       do 23,i=1,n,2
c          totE=totE+ro(i)
c          totsenht=totsenht+rho*cpl*(tn(i)-tm)*zlamda(i)
c      &         +rho*(cps*(tn(i)-tm)-zltht)*(1.0-zlamda(i))
c 23    continue
c       totE=2.0*totE*dx
c       totsenht=2.0*totsenht*dx
c       ...energy balance should be zero...
c       engyBal=totE-totsenht
c       ...energy gain should = heat gain...
c       Egain=totE-totEold
c       heatgain=sigma*dx*(flxoldin-flxoldout)
c       heatBal=Egain-heatgain
c       flxoldin=zk(1)*(t0-tn(1))/dx
c       flxoldout=zk(n)*(tn(n)-rx(m2))/dx
        return
        end
c
c
c
        subroutine compare
```

```
c
c     <<<<<common block>>>>>
c
      dimension gnn(0:400),gdd(0:400)
c
c     ... calculate front error...
      errX=ABS(frntn-front)
      errXmax=MAX(errXmax,errX)
c
c     ...calculate the output node temps...
c     ...by linear interpolation...
c
      Tempout(1)=T0
      TNeumout(1)=T0
      errTout=0.0
      do 100,k=2,numout
        jm=ixout(k,1)
        jp=ixout(k,2)
        xt=xout(k)
        xm=xi(jm)
        Tempout(k)=tn(jm) + (tn(jp)-tn(jm))*(xt-xm)/(2.0*dx)
c
c    ...Neumann temps...
c
       tfac1=2.0*SQRT(alphl*time)
       tfac2=2.0*SQRT(alphs*time)
c      frntn=zlam*tfac1
        if(xout(k).lt.frntn)then
         TNeumout(k)=t0-(t0-tm)*ERF(xout(k)/tfac1)/erfzlam
        else
         TNeumout(k)=tinit+(tm-tinit)
     &                        *ERFC(xout(k)/tfac2)/erfczlam
        endif
c
c     ..calculate diff at each output node & maximun error..
c
        diffT(k)=ABS(Tempout(k)-TNeumout(k))
        errTout=MAX(diffT(k),errTout)
 100  continue
c      Tempout(m2)=rx(m2)
c      TNeumout(m2)=rx(m2)
c     ...calculate maximum differenced so far...
```

```
      errTmax=MAX(errTmax,errTout)
c
c
c     ..calculate relative error using L1 norms at output pts..
c     ...L1 error is found using Simpson's rule...
c
      gnn(0)=0.0
      gdd(0)=t0+rx(m2)
      errsum1=gnn(0)
      areaTN=gdd(0)
c      k=0
      do 9,i=1,numout
        if(mod(i,2).gt.0.0)then
           gnn(i)=4.0*diffT(i)
           gdd(i)=4.0*ABS(TNeumout(i))
        else
           gnn(i)=2.0*diffT(i)
           gdd(i)=2.0*ABS(TNeumout(i))
        endif
        errsum1=errsum1+gnn(i)
        areaTN=areaTN+gdd(i)
 9    continue
      errsum1=dx*(errsum1)/3.0
      areaTN=dx*(areaTN)/3.0
      if(areaTN.eq.0.0)areaTN=1.0
      errL1=errsum1/areaTN
      errL1max=MAX(errL1,errL1max)
c      print*,'error using L1 norms: ',errL1
c      print*,'max L1 error thus far: ',errL1max
c      print*,'************************************'
      return
      end
c
c
c
      subroutine outhist
c
c     <<<<<common block>>>>>
c
      if(time.lt.touthist)goto 10
c      if(time.lt.0.05)then
c        touthist=touthist+dtout/20.
```

```
c       else
         touthist=touthist+dtout/5.
c       endif
c       write(7,35) time,front
       write(7,34) time,Tempout(3),TNeumout(3),
      &                      diffT(3),front,frntn,errX
 34    format(3f10.4,f10.6,3f10.6)
c 35    format(f10.4,f10.6)
 10    return
       end
c
c
c
       subroutine output
c
c      <<<<<common block>>>>>
c
       dimension gn(0:400),gd(0:400)
c
       print*,'time of output:',time
       print*,'# of steps so far: ',nsteps
       print*,'# of tridiag iterations so far: ',itsttl
       aveits=real(itsttl)/real(nsteps)
       print*,'average # of tri-diag its per step:',aveits
       print*,'Neumann & numerical front position:'
       print*,frntn,front
c       fntdif=ABS(frntn-front)
       print*,'front tracking error this time step:',errX
c       fdifmax=MAX(fdifmax,fntdif)
       print*,'maximum front tracking error thus far:',errXmax
c       print*,'phase at each node:'
c       print*,(jphase(i),i=1,n,2)
c       print*,'Neumann temps'
c       print*,(ta(i),i=1,n,2)
c       print*,'numerical temps'
c       print*,(tn(i),i=1,n,2)
c       print*,' difference in temps at each node:'
c       print*,(tdiff(i),i=1,n,2)
c
c      ...calculate  difference at each node & maximun error...
c
       terr=0.0
```

```
      do 17,i=1,n,2
        tdiff(i)=ABS(tn(i)-ta(i))
        if(jphase(i).eq.1)tdiff(i)=0.
        terr=MAX(tdiff(i),terr)
 17   continue
      terrmax=MAX(terrmax,terr)
      print*,'max|TNeumann-T(j)| at nodes: ',terr
      print*,'maximum difference in temps thus far:',terrmax
      print*,'maximum difference in output temps:',errTmax
      print*,'error at output nodes using L1 norms: ',errL1
      print*,'max L1 error thus far: ',errL1max
c      print*,'energy balance: ',engyBal
c      print*,'heat balance: ',heatBal
c
c     ...calculate relative error using L2 norms...
c     ...L2 error is found using Simpson's rule...
c
      gn(0)=0.0
      gd(0)=t0**2+rx(m2)**2
      errsum1=gn(0)
      areaTN=gd(0)
c      k=0
      do 9,i=1,n
        if(mod(i,2).gt.0.0)then
           gn(i)=4.0*tdiff(i)**2
           gd(i)=4.0*ta(i)**2
        else
           gn(i)=2.0*((tn(i-1)+tn(i+1)/2.0)
     &                -(ta(i-1)+ta(i+1)/2.0))**2.0
           gd(i)=2.0*(ta(i-1)+ta(i+1)/2.0)**2
        endif
        errsum1=errsum1+gn(i)
        areaTN=areaTN+gd(i)
 9    continue
      errsum1=dx*(errsum1)/3.0
      areaTN=dx*(areaTN)/3.0
      if(areaTN.eq.0.0)areaTN=1.0
      errL2=SQRT(errsum1)/SQRT(areaTN)
      errL2max=MAX(errL2,errL2max)
      print*,'error using L2 norms: ',errL2
      print*,'max L2 error thus far: ',errL2max
      print*,'************************************'
```

```
      print*,'************************************'
      return
      end
c
c
c
      subroutine failed
c
c     <<<<<common block>>>>>
c
      print*,'method failed ',its,' iterations'
      print*,'T0,e1,...,e2M-1,T2M'
      rx(0)=ro(0)
      rx(m2)=gamma(n)*rx(n)
      print*,(rx(n),n=0,m2)
      return
      end
c
c
c
      subroutine finalout
c
c     <<<<<common block>>>>>
c
      print*,'time of final output:',time
      print*,'# of steps: ',nsteps
      print*,'# of tri-diag iterations: ',itsttl
      aveits=real(itsttl)/real(nsteps)
      print*,'average # of tri-diag its per step:',aveits
      print*,'Neumann & numerical front position:'
      print*,frntn,front
      print*,'front tracking error this time step:',errX
      print*,'maximum front tracking error thus far:',errXmax
c      print*,'max|TNeumann-T(j)| at nodes: ',terr
c      print*,'max difference in temps at nodes:',terrmax
c      print*,'error at nodes using L2 norms: ',errL2
c      print*,'max L2 at nodes error thus far: ',errL2max
      print*,'maximum difference in output temps:',errTmax
      print*,'error at output nodes using L1 norms: ',errL1
      print*,'max L1 error thus far: ',errL1max
c      print*,'energy balance: ',engyBal
c      print*,'heat balance: ',heatBal
```

```
print*,'****************end******************'
return
end
```

# VITA

Michael J. Gilbert was born on November 11, 1959. He received a Bachelor's Degree from Shippensburg University in Shippensburg, Pennsylvania, in 1989. He continued his education to study for a Master's of Science degree in Mathematics at The University of Tennessee, Knoxville. He was awarded this degree in 1992.