

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

**A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600**

**NUMERICAL SIMULATION OF LASER ABLATION IN
PULSED LASER DEPOSITION PROCESS**

A Thesis

**Submitted to the Graduate Faculty of
The University of South Alabama
in partial fulfillment of requirement for the Degree of**

**Master of Science
in
Mechanical Engineering**

**by
Zhaohui Wei
B. S., Mechanical Engineering
Chengdu University of Science and Technology, China, 1990
December, 1997**

UMI Number: 1387220

**UMI Microform 1387220
Copyright 1997, by UMI Company. All rights reserved.**

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

The University of South Alabama
Department of Mechanical Engineering

NUMERICAL SIMULATION OF LASER ABLATION IN PULSED LASER
DEPOSITION PROCESS



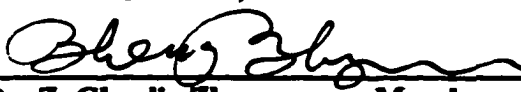


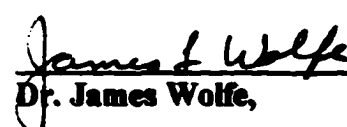
By
Zhaohui Wei
A Thesis

Submitted to the Graduate Faculty of The University of South Alabama
in partial fulfillment of requirement for the Degree of

Master of Science
in
Mechanical Engineering

December, 1997

APPROVED:

	12-1-97
Dr. Jayanta S. Kapat, Chair of Thesis Committee	Date
	11-11-97
Dr. Ashok Kumar, Co-Chair of Committee	Date
	12/1/97
Dr. Z. Charlie Zheng, Member of Committee	Date
	12/1/97
Dr. Ali E. Engin, Chair of Department & Member of Committee	Date
	12-3-97
Dr. Keith Harrison, Director of Graduate Studies	Date
	12-5-97
Dr. James Wolfe, Dean of Graduate School	Date

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my thesis advisor, Dr. Jayanta S. Kapat, for his sponsorship, guidance, encouragement, and patience throughout the development of this thesis.

I would like to thank Dr. Z. Charlie Zheng for his direction and assistance in my graduate study, and Dr. Ashok Kumar for his sponsorship and guidance in my thesis work. I would like to thank Dr. Ali Engin for his reviewing of my thesis and for giving me encouragement to pursue a Ph.D. study, and Dr. Eugene I. Odell for his teaching.

Special thanks to Ms. Nancy Reeves, the secretary of Department of Mechanical Engineering of the University of South Alabama, for her precious assistance and friendly attitude toward me during my graduate study and research work.

My thanks also go to Mr. Dajiang Wang, Mr. Gangfu Zhang, Mr. Padungsak Unontakarn, Mr. Ki-Hyun Baek, Mr. David D. Budlong, Mr. Arthur Durkin, Mr. Gary Landsberg, Majdi Alhasan and Mr. Robertson Frederick for their assistance and friendship while we were working together in the Computational Fluids Lab of the Department of Mechanical Engineering of the University of South Alabama.

Appreciation is extended to the Department of Mechanical Engineering and the University of South Alabama for providing me this education toward a Master's degree.

Finally, I would like to thank my parents, my sister, and my brother for their love.

This research was supported by Alabama NASA EPSCoR (Experimental Program to Simulate Competitive Research) program.

TABLE OF CONTENTS

LIST OF TABLES	v
LIST OF FIGURES	vi
LIST OF NOTATIONS	vii
ABSTRACT	ix
1. INTRODUCTION	1
2. BACKGROUND	2
2.1. PLD Process	2
2.2. Laser - Solid Interaction	4
2.3. Moving Boundary Problems	6
2.3.1. An Overview of Phase-Change Problems	6
2.3.2. Assumptions	8
2.3.3. Stefan Condition	9
3. PROBLEM STATEMENT	11
3.1. Governing Equations of Unsteady Heat Conduction	12
3.2. Boundary and Initial Conditions	13
3.2.1. Imposed Flux	13
3.2.2. Convective Flux	14
3.2.3. Interface Condition	14
3.2.4. Role of Kinetics in Melting	15
3.3. Mathematical Model	17
3.3.1. Heat Transfer before Phase Transition	17

3.3.2. Heat transfer with Phase Transition	18
3.4. Further Consideration	19
4. DISCRETIZATION ALGORITHM	25
4.1. Some Basic Aspects of Numerical Computations	26
4.1.1. Iterative Methods.....	27
4.1.2. Numerical Errors	27
4.1.3. Stability of Numerical Methods	28
4.2. Computational Methods	29
4.2.1. Finite-Difference	29
4.2.2. Finite-Element and Its Comparison with Finite-Difference	30
4.2.3. MWR and Its Spectral Methods	31
4.3. Chebyshev Collocation Method	32
4.3.1. Selection of Spectral method	33
4.3.2. Description of Chebyshev Collocation Method	34
4.4. Discretization of Time Derivatives.....	38
4.4.1. Adams and Backwards Differentiation formulas	39
4.4.2. Predictor-Corrector	42
5. NUMERICAL MODELS	44
5.1. Non-dimensional Discrete Equations	44
5.2. The Flowchart of Programming	48
6. RESULTS AND CONCLUSIONS	51
REFERENCES	56
APPENDIX (Computer Program)	59
VITA	86

LIST OF TABLES

Table

1. Coefficients $\{\beta, \}$ of Adams-Bashforth formulas 40
2. Coefficients $\{\beta, \}$ of Adams-Moulton formulas 41
3. Coefficients $\{\beta, \}$ of Backwards Differentiation Formulas 41
4. Thermodynamic Properties of Target (Pure Iron) & Test Parameters 51

LIST OF FIGURES

Figure

1. Schematic of Pulsed Laser Deposition Process	2
2. Simplified One-Dimensional Model of PLD Process	3
3. Interface Between Liquid and Solid Regions	9
4. Geometric Parameters of the Target (as used in this problem)	11
5. Schematic Variation of Free Energy in Melting	16
6. Geometric Parameters of the Target in Laser Ablation	19
7. A Typical Excimer Laser Pulse	21
8. Variation of $\theta_s(-1, \tau)$ with respect to τ Around the Critical Point (Melting Temperature)	49
9. Change in Melting Interface Location with Time	52
10. Change in Melt Interface Temperature with time	53
11. Change in Free Surface Temperature with time	54
12. 2-Phase Temperature Profiles	54

LIST OF NOTATIONS

c	Specific heat
h	Heat transfer coefficient
I	Laser intensity
I_s	Time-dependent laser intensity arriving at the surface of the target
k	Thermal conductivity
k_B	Boltzman constant
l	Thickness of target
L	Latent heat (heat of fusion)
M	molecular
m	Mass of the ejected particles (molecules)
m_e	Electron mass
N_A	Avogadro's number
n_c	Number density of molecules in the condensed phase
P_0	Reference pressure used in the Clausius-Clapeyron equation
q''	Laser heat flux at the surface
R	Surface reflectivity
t	Time
T	Temperature of target
T_a	Ambient temperature
T_D	Debye temperature of the material
T_e	Temperature of free electron
T_E	Equilibrium melting temperature
T_p	Temperature of phonon

T_i	Initial temperature of target
T_m	Melting temperature in the moving interface
V_s	Speed of sound in the solid
V_c	Characteristic speed of the material
V_v	Speed of vaporization front
X	Moving front (interface) between liquid phase and solid phase, a function of time
x	Spatial coordinate
α_l	Linear absorption coefficient for volume absorption
α	Thermal diffusivity
Δg_b	Gibbs free energy change due to the phase transition barrier
Δg_m	Net Gibbs free energy change due to melting (per molecule)
Δg_v	Latent heat of vaporization per molecule
\hbar	Angular Planck constant
ρ	Mass density
Σ	Physical moving front (interface)
τ_r	Thermal relaxation time
Ω	Physical domain

Subscripts

L	Liquid region
S	Solid region

ABSTRACT

Wei, Zhaohui, M. S. M. E., University of South Alabama, December, 1997. Numerical Simulation of Laser Ablation in Pulsed Laser Deposition Process. Chair of Committee: Dr. Jayanta S. Kapat

A computational model has been developed in order to simulate heat transfer with phase transition in laser ablation of Pulsed Laser Deposition (PLD) Process with or without the kinetics condition. Front fixing method with Landau transformation has been used to represent the geometric domain. Spectral collocation method has been utilized for spatial discretization in this model, while backward differentiation techniques have been used for temporal discretization.

The computational model has been applied to the problem of laser melting of a pure iron sheet with a medium power laser. In order to understand the importance of kinetics of melting, the problem has been solved with and without the kinetics condition at the moving interface, and the results have been compared. It shows that if kinetics condition is not used, the interface location is over predicted and the free interface temperature is under-predicted. Both of these factors may have important consequences in the laser ablation of PLD process.

1. INTRODUCTION

Pulsed Laser Deposition (PLD) plays a significant role in the processing of advanced materials. For the case of multi-elemental compounds such as high-temperature superconductors, ferroelectrics, electro-optic materials, this technique has been extremely successful in a short span of time [Chrisey and Huber, 1994; Miller, 1994] [1, 2].

If a mathematical model of a physical process may be thought of as a simulation of this process by using mathematical tools, then, in the same spirit, a laboratory-scale experiment of an industrial process is an imitation of the process by the means and capabilities of the laboratory, and a numerical simulation is an imitation of the process by the means and capabilities of the computer [Alexiades and Solomon, 1993]. Because of the importance and industrial viability of the PLD technique, a computational model to predict the physical processes that take place during PLD will provide a cost-effective and time-efficient tool for design optimization, and will help to limit the use of the expensive and time-consuming experimental tests to the last stages of design verification [3].

Currently, efforts are being made through this research work to incorporate the effect of kinetics of phase transitions and other non-equilibrium phenomena into a computational model for laser ablation in PLD process [4]. A mathematical model to include kinetics of melting in laser ablation has been considered in this thesis. A numerical model has been developed and results for a simple test case have been presented. For comparison, results from numerical simulations based on Stefan condition where the kinetics of melting is ignored are also presented.

2. BACKGROUND

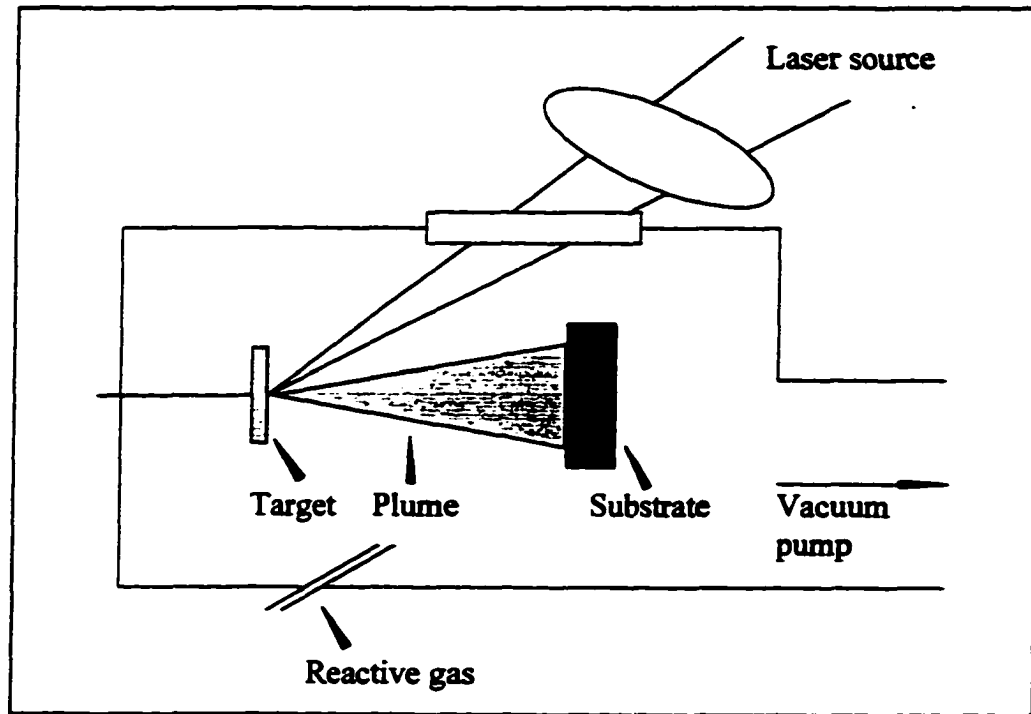


Figure 1
Schematic of Pulsed Laser Deposition Process

2.1. PLD Process

PLD is a technique for thin film deposition, which can be described as a three-step process [5]: (i) vaporization of a target material under laser irradiation; (ii) transport of the vapor plume; (iii) film growth on a substrate. Figure 1 shows a schematic diagram of an experimental setup, which consists of a target holder and a substrate holder housed in a vacuum chamber. A high-power laser that is used as an external energy source is irradiated

on the target surface to vaporize materials and to produce a plume. This plume impinges onto the substrate surface and deposits a thin film with the same stoichiometry as the target. A set of optical components is used to focus and raster the laser beam over the target surface. This method is easily adaptable to different operational modes because of decoupling of the vacuum hardware and the evaporation power source (i.e. the laser). Film growth can be carried out in a reactive environment containing one or more gases. It can also be operated in conjunction with other types of evaporation sources in a hybrid approach [1, 2, 5, and 6].

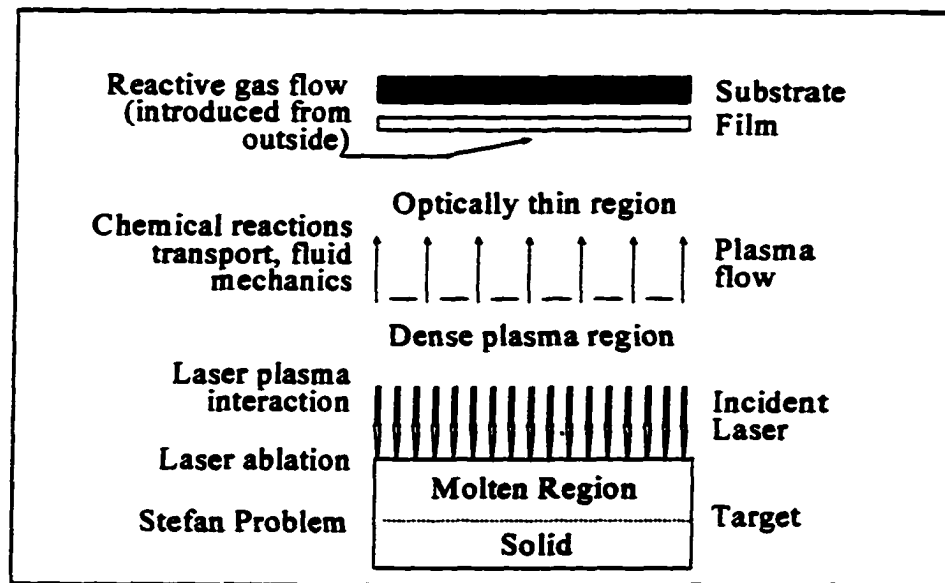


Figure 2
Simplified One-Dimensional Model of PLD Process

The first PLD experiment was carried out in 1965 by Smith and Turner. A ruby laser was used to deposit thin films on different substrates. The advantages of the PLD technique stem from the fact that the dislodgment of atoms from a source is accomplished by depositing energy at the target surface via a laser pulse that is a source of "pure" energy in the form of monochromatic and coherent photons. Unlike ions or electrons, laser beams are much easier to transport and manipulate, and since laser interaction with gas-phase

species is relatively weak, the dynamic range of deposition pressure is the largest compared to any deposition processes. Furthermore, at wavelengths of 250 nm and below, all materials absorb the laser beam either via linear or nonlinear processes whereby coupling of energy is possible to most surfaces, making the process very adaptable. The capability of the technique to reproduce the target composition in the deposited film with relative ease under the appropriate conditions, is one of the key features. Deposition of multi-layers involves sequential ablation of multiple targets with a laser beam in a relatively straightforward operation, is another key feature of the technique. Since any materials can be ablated with an appropriate laser, this technique is not as much affected by the properties of the target as the other techniques. The process is so different in many ways from conventional deposition processes that it will take many years to fully realize and utilize all the unique features of PLD. The past research in PLD indicates a breathtaking scenario for this technology in the future [2, 6, 7, and 8].

2.2. Laser-Solid Interaction

The problem of laser-target interaction was studied long before the first PLD experiment. In contrast to the simplicity of the hardware of PLD, the laser-solid interaction is a very complex physical phenomenon. Theoretical descriptions are multidisciplinary and combine both equilibrium and non-equilibrium processes. The effect of electromagnetic energy of laser pulses irradiating a solid surface, if sufficiently intense, is to cause evaporation, ablation, excitation, plasma formation, and exfoliation. Electromagnetic energy is converted to electronic, thermal, chemical, and mechanical energy at the solid surface [2, 6].

The ablation threshold of a solid at a certain wavelength is determined by the amount of laser power required to reach a surface temperature that gives rise to significant ablation or material removal. Ablated material forms a "plume" consisting of a mixture of energetic species including atoms, molecules, electrons, ions, clusters, micron-sized solid

particulate, and molten globules. The mean free path inside the dense plume is very short. As a result, immediately after the laser irradiation, the plume rapidly expands into the background environment (which can be vacuum or may contain some reactive gases) from the target surface. Gasdynamic flow characteristics of the expanding plume are similar to those of an expanding nozzle jet. During a PLD process, the spatial distribution of the plume is influenced by target topography, target-substrate distance, laser spot dimensions, ambient gas, gravity, film, and plume composition [2, 6, and 7]. The plume propagation also involves complex gasdynamic interactions such as plume splitting [Geohegan, 1997].

Mechanisms that lead to material ablation depend on laser characteristics, as well as the optical, topological, and thermodynamical properties of the target. When a laser beam irradiates on a target surface that is metallic, the incident energy is absorbed primarily by free electrons [Bloembergen] [39]. The absorbed energy propagates through media by free electron motion, as well as is converted to lattice vibration (or phonon) energy through electrons-lattice energy collisions. During high-power laser irradiation, the solid can be regarded as a two-temperature system, where the free electrons are heated to an effective temperature much higher than that of the lattice (or phonon). Thus, large local temperature differences arise between the electrons and lattice. Subsequently, exchange of energy from the electrons to the lattice takes place by means of a relaxation mechanism, and a heat transfer coupling coefficient between electron and phonon subsystems is considered [2, 6, and 7]. In semiconductors and dielectrics, the absorbed energy leads to the formation of electron-hole pairs and can be represented by a similar model. For polymers, however, a very different phenomenological model has to be considered, and the discussion presented in this thesis is not appropriate for those materials [Srinivasan and Miller, 1994].

The laser-solid interaction is relatively independent of other system parameters such as background gas pressure or electrode assemblies, making PLD one of the most versatile techniques. The spatial confinement of this interaction and the subsequent spatial

confinement of the ablation plume make PLD an inherently clean process compared to sputtering methods where the plasma tends to come in contact with various surfaces inside the chamber, thereby contributing to the contaminants in the film. The background gas pressure, laser energy densities, and so on, are well decoupled, giving the process a significant degree of parameter freedom [2, 6]. However, PLD process suffers from a serious disadvantage, which is the film non-uniformity caused by the condensed matter droplets in the expanding plume (Chrisey and Hubler, 1994). These droplets are formed during the laser-solid interaction process, and hence further improvement of PLD requires better understanding of laser-target interaction. For clarity and convenience, a one-dimensional model, as shown in Figure 5, has been considered in this work.

2.3. Moving Boundary Problems

Moving-boundary problems are typically called Stefan problems [3] in classical textbooks. When the boundary surface of a solid target absorbs thermal energy by conduction, convection and/or radiation, the temperature of target increases gradually. When the temperature of the boundary surface receiving energy gets to the equilibrium melting temperature T_E , the binding force that maintains its solid structure, namely latent heat L , is overcome by incoming energy, and the solid starts melting at the surface to become liquid. An interface between solid and liquid is formed and moves with time as energy is absorbed at the boundary surface, which is also referred to as the free surface.

2.3.1. An Overview of Phase-Change Problems

Both solid and liquid phases are characterized by the presence of cohesive forces that keep atoms in close proximity. In a solid the molecules vibrate around fixed equilibrium positions, while in a liquid they may "skip" between these positions. Clearly atoms in the liquid phase are more energetic than those in the solid phase. Thus before a solid can be melted, it must acquire a certain amount of energy to overcome the latent heat L that is

the quantity of energy transferred during the phase-change process. Of course, solidification of liquid requires the removal of this latent heat and the structuring of atoms into more stable lattice positions [3].

The temperature at which melting occurs depends on the kinetics of the heat removal process. If heat removal is done quite slowly compared to the time needed by the solid atoms to get energized and free themselves off the lattice structure, the melting process is called equilibrium (or more correctly, quasi-equilibrium) melting. The temperature at which a solid melts under equilibrium condition is a property of the solid, and is called the equilibrium melting temperature, T_E . In general, T_E depends on pressure. Under fixed pressure, T_E may be a fixed value characteristic of the material, which is a function of other thermodynamic variables [3].

The interface between solid and liquid, where the two phases coexist has a thickness that may vary from a few Angstroms to a few centimeters. Depending on several factors (the material itself, the rate of cooling, the temperature gradient in the liquid, surface tension etc.), interfacial microstructure may be complex. For most pure materials, during freezing or melting under ordinary or equilibrium conditions, the interface appears to be (locally) planar, of negligible thickness, and at a fixed temperature T_E . In these cases, the interface may be thought of as a "sharp front" separating solid from liquid at temperature T_E [3, 6], thus forming an isotherm.

Most thermophysical properties of a material (usually varying smoothly with temperature) undergo more or less sudden changes at T_E . Such discontinuities in thermophysical properties complicate the mathematical problems because they induce discontinuities in the coefficients of differential equations. However, the most fundamental and pronounced effects are due to changes in density. Typical density changes upon freezing or melting are in the range of 5% to 10% but can be as high as 30%. For most materials the solid is denser than the liquid, although the opposite is true for water. The density variation with

temperature induces flow by natural convection in presence of gravity, rapidly equalizing the temperature in the liquid and greatly affecting heat transfer [3, 6, and 9].

2.3.2. Assumptions

In the process of formulating the computational model, the following physical assumptions are typically taken into account so that the mathematical model is computable while maintaining the computational precision. According to these assumptions, the phase-change process involves a PCM (phase change material) with density ρ , latent heat L , melting temperature T_E , phase-wise specific heats c , and thermal conductivities k . In the most common cases, it is very reasonable for pure materials, small container and moderate temperature gradient, that heat is transferred isotropically only by conduction through both the solid and the liquid, and all other effects including possible gravitational, elastic, chemical, and electromagnetic effects are assumed to be negligible. Latent heat that is assumed to be constant is released or absorbed at the phase-change temperature, which is regarded as a fixed known temperature, a property of the material. In many situations, nucleation difficulties and supercooling effects are assumed to be absent. For interface between liquid and solid phases, interfacial thickness and structure are assumed to be locally planar and sharp at the phase-change temperature, and the surface tension and curvature at the interface are assumed to be insignificant. In order to avoid bulk movement of material, density is assumed to be same in the solid-liquid phase-change process [3, 9].

In deriving the model it is typically assumed that all the functions representing physical quantities are as smooth as required by the equations in which they appear. Of course this assumption has to be used a-priori in order to proceed with the formulation of the mathematical model, but has to be justified a-posteriori by proving that the resulting mathematical problem does indeed admit such smooth solutions [3, 9].

2.3.3. Stefan Condition

In a melting or solidification process, consistent with the assumptions above, the physical region Ω occupied by the phase change material will be subdivided into two phases, liquid and solid, separated by a sharp interface with zero thickness [3].

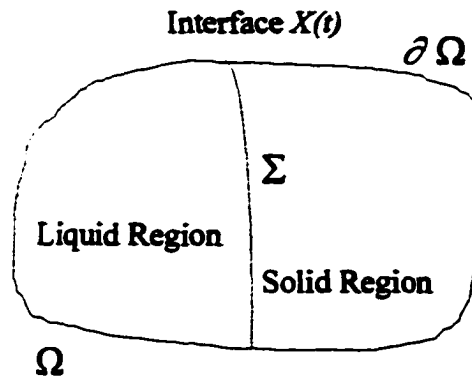


Figure 3

Interface Between Liquid and Solid Regions

At the interface $\vec{X} = \vec{X}(t)$, the following two conditions are typically satisfied under equilibrium phase change. Both of these conditions were first formulated by Stefan [Alexiades and Solomon, 1993], and commonly referred to as Stefan Conditions.

$$\lim_{\vec{x} \rightarrow \text{interface } \vec{X}(t)} T_L(\vec{x}, t) = T_E \qquad \lim_{\vec{x} \rightarrow \text{interface } \vec{X}(t)} T_S(\vec{x}, t) = T_E \qquad (1a)$$

$$\vec{x} \rightarrow \text{interface } \vec{X}(t)$$

$$\vec{x} \in \text{liquid}$$

$$\vec{x} \rightarrow \text{interface } \vec{X}(t)$$

$$\vec{x} \in \text{solid}$$

Interfacial energy balance:

$$\rho L \frac{dX}{dt} = \vec{q}_L \cdot \vec{n} - \vec{q}_S \cdot \vec{n} \qquad (1b)$$

where $\vec{q}_L \cdot \vec{n}$ and $\vec{q}_S \cdot \vec{n}$ are the heat flux normal to the moving surface in the liquid and the solid, respectively. Conservation of energy in each phase demands that a heat conduction equation be satisfied there. The latent heat released due to the interface displacement equals the net amount of heat delivered to (or from) the interface per unit area per unit time. Thus, the second condition (equation 1b) is a statement of heat balance across the interface [3]. The first condition (equation 1a) is, however, an assumption, and is only true under equilibrium phase change. In this thesis, it is only this first interfacial condition that is referred to as Stefan condition, in order to distinguish it from the more general kinetics condition, which will be presented later.

Around 1890, J. Stefan formulated the problem of solving the temperature distribution and freezing front history of a solidifying slab of water. In the last 30 years, the problem bearing his name has been extended to include such complex phenomena as the solidification of alloy systems, supercooling, melting due to Joule heating and laser irradiation which we are working with [3].

For more accurate answers for the physical phenomena, some of the above assumptions are not valid. For example, the interfacial temperature, as given by equation (1a), is not equal to equilibrium temperature, T_E , of the material under intense laser irradiation because of the influence of kinetics in phase transitions.

3. PROBLEM STATEMENT

The objective of this study is to build a one-dimensional mathematical model to simulate the heat transfer process in laser ablation of a solid slab with a finite thickness. In the process of laser irradiation, the target will undergo a transition in its physical structure from a solid phase to a liquid phase [Alexiades and Solomon, 1993] [1, 2, 3, 4, 6, 10, 11, 12, 13, and 14]. This numerical simulation also tracks the sharp moving boundary that separates liquid phase and solid phase. In the numerical model, two alternative interfacial conditions have been used: (1) Stefan condition where the interfacial temperature is assumed to be always equal to the equilibrium melting temperature, and (2) the kinetics condition where the interface is superheated [Sobol, 1995]. This study also compares the computational results that are obtained with those two alternative interfacial conditions [1, 4, 6, and 15].

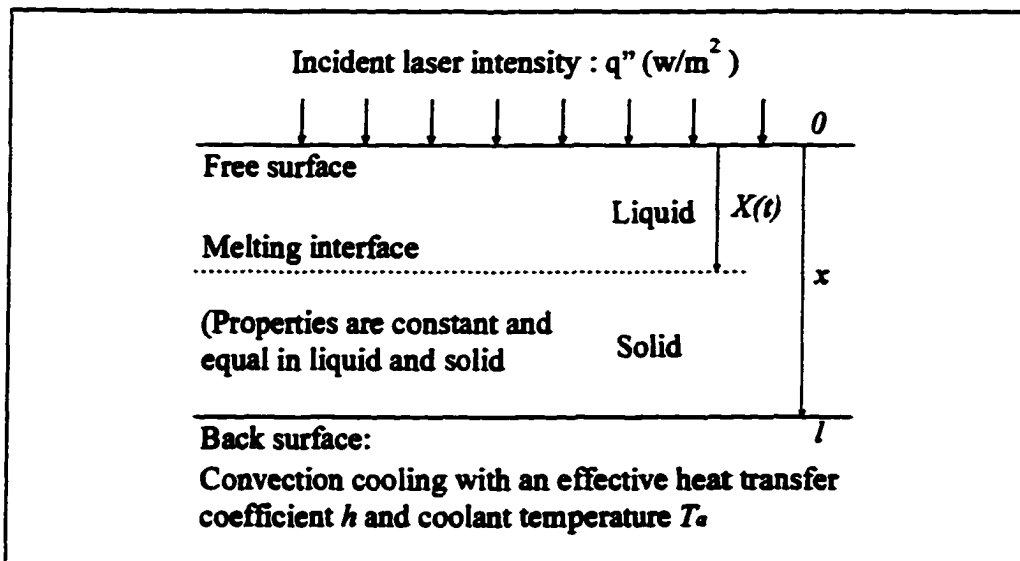


Figure 4
Geometric Parameters of the Target

Definition of the computational problem includes the governing differential equations, and boundary, interfacial and initial conditions [3, 16]. The test material for the target is iron with pure crystalline structures. The one-dimensional computational domain used in this mathematical model is presented in Figure 4 [4].

3.1. Governing Equations of Unsteady Heat Conduction

When a high-power laser irradiates a solid with a high heat flux, immediately after the irradiation starts, energy transport within the solid is governed by non-Fourier equation that accounts for the finite thermal propagation speed. The use of the non-Fourier equation removes the nonphysical phenomenon of the diffusion equation analysis that predicts instantaneous temperature disturbances at all points in the medium for a step heat flux at the boundary. It further removes the singularity of an infinite temperature at the irradiated surface as time goes to be zero. Adversely, the classical Fourier theory is based on an infinite speed of propagation of the thermal signal and indicates that a local change of temperature causes an instantaneous perturbation in the temperature at each point of the medium, even if the intervening distances are very large. Thus, the uses of theory of Fourier heat conduction leads to inaccurate temperature and heat flux profiles at high irradiation and at early times. At later times (compared to thermal relaxation time), Fourier conduction laws provide accurate predictions [16, 17, 18, 19, and 20].

Since validity of Stefan condition at the interface is the main focus of this research, and since results for very short time intervals are not the prime objectives, Fourier conduction laws have been used here as the governing equations. The computational domain can be idealized as an infinite plate with finite thickness and with no temperature gradients along the y and z coordinates. The unsteady state temperature distribution will be one dimensional and will satisfy the following diffusion equation [3, 4, 13, 16, 17, and 21]:

$$\rho c \frac{\partial T_L(x,t)}{\partial t} = k \frac{\partial T_L^2(x,t)}{\partial x^2} \quad \text{for liquid region} \quad (2)$$

$$\rho c \frac{\partial T_s(x,t)}{\partial t} = k \frac{\partial T_s^2(x,t)}{\partial x^2} \quad \text{for solid region} \quad (3)$$

Here, k is the thermal conductivity of a material and depends on the material chemical composition, physical structure, and state. Also it varies with the temperature, and mildly with the pressure, of the material. The value of thermal conductivity in a material is changed with its phases. In liquid phase, molecular force fields exert a strong influence on the energy exchange during molecular collisions. However, heat conduction in solids with crystalline structure depends on the energy transfer by molecular and lattice vibrations and free electrons. Generally, energy transfer by molecular and lattice vibrations is not as large as the energy transported by free electrons. The value of thermal conductivity for the solid is higher than that for the liquid [16, 21, and 22]. To simplify the relevant equations and calculation, the values of thermal conductivity, specific heat c and density ρ are assumed to be constant and keep unchanged in the melting process [3].

3.2. Boundary and Initial Conditions

In order to have a well-posed mathematical problem [3], the appropriate boundary and initial conditions need to be specified. In the Pulsed Laser Deposition reactor [1, 5], the target is mounted on the target holder in the chamber that is actively cooled by a cooling gas with the constant temperature T_a , which is the ambient temperature for the backside thermal convection to the coolant, and is same as the target's initial temperature T_i .

3.2.1. Imposed Flux

When laser radiation is incident on the target, the plume that is formed over the target surface absorbed a part of the laser energy, before the laser beam can reach the target. The rest of the energy is partially reflected by the boundary surface. This study does not consider either of these two losses. The laser beam over the boundary surface is simply considered as a time-invariable fixed heat source in this one-dimensional mathematical

model so that Fourier's law of heat transfer is applicable. The corresponding boundary equation is presented as [3, 16, and 18],

$$-k \frac{\partial T(0,t)}{\partial x} = q'' \quad (4)$$

where q'' is viewed as a constant surface heat flux and is fixed at the target surface.

3.2.2. Convective Flux

A convection cooling condition is considered on the back surface of the target. The conduction flux through the solid target is equal to the convection heat flux, as given by Newton's law of cooling [3, 6, and 16].

$$-k \frac{\partial T}{\partial x} T_S(l,t) = h [T_S(l,t) - T_a] \quad (5)$$

where h is the heat transfer coefficient between the back surface of the target and the ambient air. This coefficient depends on the flow conditions, thermophysical properties of heat transfer fluid and dimensions of the target boundary. In this situation, it is assumed to be a constant.

3.2.3. Interface Condition

One major characteristic of phase-change problems is that, in addition to the temperature field, the location of the interface between liquid and solid regions is also unknown. Energy balance at the interface shows that the rate of change in latent heat $\rho L X'(t)$ equals the amount by which the heat flux jumps across the interface [3].

$$\rho L \frac{dX}{dt} = -k \frac{\partial T_L(X,t)}{\partial x} + k \frac{\partial T_S(X,t)}{\partial x} \quad (6)$$

where L is the latent heat that represents the difference in thermal energy (enthalpy) levels between liquid and solid states. It is also called heat of fusion in the case of solid-liquid transformation [3].

3.2.4. Role of Kinetics in Melting

One additional question is needed at the interface in order to have a complete and well-posed mathematical formulation of the problem. The main question that needs to be answered is which physical factor determines the interfacial temperature? At low rates of phase transitions, the interface temperature is a fixed known temperature equal to the equilibrium melting temperature, T_E . The growth of a new phase is governed not only by the rate of energy input (or removal), but also by the kinetics of the processes taking place at the moving interface [Sobol, 1995] [4, 6, 15, and 23]. Depending on which of these two processes in a given case, the mathematical and computational formulations of the problem will be significantly different. This difference is the focus of this study.

Where the rate of heat input is slow and the rate of atomic transitions at the interface can keep up with the rate of heat transfer, the growth of the new phase takes place in isothermal conditions around the quasi-equilibrium temperature. In this situation, the assumption of Stefan problem, as presented in equation (1b), is reasonable. The law governing the movement of the phase boundary can be found by solving the heat conduction problem in a domain with a moving boundary on the basis of the condition that melting temperature T_m equals thermodynamic equilibrium temperature T_E [4]. As a result, the phase transition takes place in the condition that the moving interface is at a constant temperature T_E .

$$T_l(X, t) = T_s(X, t) = T_m = T_E \quad (7)$$

However, in the second situation where the rate of heat input is high and the atomic transitions at the interface can not keep up with the rate of energy transport, the kinetics

of phase transition dictates the movement of the interface and the interfacial temperature. In this case, the interfacial temperature T_m is possible to be substantially higher than the thermodynamic equilibrium temperature T_E , and the interface is said to be superheated. The rate of melting and the rate of interface motion depend on the rate at which atoms can cross the phase-separation interface, as given by [4, 23]

$$\frac{dX}{dt} = V_s \cdot \exp\left(-\frac{\Delta g_b}{k_B T_m}\right) \cdot \left(1 - \exp\left(-\frac{\Delta g_m}{k_B T_m}\right)\right) \quad (8)$$

Here, Δg_b and Δg_m are the Gibbs free energies (per molecule) corresponding to the phase transition barrier and the latent heat of phase transition, as shown schematically in Figure 5 [4, 15],

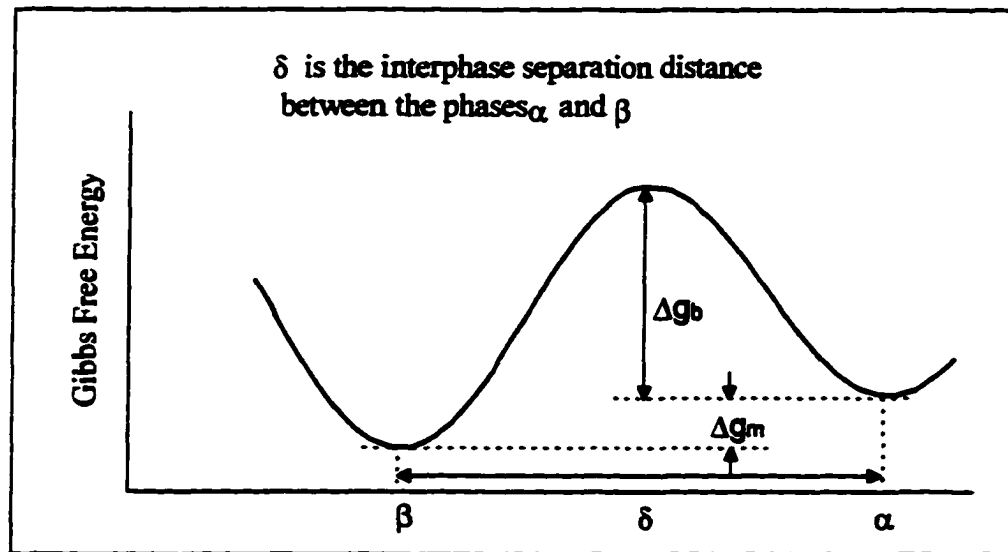


Figure 5
Schematic Variation of Free Energy in Melting

V_s is a characteristic speed of the material that can be approximated by the speed of sound, k_B is the Boltzman constant and T_m is the interface temperature which is higher than the thermodynamic equilibrium temperature (here, T_E) in order to cause any phase

transition. It should be noted that Δg_b can also be considered as the activation Gibbs free energy necessary for the diffusion/migration of the atoms/particles across the interface. Under small departure from the equilibrium condition, the Gibbs free energy for phase transformation can be approximated as [4, 6, and 15]

$$\frac{\Delta g_m}{T_m} \approx \frac{\Delta h_m (T_m - T_E)}{T_E^2} \quad (9)$$

where Δh_m is the latent enthalpy of melting per molecule and is equal to LM/N_A , M is the molecular weight, and N_A is Avogadro's number.

3.3. Mathematical Model

The entire process of heat transfer with phase-change in a target with a finite thickness, l , can be divided into two stages. The first stage is the heat transfer before melting. In this stage, the equation of unsteady heat conduction in the solid without any phase transition is to be solved until the free surface of the target gets to the equilibrium melting temperature. The second stage is heat transfer with phase transition, and starts after the first stage ends.

3.3.1. Heat Transfer before Phase Transition

$$T = T_s(x, t) \quad \text{for } 0 \leq x \leq l, \quad t > 0$$

$$\rho c \frac{\partial T_s(x, t)}{\partial t} = k \frac{\partial^2 T_s(x, t)}{\partial x^2} \quad (10)$$

$$T_s(x, 0) = T_i \quad (11)$$

$$-k \frac{\partial T_s(0, t)}{\partial x} = q'' \quad (12)$$

$$-k \frac{\partial T_s(l, t)}{\partial x} = h [T_s(l, t) - T_a] \quad (13)$$

3.3.2. Heat transfer with Phase Transition

$$\begin{aligned}
 X &= X(t) & 0 \leq X \leq l \\
 T &= T_L(x, t) & \text{for } 0 < x < X(t), t > 0 \text{ (liquid region)} \\
 T &= T_S(x, t) & \text{for } X(t) < x < l, t > 0 \text{ (solid region)}
 \end{aligned}$$

$$\rho c \frac{\partial T_L(x, t)}{\partial t} = k \frac{\partial T_L^2(x, t)}{\partial x^2} \quad (14)$$

$$\rho c \frac{\partial T_S(x, t)}{\partial t} = k \frac{\partial T_S^2(x, t)}{\partial x^2} \quad (15)$$

$$T_L(x, 0) = T_m \quad (16)$$

$$X(0) = 0 \quad (17)$$

$$-k \frac{\partial T_L(0, t)}{\partial x} = q'' \quad (18)$$

$$-k \frac{\partial T_S(l, t)}{\partial x} = h [T_S(l, t) - T_a] \quad (19)$$

$$\rho L \frac{dX}{dt} = -k \frac{\partial T_L(X, t)}{\partial x} + k \frac{\partial T_S(X, t)}{\partial x} \quad (20)$$

$$T_i(X, t) = T_s(X, t) \quad (21)$$

$$T_i(X, t) = T_s(X, t) = T_m = T_E \quad \text{in Model I} \quad (22-1)$$

$$\frac{dX}{dt} = V_s \cdot \exp\left(-\frac{\Delta g_b}{k_B T_m}\right) \cdot \left(1 - \exp\left(-\frac{\Delta g_m}{k_B T_m}\right)\right) \quad \text{in Model II} \quad (22-2)$$

The time when the target starts melting is preset at zero for the computational convenience. As mentioned earlier, two mathematical models are developed here. The first model, which is based on Stefan condition at the interface, is made up of the equation (10) -(21) plus the equation (22-1) without consideration of the influence of kinetics in phase transition. The second one, is made up of the equation (10) -(21) plus the equation (22-2) with consideration of the influence of kinetics in phase transition.

3.4. Further Consideration

In order to find more exact answers, more complicated physical processes that take place as a result of laser-target interactions in the target region of a PLD reactor, have to be considered [1]. Some of these processes may even be different from one case to another depending on whether the target is a metal, or a semiconductor, or a superconductor, or a polymer, and depending on the pulse duration and peak power density. It should be noted that in a PLD process, there may be two different moving interfaces: the ablation interface and the melting interface, as shown in Figure 6 [1].

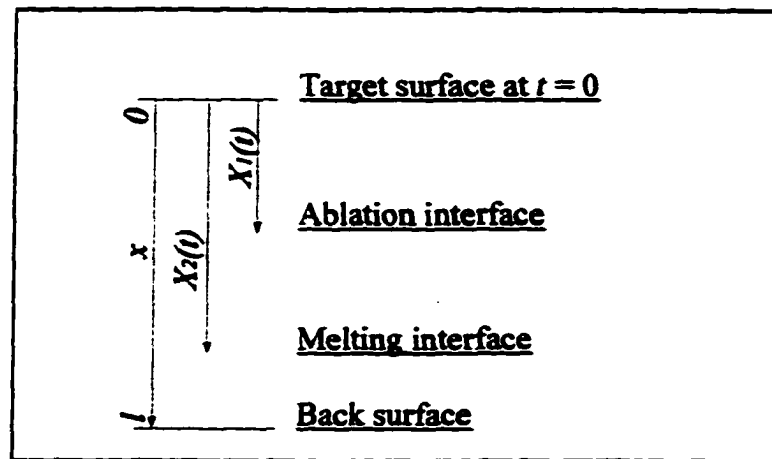


Figure 6

Geometric Parameters of the Target in Laser Ablation

Some of these physical processes that may have to be considered in a complete mathematical model for laser-solid interaction as applicable to PLD are described below.

1. Bulk absorption of laser intensity: The laser energy that is incident on the target surface is partially reflected and the rest is absorbed over a depth into the target, not just at the surface. As a result of this volume absorption, the laser intensity decreases with increasing depth into the target and is given by [1, 24]

$$I(x,t) = I_s(t) \cdot \{1 - R[T(X_1(t))]\} \cdot \exp\left[-\int_{X_1(t)}^x \alpha_l(T(x')) dx'\right] \quad (23)$$

where $I_s(t)$ is the time-dependent laser intensity arriving at the surface of the target. R is the surface reflectivity, which is a function of the boundary temperature that is measured at the ablation interface that is located at $x = X_1(t)$ as shown in Figure 6. α_l is the linear absorption coefficient for volume absorption, and is also a function of local temperature and hence a function of the spatial coordinate. As a result, the integral on the right hand side of the equation (23) needs not be a trivial one. The bulk absorption affects the energy balance inside the target as the laser energy is converted into internal energy and hence appears as an energy source term in the appropriate energy conservation equation. This source term per unit time per unit volume is given by [1, 13]

$$\frac{dI(x,t)}{dx} = -I_s(t) \cdot [1 - R(T(X_1(t)))] \cdot \alpha_l(T(x)) \cdot \exp\left[-\int_{X_1(t)}^x \alpha_l(T(x')) dx'\right] \quad (24)$$

2. Shape of laser pulse: The temporal variation of laser intensity, $I_s(t)$, within each pulse is not rectangular as assumed in numerical modeling. The temporal shape of a typical excimer laser pulse is shown in Figure 7 [1]. The actual pulse shape should be considered in any numerical model for industrial usage.

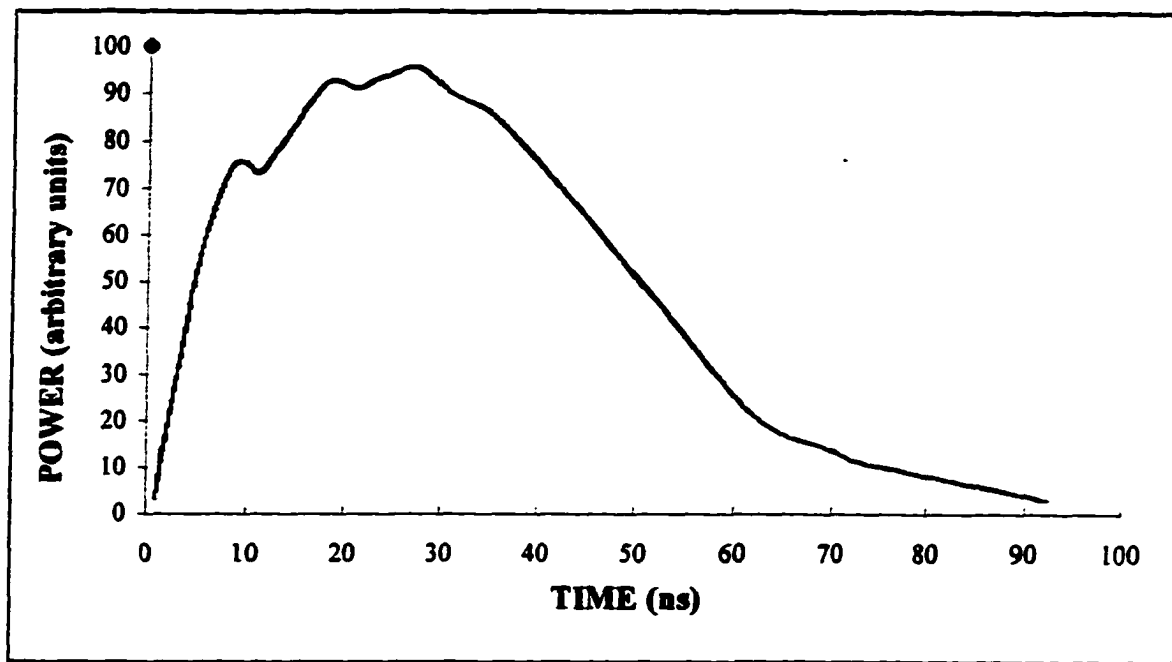


Figure 7
A Typical Excimer Laser Pulse

3. Non-equilibrium heat conduction in solids: The primary dissipative interaction of the incident laser beam with a solid is the absorption of photons by electrons, although under certain conditions, optical phonons may also participate in the absorption process. In metals, the dominant absorption mechanism is through the free-free transitions of the conduction electrons, whereas in the semiconductors, it is through the creation of electron-hole pairs and/or excitons. However, in semiconductors, excitons are converted immediately to electron-hole pairs, and hence a separate energy balance equation for excitons is not necessary. The same thing cannot be said about insulators with small dielectric constants for which a separate energy balance equation is necessary. The electrons in their excited states transfer some of their energy to other electrons and to lattice vibrational modes or phonons. At higher laser power density and short pulse duration typical of PLD, the energy in the electronic and phonon modes do not reach an equilibrium, and hence separate energy balance equations have to be considered. Here, the two-temperature model is used where the energy balance

equations for the electrons and the phonons are expressed in terms of the electron temperature, T_e , and the phonon temperature, T_p , respectively. The electron and phonon temperatures are measures of their respective energies such that the temperature would be equal to the conventional temperature of the solid in case of equilibrium between electron and phonon energies. The net rate of energy transfer, per unit volume, from the electrons to the phonons is given by [1, 13],

$$G(T_e, T_p) \cdot (T_e - T_p) \cong \frac{m_e^2 U^2 (k_B T_D)^5}{16 \pi^3 \hbar^7 \rho_s V_s^4} \cdot \frac{T_e}{T_D} \cdot \left[D\left(\frac{T_D}{T_e}\right) - \frac{T_p}{T_e} D\left(\frac{T_D}{T_p}\right) \right] \cdot (T_e - T_p) \quad (25)$$

$$D(y) \cong \frac{4}{y^4} \int_0^y \left(\frac{x^4}{e^x - 1} dx \right) \quad (26)$$

where, m_e is the electron mass, T_D the Debye temperature of the material, k_B the Boltzman constant, \hbar the angular Planck constant, V_s , the speed of sound in the solid and U a term which is affected by the electron-phonon collision time. The coefficient G that is defined in equation (25) and that is a function of the electron and the phonon temperature, is called the electron-phonon coupling coefficient.

- I. Hyperbolic conduction: For short pulse duration and high laser power density, thermal relaxation time, τ_r , for diffusional transport of heat in the solid may not be negligible compared to the pulse duration, and in that case, Fourier heat conduction equation should be replaced by the hyperbolic conduction equation [1, 13]:

$$q + \tau_r \frac{\partial q}{\partial t} = -k \frac{\partial T}{\partial x} \quad (27)$$

where q is the heat flux (in the x -direction). It should be noted that typical values of the relaxation time are 10 seconds for biological tissues and porous materials at room temperatures to 1 nano-second for superconductors at cryogenic temperatures to 1 pico-second or less for metals and semiconductors at room temperatures. Hence,

whether equation (27) should be considered in the model depends on the specific application and the pulse width [1].

II. Kinetics of vaporization: Under the assumption that the vapor pressure at the surface is equal to the saturated vapor pressure, P_{sat} , at the free surface temperature and that the flow normally outward from the surface is described by a one-sided Maxwell-Boltzman distribution at the free surface temperature, and with the help of the Clausius-Clapeyron equation for P_{sat} , the mass flux due to vaporization can be expressed as [1]:

$$\rho V_v = \rho \frac{P_0}{n_c} \exp\left(\frac{-\Delta H_v}{k_B T_m}\right) / \sqrt{2\pi m k_B T_m} \quad (28)$$

where P_0 is the reference pressure used in the Clausius-Clapeyron equation, ΔH_v is the latent heat of vaporization per molecule, m is the mass of the ejected particles (molecules), T_m is the free surface or ablation interface (Figure 6) temperature, k_B is the Boltzman constant, and n_c is the number density of molecules in the condensed phase. Equation (28) which is also referred to as the Hertz-Langmuir-Knudsen formula can be rearranged as [1]

$$\frac{dX_1}{dt} \equiv V_v = \frac{P_0}{n_c} \exp\left(\frac{-\Delta H_v}{k_B T_m}\right) / \sqrt{2\pi m k_B T_m} \quad (29)$$

where $X_1(t)$ indicates the location of the free surface or the ablation interface, shown in Figure 6.

Besides those processes that are described above, there are other processes that could be present in the target region of a PLD reactor. Some of these other processes are listed below so that they can be included in the future mathematical models [1]:

1. **Absorption by optical phonons:** In the infrared region, the incident photons may be absorbed by optical phonons. This is particularly important for organic polymers.
2. **Absorption of incident laser energy in organic polymers:** Organic polymers interact with photons in ways that are unlike the pathways by which metals and other inorganic materials interact with photons, hence the models presented in this paper have to be modified in order for simulation of laser ablation of polymers.
3. **Interaction with the plasma region:** The plasma region influences the processes in the target region in multiple ways. For example, as a result of the absorption of a fraction of the laser energy by the plasma, the laser intensity at the free surface, $I_s(t)$, is not the same as the originating laser intensity. Or, the behavior of the plasma dictates the boundary condition for the target region at the free surface.
4. **Some non-metals behave as metals in the molten state, and this behavior should also be included in the computational model.**
5. **There can be fluid motion in the molten region, induced by buoyancy and Marangoni forces.**

Although the complicated physical processes are not included in this thesis, they have to be considered in any future mathematical model for the complete PLD process.

4. DISCRETIZATION ALGORITHM

For scientific and engineering problems involving complicated physical processes, boundary conditions and material properties, it is generally impossible to obtain analytical mathematical solutions, which generally require the solution of partial differential equations. The rapid development of computers has completely revolutionized research and practice in each scientific and engineering field. Computerized methods provide reasonable alternate solutions to these complex mathematical problems. Through numerical simulation of physical processes, the computational methods working with computers yield approximate values of the unknowns at a finite numbers of discrete points in the domain. This is the approach that has been taken in this study [3, 25].

In any computational method, the governing differential equations that define a given problem mathematically have to be replaced by a system of algebraic equations through a process called discretization. The process of discretization is a crucial step in solving a mathematical problem numerically since this process determines, to a large extent, accuracy, stability, and even feasibility of a numerical solution. For a complete numerical solution of a problem, boundary and initial conditions that have been specified for the problem must be discretized as well. Once a system of algebraic equations is obtained by discretization, this system has to be solved in order to obtain the numerical solutions.

As a result of the discretization, the final numerical solution is obtained either as a collection of the values of the unknown field variable(s) at a finite number of discrete points in space and/or time, or as functions of finite complexity of space and/or time. The number of those discrete points or the order or complexity of those functions is commonly referred to as the order of discretization. Typically, more is the order of discretization,

more accurate is the solution, but at the same time more time-consuming and expensive is the computation. Hence a compromise has to be made always in any numerical solution, and the key question that needs to be answered is: what should be the order of discretization so that numerical solutions are accurate enough, yet computations can be performed within the given limited time and resources.

4.1. Some Basic Aspects of Numerical Computations

In computational methods for solving problems, as mentioned earlier, governing equations are always reduced to a set of algebraic equations. All methods for obtaining a solution for a system of algebraic equations can be categorized into two groups: direct methods and iterative methods. Direct methods are typically suitable for special and “well-behaved” system of equations, whereas iterative methods are the most widely used ones in numerical computation. Choice of a method to solve the system of discretized algebraic equations determines the accuracy and stability of the numerical solutions. The solution method or algorithm should be chosen so as to minimize the error accompanying the numerical method. The stability of each method should also be considered in selection of numerical methods because lack of stability gives a solution that may exhibit non-physical or non-real oscillations in time or space [26].

All discretization methods call for some form of iterations in order to obtain more accurate numerical solutions. These iterations can be of two types. First, the numerical solutions have to be obtained with larger and larger order of discretization until the accuracy of the solutions obtained is acceptable. This type of iterations is called “grid convergence check”. Second, some discretization methods, specially some methods to discretize temporal derivatives, require calculation (or prediction) of approximate solutions to be followed by repeated refinement (or correction) of those solutions until solutions with acceptable accuracy is obtained. These iterations are called prediction-correction iterations and the corresponding discretization methods are called predictor-corrector methods [26].

4.1.1. Iterative Methods

The concept of iteration is one of the most important concepts in computational methods for solving problems. In general, numerical solutions are often formulated in terms of an infinite process of repeated iterations. A particular iterate is determined according to some formula from the previous iterates [27, 28]. Iterative formulations of problems often suggest useful computational solutions. However, two problems should be figured out for the accuracy of solution and validity of computational tools. The first one is the "stopping criteria": which iterate should be accepted as the desired approximation so that the process of repeated iterations can be stopped. The second problem that arises in connection with iterative processes is: does that process actually converge?, or does it converge to the desired quantity? In some cases, non-convergence may be discovered by actually computing iterates. In more complicated iterations, it may be expensive to compute even "one" iterate. For this reason, it is necessary to predict in advance whether the iterates will be likely to converge.

4.1.2. Numerical Errors

Any numerical solutions should be considered as a mere approximation to the exact solution, and the difference between the two is called numerical error. Numerical errors can be separated into two groups [26, 29]:

1. **Round-off errors:** Errors caused by finiteness of number representations in the computer are called round-off error that occurs when data must be rounded to a certain number of digits before they can be accepted by the computer. In order to evaluate the effectiveness of any numerical algorithm, it is necessary to examine the effect of rounding errors. Many algorithms that would work quite well if there were no rounding errors fail completely in actual use because of them. Sometimes, in some

situations, there are algorithms that work effectively only because round-off errors perturb the results just enough to avoid certain anomalies [26].

2. **Truncation errors:** Errors caused by the finiteness of resources available for solving the problem are called truncation errors. The process of approximating functions by appropriate rational functions introduces this kind of error, as do the discretization processes. Also, the error caused by stopping an iteration after finitely many steps is of this type. In most cases, the truncation error is introduced before the numerical computation ever begins. When truncation error is caused by a discretization process, the smaller this error is, the more complicated is the resulting numerical process. Furthermore, the amount of round-off error is directly proportional to the complexity of the computation. Thus, in many cases, reducing the truncation error causes the round-off error to increase [26].

4.1.3. Stability of Numerical Methods

Many numerical methods that have been proposed for solving certain problems would work quite well if exact arithmetic were used. Unfortunately, numerical errors can never be avoided in a computational method, and hence the appropriate question is whether a numerical method is stable. If small perturbations caused by ever-present numerical errors get amplified and produce large errors in the numerical solutions, the corresponding numerical method or algorithm is called an unstable one.

Many numerical methods, including the ones that this study is applying, determine a sequence of values, each of which is obtained from previously computed values. For an unstable method, small errors in the earlier values are grossly magnified by later computations, and the final results are quite inaccurate. Also, in some situations, errors made in early stages of a computation may have a strong effect on later stages of the numerical solution of differential equations [26].

4.2. Computational Methods

Not all kinds of numerical methods are amenable to any given numerical method. Neither are all kinds of numerical problems equally effective for any given mathematical problems. Analyzing the effectiveness of numerical methods for any given mathematical problems is essential for better computational simulation [26]. The following discussion is focused on different approaches for the discretization of spatial derivatives in governing differential equations: finite-difference, finite-element, and spectral methods.

4.2.1. Finite-Difference

Of all numerical methods available, the finite-difference methods are the most frequently used. The main idea in finite-difference methods is to replace derivatives with linear combinations of discrete function values at two or more neighboring points [25, 30]. The problem domain is “discretized” so that the dependent variables are considered to exist only at finite number of discrete points [17, 20]. The principle of Taylor series expansion plays a very important role in the formulation and error analysis of finite-difference discretizations [22, 25, and 30].

Finite-difference methods have the virtue of simplicity and they account for a large proportion of the numerical methods actually used in applications. However, it is never possible to refine the mesh size to zero, and the calculation must be made on a finite grid of discrete points. There are discretization errors that are caused by the truncation error in the difference representation of the partial differential equations (PDE) plus any error induced by treatment of boundary conditions in the numerical solution [25].

4.2.2. Finite-Element and Its Comparison with Finite-Difference

In finite-element method, the whole space or computational domain is divided into a finite number of regions or elements. Instead of solving the problem for the entire body in one operation, the finite-element method formulates the equations for each finite-element and combines them to obtain the solution of the whole body [31, 32]. The underlying principle of the finite-element method is its ability to easily solve problems described by complex boundary shapes [25]. This method was initially developed to calculate stress in irregularly shaped objects and analyze structural problems in aircraft. Since its inception, the finite-element method has been found to be equally effective in nonstructural problems, particularly those in heat transfer and fluid dynamics [25, 31]. Conventional finite-difference methods are based on the assumption that truncated Taylor series expansions of the spatial derivatives yield adequate approximations to differential equations. Finite-difference algorithms display good accuracy in the limit as $\Delta x \rightarrow 0$ [25]. No such assumption is implied in the finite-element method; finite-element algorithms are based on finite Δx . This conceptual difference helps to explain the general superiority of the finite-element method over finite-difference method on coarse grids; however, as $\Delta x \rightarrow 0$, the finite-difference method also becomes more accurate [25], and differences between the two methods disappear.

In the finite-element method, reduction of the governing partial differential equations to a finite system of ordinary differential equations with time-derivatives is typically performed using either the Raleigh-Fitz method or the method of weighted residuals (MWR) [22, 25, and 27].

Like many analytical methods, the finite-element method is based on the series expansion of the unknown functions themselves. In a typical series expansion, an infinite number of global basis functions span the entire domain [25]. However, in the finite-element method,

only a finite number of basis functions that are local in nature (nonzero over only a small segment of the domain) are employed.

4.2.3. MWR and Its Spectral Methods

Spectral methods have become increasingly popular in recent years, especially since the development of fast transform methods. It may be viewed as an extreme development of the class of discretization schemes for differential equations known generically as MWR [27, 28, and 29]. The key elements of the MWR are:

1. A set of trial functions (also called the expansion or approximating functions) is used as the basis functions for a truncated series expansion of the solution [27, 28].
2. A set of test functions (also known as weight functions) is used to ensure that the differential equation is satisfied as closely as possible by the truncated series expansion. This is achieved by minimizing the residual, i.e., the error in the differential equation produced by using the truncated expansion instead of the exact solution, with respect to a suitable norm. An equivalent requirement is that the residual satisfies a suitable orthogonality condition with respect to each of the test functions [27, 28].

The choice of trial functions is one of the features that distinguish spectral methods from finite-element methods and finite-difference methods. The trial functions for spectral methods are infinitely differentiable global functions. On the contrary, in the case of finite-element methods, the domain is divided into small elements, and a trial function is specified in each element. The trial functions are thus local in character, and well suited for handling complex geometries. The finite-difference trial functions are likewise local [27].

The choice of test functions distinguishes among the three most commonly used spectral schemes, namely, the Galerkin, collocation, and tau versions. In the Galerkin approach,

the test functions are the same as the trial functions. They are infinitely smooth functions that individually satisfy the boundary conditions. The differential equation is enforced by requiring that the integral of the residual times each test function be zero. In the collocation approach, the test functions are translated Dirac delta functions centered at special, so-called collocation points. This approach requires the differential equation to be satisfied exactly at the collocation points. Spectral tau methods are similar to Galerkin methods. However, none of the test functions needs to satisfy the boundary conditions. Hence, a supplementary set of equations should be used to apply the boundary conditions [27].

The collocation approach is the simplest of these three spectral methods. The Galerkin approach is an elegant example of the method of weighted residuals since the trial functions and the test functions are the same and the physical problem can be discretized in terms of a variational principle. Finite-element methods customarily use this approach. The tau approach is a modification of the Galerkin method that is applicable to problems with non-periodic boundary conditions. It may be viewed as a special case of the so-called Petrov-Galerkin method. The interpretation of spectral methods as MWR methods has proven very successful in the theoretical investigation. As a matter of fact, it has opened the road to the use of techniques of functional analysis to handle complex problems and to obtain the most accurate results [22, 27, and 28].

4.3. Chebyshev Collocation Method

On the basis of the following comparisons with finite-difference and finite-element methods, the spectral method using Chebyshev polynomials as the trial functions is selected for the current problem [13].

4.3.1. Selection of Spectral method

- 1. For an infinitely smooth (i.e. differentiable) solution to be considered, the properly designed spectral method may make errors go to zero faster than any finite power of the number of expansion terms of retained modes. In contrast, finite-difference and finite-element methods yield finite-order rates of convergence. The spectral methods can achieve high accuracy with little more resolution than is required to achieve moderate accuracy [13, 27, and 28].**
- 2. Since spectral methods typically require a factor of 2-5 fewer degrees of freedom in each space direction to achieve moderate accuracy, the spectral computations can be considerably more effective [13, 27, and 28].**
- 3. The mathematical features of spectral methods follow very closely the partial differential equation being solved [27]. Thus, the boundary conditions imposed on spectral approximations are normally the same as those imposed on the differential equation. In contrast, finite-difference methods of higher order than the differential equation require additional "boundary conditions." Many of the complications of finite-order finite-difference methods disappear with the infinite-order-accurate spectral methods.**
- 4. Relatively few grid points are needed for spectral discretization because of the global nature of the interpolating functions [13, 27, and 28].**

In spectral methods, Galerkin spectral method produces very accurate solutions with relatively few unknown coefficients in the approximate solution [27]. However, when nonlinear terms are involved, the evaluation of products of approximate solutions becomes very time-consuming. This lack of economy motivates the use of collocation instead of a Galerkin formulation. Collocation facilitates seeking the solution in terms of nodal

unknowns, like the finite-difference and finite-element methods, rather than in terms of the unknown coefficients in the approximate solution. The explicit use of nodal unknowns also permits boundary conditions to be incorporated more efficiently than does the Galerkin spectral method [28]. Consequently, the collocation spectral method is used in this study.

4.3.2. Description of Chebyshev Collocation Method

Governing differential equations with first order time derivatives, like the ones for the current problem as described in equations (10), (14), or (15), can be written as [27, 28, and 34]

$$\frac{\partial u}{\partial t} = M(u) \quad (30)$$

where $u(x,t)$ is the unknown function that is to be solved and $M(u)$ is an operator that contains all the spatial derivatives of u . In MWR, the approximate solution is represented as [16, 22, 27, 28, and 29]

$$u^N(x,t) = \sum_{k=-N/2}^{N/2} a_k(t) \phi_k(x) \quad (31)$$

where ϕ_k are the trial functions, and a_k are the expansion coefficients. Generally, the approximate solution u^N is different from the exact solution u , and larger is the order of discretization, N , smaller is the difference or error. Thus, the residual [27, 35]

$$\frac{\partial u^N}{\partial t} - M(u^N)$$

will not vanish everywhere. The MWR approximation results by demanding [22, 27]

$$\int_{\Omega} \left[\frac{\partial u^N}{\partial t} - M(u^N) \right] \psi_k(x) dx = 0 \quad (32)$$

for $k = 0, \dots, N$, where test function ψ_k determine the weights of the residual, and Ω represents the limits of the computational domain.

In Chebyshev Collocation Method, the Chebyshev polynomials are chosen as trial functions [5, 29, and 33]

$$\phi_k(x) = T_k(x) \quad K = 0, 1, \dots, N, \quad (33)$$

the test functions are the shifted Dirac delta-functions [22, 27]

$$\psi_j(x) = \delta(x - x_j) \quad j = 0, 1, \dots, N, \quad (34)$$

where x_j are the collocation points in Ω and will be defined later, and computational domain Ω is $[-1, 1]$.

The Chebyshev polynomials are cosine functions after a change of independent variable [36], and hence the function values are always limited between ± 1 . This property is the origin of the widespread popularity of Chebyshev polynomials in the numerical approximation of non-periodic boundary value problems. The Chebyshev polynomial of degree n , $T_n(x)$, is defined by [27, 36]

$$T_n(\cos \theta) = \cos n\theta \quad \text{for } n = 0, 1, \dots \quad (35)$$

$$\theta = \arccos x, \quad x \in [-1, 1], T_n \in [-1, 1] \quad (36)$$

The Chebyshev polynomial can be expanded in power series as [27]

$$T_k(x) = \frac{k}{2} \sum_{l=0}^{\lfloor k/2 \rfloor} (-1)^l \frac{(k-l-1)!}{l!(k-2l)!} (2x)^{k-2l} \quad (37)$$

where $[k / 2]$ denotes the integral part of $k / 2$. Moreover, the trigonometric relation $\cos(k+1)\theta + \cos(k-1)\theta = 2\cos\theta\cos k\theta$ gives the recurrence relation [27]

$$\begin{aligned} T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x) \quad \text{with} \\ T_0(x) &\equiv 1 \\ T_1(x) &\equiv x \end{aligned} \quad (38)$$

As a result of the choicer of the test functions, the standard MWR condition, as given by equation (32), reduces to the requirement that governing equations be satisfied at each of the collocation point x_j [27]:

$$\left. \frac{\partial u^N}{\partial t} - M(u^N) \right|_{x=x_j} = 0 \quad (39)$$

As is typically the case for ended two-ended boundary value problems where boundary conditions are specified at both boundaries of the computational domain, Gauss-Lobatto points are chosen here as the collocation points as follows [27, 29, and 33]

$$x_j = \cos \frac{\pi(N-j)}{N}, \quad (40)$$

then, values of the trial functions at the collocation points are given by [27, 29, and 33]

$$\phi_k(x_j) = \cos \frac{\pi(N-j)k}{N} \quad (41)$$

One of the biggest advantage of collocation method over Galerkin or tau method is that the expansion coefficients can be expressed in terms of the nodal values, $u_j^N(t)$, of the approximate solution [27, 28], as given by

$$a_k(t) = \frac{2}{N\bar{c}_k} \sum_{j=0}^N \bar{c}_k^{-1} u_j \cos \frac{\pi(N-j)k}{N} \quad K = 0, 1, \dots, N, \quad (42)$$

$$\text{where } \bar{c}_k = \begin{cases} 2 & j=0 \text{ or } N \\ 1 & 1 \leq j \leq N-1 \end{cases}$$

It should be noted that the superscript N in $u_j^N(t)$ indicates that those nodal values are for the approximate solution $u_j^N(x, t)$, where larger is the order of discretization N is, better is the approximation.

The analytic first and second order derivatives are expressed by [27, 29, and 33]

$$\frac{\partial u^N}{\partial x} = \sum_{k=0}^N a_k^{(1)}(t) T_k(x) \quad (43)$$

$$\frac{\partial^2 u^N}{\partial x^2} = \sum_{k=0}^N a_k^{(2)}(t) T_k(x) \quad (44)$$

$$\text{where } a_{N+1}^{(1)}(t) = 0$$

$$a_N^{(1)}(t) = 0$$

$$\bar{c}_k a_k^{(1)}(t) = a_{k+2}^{(1)}(t) + 2(K+1)a_k(t) \quad K = N-1, N-2, \dots, 0,$$

$$\text{and } a_{N+1}^{(2)}(t) = 0$$

$$a_N^{(2)}(t) = 0$$

$$\bar{c}_k a_k^{(2)}(t) = a_{k+2}^{(2)}(t) + 2(K+1)a_k^{(1)}(t) \quad K = N-1, N-2, \dots, 0,$$

Once Chebyshev collocation method is applied to a governing PDE, such equations (10), (14) or (15), a set of $(N+1)$ ODEs (ordinary differential equations) are obtained by combining equations (31), (39), (41) and (44). These ODEs involve $du_j^N(t)/dt$, first-order time derivatives of the nodal values, $u_j^N(t)$, and are of the form

$$\frac{du_j^N}{dt} = f(u_j^N(t), t) \quad K = 0, 1, \dots, N-1, \quad (45)$$

It should be noted that two additional equations, corresponding to $K = 0$ and N , are obtained from the boundary conditions.

4.4. Discretization of Time Derivatives

Among the numerical methods for discretization of first-order time derivatives, linear multistep methods constitute one of the most important families [16]. According to Cauchy's theorem, for an initial-value problem of the type [29, 33]

$$\frac{dv(t)}{dt} = f(v(t), t). \quad (46)$$

If $f(v, t)$ is continuous with respect to t and uniformly Lipschitz continuous with respect to v for $t \in [0, T]$, and $T > 0$, then there exists a unique differentiable solution $v(t)$ that satisfies equation (46). However, the existence and uniqueness of a numerical solution of equation (47) depend on the numerical method or algorithm, and is not guaranteed by the above theorem. The objective of the numerical method is to construct a sequence of values $v^{(0)}, v^{(1)}, \dots$ such that

$$v^{(n)} \approx v(t_n), \quad n > 0. \quad (47)$$

where $t_n = n \cdot \Delta t$, and $\Delta t > 0$ is the time interval that defines the size of the temporal discretization. A linear multistep method is a formula for calculating each new value $v^{(n+1)}$ with the help of the values from the previous s time steps $v^{(n+1-s)}, \dots, v^{(n)}$ and $f^{(n+1-s)}, \dots, f^{(n)}$ where

$$f^{(n)} = f(v^{(n)}, t_n). \quad (48)$$

Thus, the general expression for a s -step linear multistep formula for calculation of $v^{(n+1)}$ is given by [29],

$$\sum_{j=0}^s \alpha_j f^{(n+1-j)} = \Delta t \cdot \sum_{j=0}^s \beta_j f^{(n+1-j)}, \quad (49)$$

for some constants $\{\alpha_j\}$ and $\{\beta_j\}$ with $\alpha_0 = 1$. Typically, many of the constants in either $\{\alpha_j\}$ or $\{\beta_j\}$ series are taken to be zero so that the rest of those multistep coefficients or constants can be uniquely determined so as to provide the maximum possible accuracy. Moreover, If $\beta_0 = 0$, the multistep formula is explicit, while if $\beta_0 \neq 0$, the formula is implicit. The simplest linear multistep method is an explicit one-step method (Euler formula) and an implicit one-step method backward Euler formula, defined by [29]

$$v^{(n+1)} = v^{(n)} + \Delta t \cdot f^{(n)}, \quad (50)$$

$$v^{(n+1)} = v^{(n)} + \Delta t \cdot f^{(n+1)}. \quad (51)$$

4.4.1 Adams and Backwards Differentiation formulas

The oldest linear multistep formulas are Adams formulas, the s -step Adams-Bashforth (ABs) and Adams-Moulton (AMs) are the optimal explicit and the optimal implicit formulas of this type [29, 32], respectively:

$$\text{AMs: } v^{(n+1)} - v^{(n)} = \Delta t \cdot \sum_{j=0}^s \beta_j f^{(n+1-j)} \quad (52a)$$

$$\text{ABs: } v^{(n+1)} - v^{(n)} = \Delta t \cdot \sum_{j=1}^s \beta_j f^{(n+1-j)} \quad (52b)$$

where the coefficients $\{\beta_j\}$ are chosen to maximize the order of accuracy, and in both cases, this choice turns to be unique. If we consider the values $f^{(n+1-j)}, \dots, f^{(n)}$ (ABs) or $f^{(n+1-j)}, \dots, f^{(n+1)}$ (AMs) as discrete samples of a continuous function $f(t) = f(u(t), t)$ that we want to integrate [29],

$$u_j^N(t_{n+1}) - u_j^N(t_n) = \int_{t_n}^{t_{n+1}} \frac{du_j^N(t)}{dt} dt = \int_{t_n}^{t_{n+1}} f(t) dt, \quad (53)$$

in order to obtain $u(t_{n+1})$. However, $f(t)$ cannot be integrated analytically as it is not a known function, and is approximated by $q(t)$, which is the unique polynomial of degree at most $s-1$ (Adams-Bashforth) or (Adams-Moulton) that interpolates the numerically obtained data $\{f^{(n+1-j)}\}$. That is, the multistep formula for Abs or AMs as given by equation (52) are obtained from [29]

$$v^{(n+1)} - v^{(n)} = \int_{t_n}^{t_{n+1}} q(t) dt \quad (54)$$

Since the integral is a linear function of the data $\{f^{(n+1-j)}\}$, the coefficients $\{\beta_j\}$ can be computed once and for all, and tabulated [Trefethen, 1989] as presented in Tables 1 and 2 [29, 34].

Table 1
Coefficients $\{\beta_j\}$ of Adams-Bashforth Formulas

number of steps s	order p	β_0	β_1	β_2	β_3	β_4
1	1	0	1			
2	2	0	$\frac{3}{2}$	$-\frac{1}{2}$		
3	3	0	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	
4	4	0	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$

Beside Adams formulas, the s -step backwards differentiation (BDs) formulas form the most important family of linear multistep formulas. Like AMs, BDs is the s -step optimal implicit formula with [29]:

$$\text{BDs: } v^{(n+1)} + \sum_{j=1}^s \alpha_j v^{(n)} = \Delta t \cdot \beta_0 f^{(n+1)} \quad (55)$$

Table 2
Coefficients $\{\beta_j\}$ of Adams-Moulton Formulas

number of steps s	order p	β_0	β_1	β_2	β_3	β_4
0	1	1				
1	2	$\frac{1}{2}$	$\frac{1}{2}$			
2	3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$		
3	4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$	
4	5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$

Unlike ABs or AMs, BDs formulas allocate the optimizing or free parameters to the $\{\alpha_j\}$ rather than the $\{\beta_j\}$, and hence are “maximally implicit” in the sense that the data $\{f^{(n+1-j)}\}$ enters the calculation only at the time step $n+1$. For this reason, BDs formulas are the hardest to implement of all linear multistep formulas, but they are also the most stable.

Table 3
Coefficients $\{\beta_j\}$ of Backwards Differentiation Formulas

number of steps s	order p	α_0	α_1	α_2	α_3	α_4	β_0
1	1	1	-1				1
2	2	1	$-\frac{4}{3}$	$\frac{1}{3}$			$\frac{2}{3}$
3	3	1	$-\frac{18}{11}$	$\frac{9}{11}$	$-\frac{2}{11}$		$\frac{6}{11}$
4	4	1	$-\frac{48}{25}$	$\frac{36}{25}$	$-\frac{16}{25}$	$\frac{3}{25}$	$\frac{12}{25}$

Here the sequence $v^{(n+1-j)}, \dots, v^{(n+1)}$ is considered as a discrete data of a continuous function $v(t)$. Since $v(t)$ is unknown, it is approximated by $p(t)$ which is the unique polynomial of degree $\leq s$ that interpolates the data $\{v^{(n+1-j)}\}$. In the definition of $p(t)$, $v^{(n+1)}$ is still an unknown, but is implicitly defined, and can be calculated by imposing the condition:

$$\frac{dp}{dt}(t = t_{n+1}) = f^{(n+1)} \quad (56)$$

which can be uniquely determine $\{\alpha_j\}$ in equation (55). These coefficients [Trefethen, 1989] [29, 34] are tabulated in Table 3.

The s -step Adams-Bashforth and Adams-Moulton formulas are stable for all $s \geq 1$. The backwards differentiation formulas are stable for $1 \leq s \leq 6$, but unstable for $s \geq 7$. The order of accuracy for each of these formulas is also tabulated in Tables 1, 2 or 3.

4.4.2 Predictor-Corrector

Typically, the explicit methods tend to be easier to implement and require less work per time step. In contrast, the implicit ones tend to be more stable as well as are able to take larger, and hence, fewer time steps with sacrificing accuracy to unstable oscillations [29].

The method of predictor-corrector takes advantage of merits of explicit and implicit methods, while removing the disadvantages of theirs. The idea of a predictor-corrector method is to settle for the approximate solution generated by an iterative procedure, instead of the exact solution, from an implicit formula. At each time-step, one explicit (such as ABs) "predictor" sub-step is taken, followed by one or more explicit (such as AMs) "corrector" sub-steps. In each corrector sub-step, the value $f^{(n+1)}$ on the right-hand side of the formula is taken from the precious sub-step, so that the Adams-Moulton

formula is effectively explicit. It can be shown that under reasonable hypotheses, the predictor-corrector iteration at each time-step would converge eventually to the exact solution of the Adams-Moulton formula [29].

Summarily, a truly effective use of numerical computation in applications requires both a theoretical knowledge of the subject and a computational experience with it. The theoretical knowledge should include an understanding of both the original problem being solved and of the numerical methods for its solution, including their derivation, error analysis, and an idea if they will perform well or poorly. The computational experience gives a sense of reality to most theoretical discussions; it brings out the important difference between the exact arithmetic implication in most theoretical discussions and the finite-length arithmetic computation. Also, rapid development of computer technology does not provide computer unlimited functions to handle all kinds of problems of any degree of complexity, constraints on the structure of numerical methods is always imposed in computer application.

5. NUMERICAL MODELS

The computer simulation of a time-dependent physical process is based upon a discretized version of a mathematical model of that process. Thus, in order to construct a numerical algorithm for the solution of the current problem, continuous field quantities such energy, temperature and spatial position, are replaced by their values at discrete points. Time itself is discretized as indicated in Section 4.4, and a marching process takes place through discrete time steps [3].

In order to simplify the numerical work for the current problem, which involves change of phase and movement of the phase-separation interface, Front-fixing method is applied to fix the moving boundary. In one-dimensional problem, the transformation [3, 13]

$$\xi = x / X(t), \quad t > 0 \quad (57)$$

which was proposed by Landau (1950), is used to fix the melting boundary at $\xi = 1$ for all t , maps the interval $0 \leq x \leq X(t)$ onto the fixed interval $0 \leq \xi \leq 1$. Equation (57) is used to transform the governing equations. In case the moving boundary does not move smoothly with time, it is possible to directly track moving boundary with help of this method [3, 4, and 13].

5.1. Non-dimensional Discrete Equations

In the present problem, the computational domain is divided into two sub-domain: the liquid region and the solid region. Moreover, Chebyshev collocation method requires that the computational domain for the space variable must be $[-1, 1]$. For these reasons, Landau transformation is modified as

$$\xi_L = \frac{2x}{X} - 1 \quad -1 \leq \xi_L \leq 1 \quad (58)$$

$$\xi_S = 2\left(\frac{x-X}{l-X}\right) - 1 \quad -1 \leq \xi_S \leq 1 \quad (59)$$

where the space in each sub-domain is discretized with the help of Gauss-Lobatto collocation points for Chebyshev polynomial:

$$\xi_{L,j} = \cos \frac{\pi(N-j)}{N} \quad j = 0, \dots, N, \quad (60)$$

$$\xi_{S,j} = \cos \frac{\pi(N-j)}{N} \quad j = 0, \dots, N, \quad (61)$$

In the first stage of this problem, the liquid region does not exist and phase transition does not happen until the free surface temperature gets to the melting temperature. Thus, in the first stage, $X(t) = 0$, the equation (52) is not valid and the equation (59) is replaced by

$$\xi_S = \frac{2x}{l} - 1 \quad -1 \leq \xi_S \leq 1 \quad (62)$$

The following dimensionless quantities are used to simplify the governing equations:

$$\tau = \frac{t}{\alpha(\rho L / q'')^2} \quad (63)$$

$$\theta_L(\xi_L, \tau) = \frac{C}{L}(T_L - T_m) \quad \text{for liquid region} \quad (64)$$

$$\theta_S(\xi_S, \tau) = \frac{C}{L}(T_S - T_m) \quad \text{for solid region} \quad (65)$$

where $\alpha = \frac{k}{c\rho}$ is thermal diffusivity. In the second stage of the problem, the interface

location is non-dimensionally represented by:

$$\psi(\tau) = X \frac{\alpha \rho L}{q''} \quad (66)$$

In the first stage of calculations, non-dimensional versions of the governing equations, corresponding to equations (10)-(13), can be rewritten as

$$\frac{\partial \theta_s}{\partial \tau} = \frac{\partial \theta_s^2}{\partial \xi_s^2} \cdot \frac{4\alpha^2 \rho^2 L^2}{q''^2 l^2} \quad (67)$$

$$\theta_s(\xi_s, 0) = \frac{c}{L}(T_i - T_m) \quad (68)$$

$$\left. \frac{\partial \theta_s}{\partial \xi_s} \right|_{\xi_s=1} = -\frac{q'' l c}{2kL} \quad (69)$$

$$\left. \frac{\partial \theta_s}{\partial \xi_s} \right|_{\xi_s=1} = -\frac{hlc}{2kL} \left[\frac{L}{c} \theta_s(1, \tau) + T_m - T_a \right] \quad (70)$$

The corresponding equations for the second stage of calculations can be rewritten as

$$\frac{\partial \theta_L}{\partial \tau} = \frac{\partial \theta_L^2}{\partial \xi_L^2} \cdot \frac{4}{\psi^2} + \frac{\partial \theta_L}{\partial \xi_L} \cdot \frac{d\psi}{d\tau} \cdot \frac{\xi_L + 1}{\psi} \quad (71)$$

$$\frac{\partial \theta_s}{\partial \tau} = \frac{\partial \theta_s^2}{\partial \xi_s^2} \cdot \frac{4}{\left(\frac{q'' l}{\alpha \rho L} - \psi \right)^2} - \frac{\partial \theta_s}{\partial \xi_s} \cdot \frac{d\psi}{d\tau} \cdot (\xi_s - 1) \cdot \frac{1}{\left(\frac{q'' l}{\alpha \rho L} - \psi \right)} \quad (72)$$

$$\theta_L(\xi_L, 0) = 0 \quad (73)$$

$$\psi(0) = 0 \quad (74)$$

$$\psi = -2 \left. \frac{\partial \theta_L}{\partial \xi_L} \right|_{\xi_L=1} \quad (75)$$

$$\theta_s(1, \tau) = -\frac{2q'' C}{hL} \cdot \left. \frac{\partial \theta_s}{\partial \xi_s} \right|_{\xi_s=1} \cdot \frac{1}{\left(\frac{q'' l}{\alpha \rho L} - \psi \right)} - \frac{c}{L}(T_m - T_a) \quad (76)$$

$$\frac{d\psi}{d\tau} = -\frac{2}{\psi} \left. \frac{\partial \theta_L}{\partial \xi_L} \right|_{\xi_L=1} + \frac{2}{\left(\frac{q'' l}{\alpha \rho L} - \psi \right)} \left. \frac{\partial \theta_S}{\partial \xi_S} \right|_{\xi_S=1} \quad (77)$$

$$\theta_L(1, \tau) = \theta_S(-1, \tau) = \theta_f \quad (78)$$

$$\theta_f = 0 \quad \text{in Model I} \quad (79-1)$$

$$\frac{d\psi}{d\tau} = \frac{\rho L V_0}{q''} \exp \left(-\frac{\left(\frac{\Delta g_b}{k_B T_m} \right)}{\left(1 + \frac{L}{c T_m} \theta_f \right)} \right) \left(1 - \exp \left(-\frac{L^2 M}{c N_A k_B T_m^2} \theta_f \right) \right) \quad \text{in Model II} \quad (79-2)$$

The above governing equations are discretized and converted into a system of algebraic equations with the procedures described in Chapter 4. Values of the discretization parameters used for different parts of the problem are discussed in the following sections.

The resulting system of algebraic equations of the form

$$\sum_{j=0}^N a_{ij} x_j^{(n+1)} = b_i \quad i = 0, 1, \dots, N, \text{ for } n \geq 0 \quad (80)$$

are solved by Gauss-Seidel iteration with under-relaxation [25]:

$$\left[x_i^{(n+1)} \right]^{(k+1)} = \left[x_i^{(n)} \right]^{(k)} + \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=0}^N a_{ij} \left[x_j^{(n)} \right]^{(k)} \right] \quad (81)$$

where $[a_{ij}]$ is the coefficient matrix, $[x_j^{(n+1)}]^{(k)}$ is the unknown variable at the j^{th} collocation point at time step $(n+1)$ in the k^{th} iteration of the solver. The following stopping criterion is used in the iterative procedure:

$$\left| \frac{x^{k+1} - x^k}{x^{k+1}} \right| \leq 10^{-10} \quad (82)$$

5.2. The Flowchart of Programming

The final step in computer simulation of a physical problem is to develop and implement algorithms in a computer code [3]. In this study, C language [37] is chosen for its powerful functions and its portability to implement the algorithms. The following procedures have been adapted in the main program:

- I. The calculation of Chebyshev polynomial and related quantities as implemented by the function "ChebysS1."
 - I.1. Calculation of Gauss-Lobatto collocation point, ζ .
 - I.2. Calculation of trial function, T .
 - I.3. Calculation of the first order derivative matrix, $D1$.
 - I.4. Calculation of the second order derivative matrix, $D2$.

- II. Calculation of transient temperature profiles inside the target for the first stage. This step is continued until melting starts, i.e. $\theta_s(-1, \tau) \geq 0$.
 - II.1. Calculation for the initial two time-step: For $time = \Delta\tau$ and $time = 2\Delta\tau$, temperature fields in the solid are solved through the method of predictor-corrector (1-step Adams-Bashforth with 0-step Adams-Moulton formulas). The calculation is implemented in the function "ex_melting_AB_AM1."
 - II.2. Calculation for subsequent time-step: temperature fields for subsequent time steps are solved through the 3-step Adams-Bashforth method. The calculation is repeated over and over again by marching forward in time until the temperature of free surface becomes equal to or more than the melting temperature, corresponding to a value of 0 for $\theta_s(-1, \tau)$. These operations are implemented by the function "ex_melting_AB3".
 - II.3. Calculation of the exact time when melting starts, and the temperature profile at that moment: In the process of computation with a given time interval $\Delta\tau$, the situation at the end of iterations in step II.2 is often as illustrated in Figure 2.

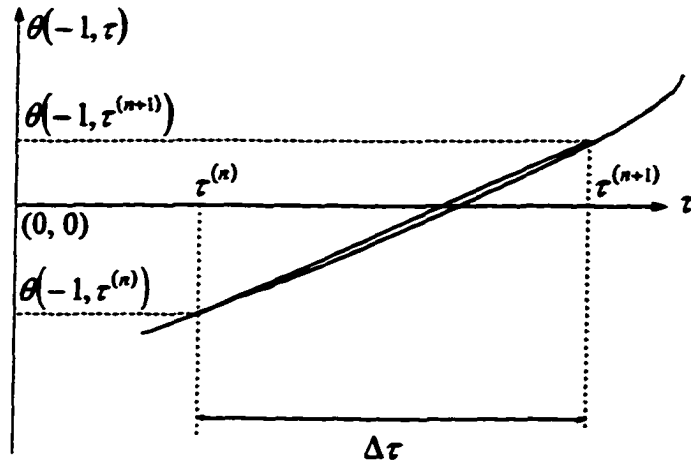


Figure 2

Variation of $\theta_s(-1, \tau)$ With Respect To τ Around The Critical Point
(The Melting Temperature)

If the time interval $\Delta\tau$ is enough small, it is reasonable that $\theta_s(-1, \tau)$ is a linear function with respect to τ between $\tau^{(n)}$ and $\tau^{(n+1)}$. Thus, the time τ_s that $\theta_s(-1, \tau) = 0$ is calculated up by linear interpolation:

$$\tau_s = \tau^{(n)} - \Delta\tau \frac{\theta_s(-1, \tau^{(n)})}{\theta_s(-1, \tau^{(n+1)}) - \theta_s(-1, \tau^{(n)})} \quad (83)$$

The corresponding temperature profile at this time τ_s is determined by

$$\theta_s(\xi_j, \tau^{(n+1)}) = \theta_s(\xi_j, \tau^{(n)}) - \frac{\theta_s(-1, \tau^{(n)}) \left[\theta_s(\xi_j, \tau^{(n+1)}) - \theta_s(\xi_j, \tau^{(n)}) \right]}{\theta_s(-1, \tau^{(n+1)}) - \theta_s(-1, \tau^{(n)})} \quad (84)$$

where j is collocation points index. This step is implemented by the function "critical_point."

- III. Calculation of post-melting temperature profiles in the target: For this calculation, the time when melting starts as calculated by the previous step is referred to as

time = 0, and the initial condition is the temperature profile as provided by step II.3. This calculation is continued until temperature at bottom boundary of target gets sufficiently close to the melting temperature, i.e. the melt interface is sufficiently close to the bottom surface. The calculation is divided into two alternative calculations: one for Stefan condition with consideration of kinetics at the interface and the other for kinetics condition at the interface.

III.1. When Stefan condition is considered, discrete equations corresponding to equations (71)-(72) and (79-1) are used.

III.1.1. Calculation for starting time steps: Calculation of the initial two time steps ($time = \Delta\tau$ and $time = 2\Delta\tau$) is performed by 1-step BDs, as implemented by the function "Tstep_BD1".

III.1.2. Calculation for subsequent time steps: 3-step BDs is used for the subsequent time steps. The algorithm is implemented in the function "Tstep_BD3".

III.2. When kinetics condition is considered at the interface, discrete equations corresponding to equations (71)-(72) and (79-2) are used. Time marching is accomplished by BDs. As the matrix created by the governing equations and conditions is quite stiff, the initial guesses are important for the stability and rate of convergence. A function "initialize_kinetics" is used before each time step to preset the temperature profile in liquid region to a linear function in space.

III.2.1. Calculation for starting time steps: Calculation of the initial two time steps ($time = \Delta\tau$ and $time = 2\Delta\tau$) is performed by 1-step BDs, as implemented by the function "Tstep_BD1_Kinetics".

III.2.2. Calculation for subsequent time steps: 3-step BDs is used for the subsequent time steps. The algorithm is implemented in the function "Tstep_BD3_Kinetics".

In the future, the program should be further developed to include (i) more efficient methods for solving stiff matrix, and (ii) more physical processes with relevant equations in the mathematical models.

6. RESULTS AND CONCLUSIONS

Laser melting of a pure iron slab is used as a test case. The thermodynamic properties of pure iron and the corresponding test parameters are listed in Table 4.

Table 4
Thermodynamic Properties of Target (Pure Iron) & Test Parameters

	Quantities	Units
density, ρ	7297.0	Kg/m^3
specific heat, c	447.0	$J/Kg - K$
thermal conductivity, k	20.0	$w/m - K$
thermal diffusivity, α	$0.226631.5 \times 10^{-10}$	m^2/s
heat transfer coefficient, h	100.0	$w/m^2 - K$
equilibrium temperature, T_E	1211.0	K
ambient temperature, T_a	300.0	K
latent heat, L	247.3	Kj/Kg
target thickness, l	1.0×10^{-3}	m
heat flux, q''	2.0×10^8	w/m^2

Solutions for the test are obtained with both non-kinetics condition and kinetics condition, and presented as from Figure 9 to 12. In both solutions, a fixed amount of time (0.0125s, Figure 12, as obtained from calculation in step II) is needed before the hot surface temperature, and the melting interface starts moving.

With either condition, only a portion of the incident laser energy is conducted to the interface through the liquid layer, while the rest is used to raise temperature of the existing liquid. A part of the heat that reaches the interface is diffused into the solid, while the rest is used in melting and moving the interface. With kinetics condition, not only the heat necessary for melting is to be transported to the interface, but also some additional heat is needed to raise the interface temperature above the melting temperature. As a result, the interface moves slower with kinetics condition, and the interface temperature is always above the melting temperature during melting (Figures 9 and 10). In other words, the interface has to be always superheated before melting can occur. Under equilibrium (i.e. Stefan) condition, the superheat is so minimal, that in all practical calculations the superheat can be neglected and the interface can be considered to be at the equilibrium melting temperature.

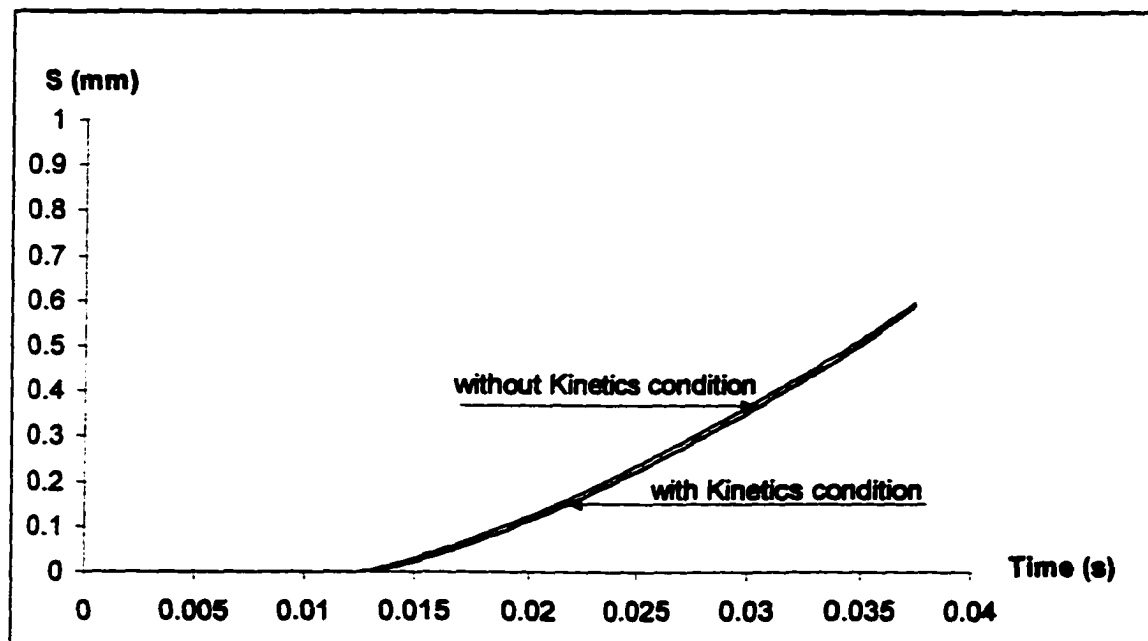


Figure 9
Change in Melting Interface Location with Time

The free surface temperature is also higher with kinetics condition than that with Stefan condition. The difference in location of melt interface between the two cases is the smallest right after melting starts. This difference gradually increases until the heat flow through the interface starts to be affected by the back surface heat transfer, and the difference starts decreasing again (Figure 9). The difference in interface temperature between the two cases continuously grows with time (Figure 10), so does the difference in free surface temperature (Figure 11). The differences in temperature profile also increase with time (Figure 12). Since the interface movement is slower in the case with kinetics condition, the kink (or discontinuity in slope) at the interface is less prominent for the case with the kinetics condition (Figure 12).

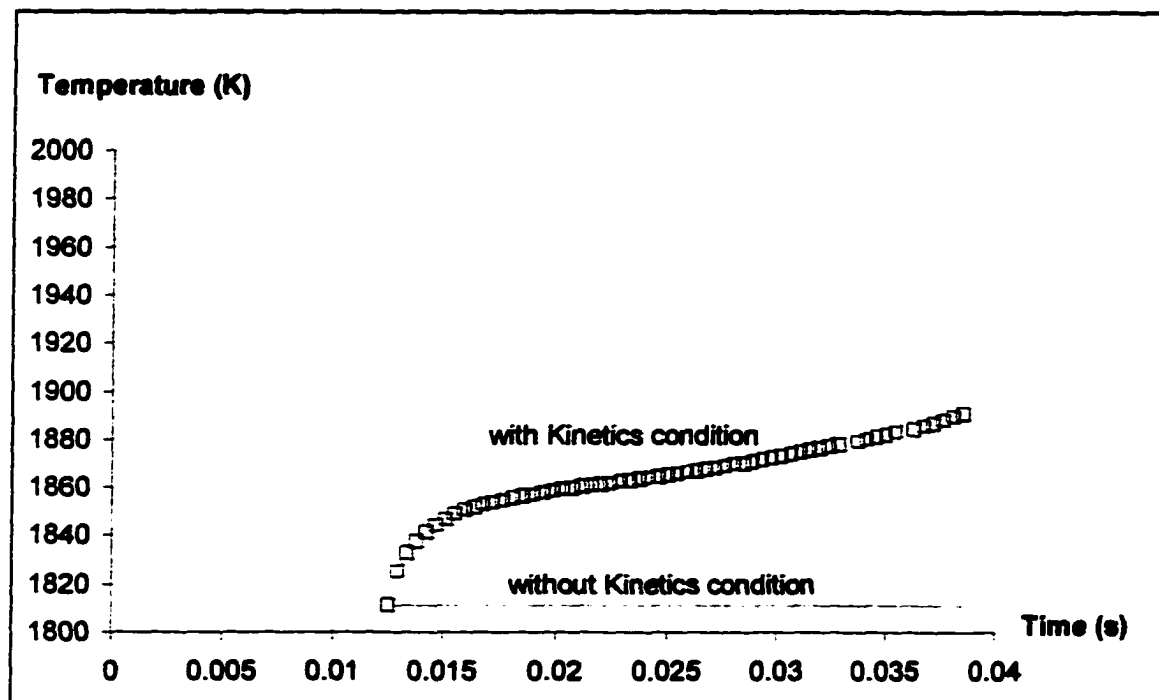


Figure 10
Change in Melt Interface Temperature with time

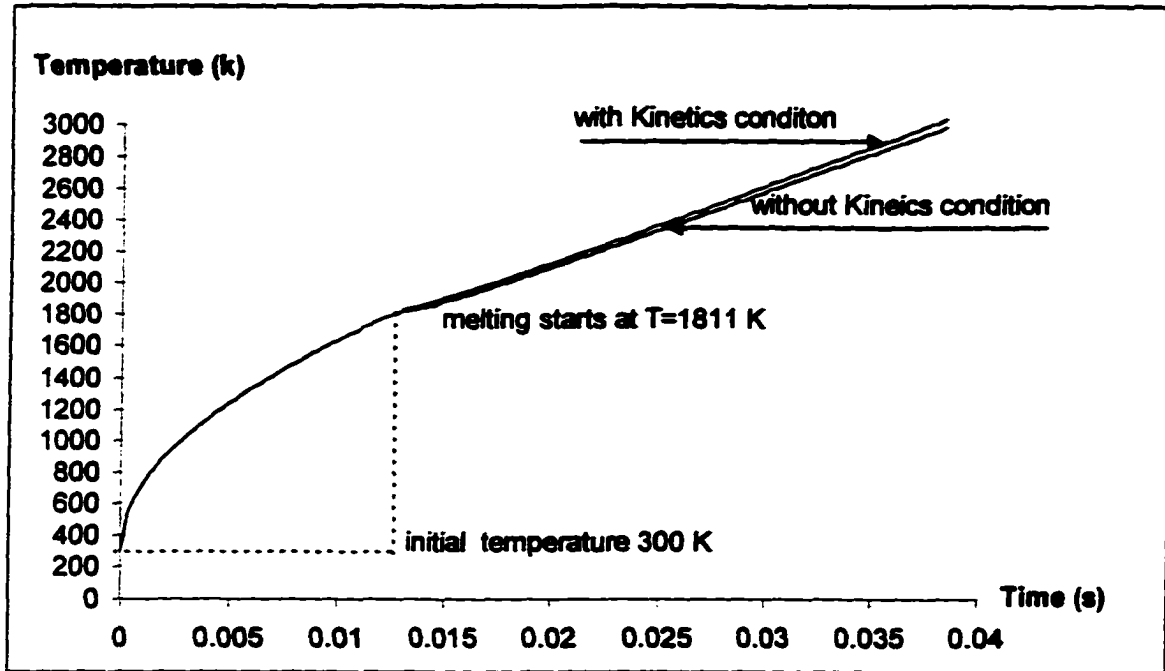


Figure 11
Change in Free Surface Temperature with Time

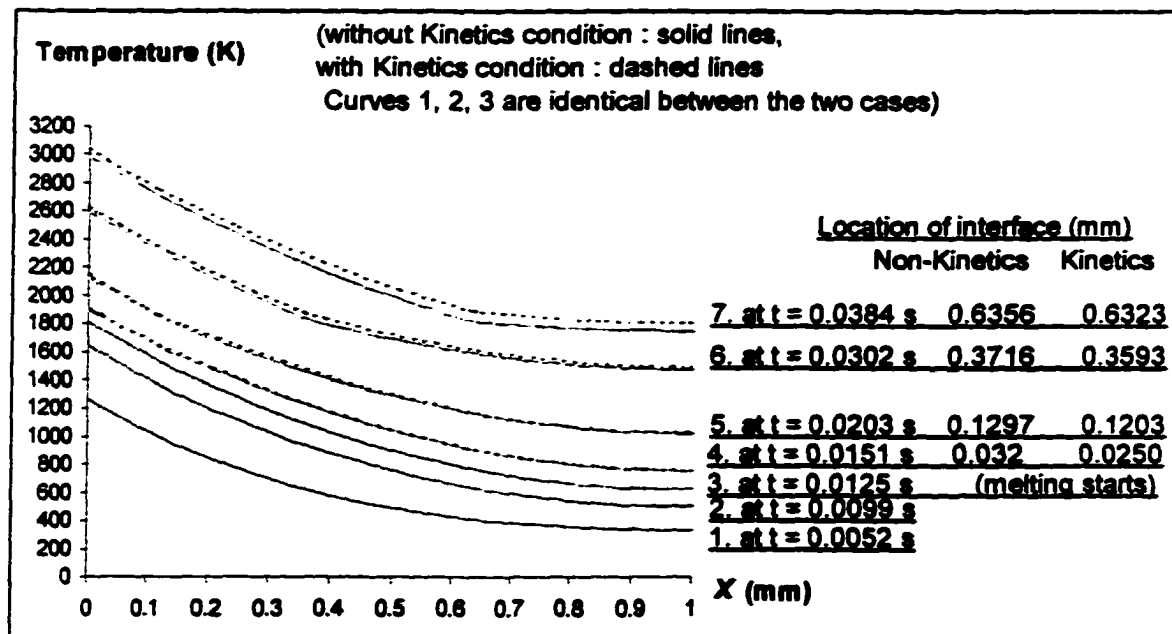


Figure 12
2-Phase Temperature Profiles

At high laser power density, and depending on the material properties, the kinetics condition can become important. If kinetics condition is not used, the moving interface location is over-predicted and the free surface temperature is under-predicted. Both of these two factors may have important consequences in different laser processing techniques, such as laser annealing, laser machining and PLD. For example, in laser annealing where the amount of superheating determines the microstructure of the annealed solid, the computation must consider the kinetics condition [4].

REFERENCES

REFERENCES

1. Kapat J. S., Wei Z. and Kumar A., "A computational model for simulation of laser ablation in a pulsed laser deposition process," Intl. Con. on Metallurgical Coatings & Thin Films, paper BP.01, San Diego, 1997; being reviewed for *Surface and Coatings Tech.*
2. Miller J. C., (Ed.), "Laser ablation principles and applications," Spring-Verlag, New York, 1994.
3. Alexiades V. and Solomon A. D., "Mathematical modeling of melting and freezing processes," Hemisphere Publishing Corporation, Washington, 1993.
4. Kapat J. S., Wei Z. and Kumar A., "Role of kinetics in laser processing," Acceptance for publication in *Applied Surface Science*.
5. Chan H. L., "Laser processing of carbide and nitride coatings for multifunctional applications," Department of Electrical Engineering, University of South Alabama, 1997.
6. Sobol E. N., "Phase transformations and ablation in laser-treated solids," John Wiley & Sons, New York, 1995.
7. Arata Y., "Plasma, electron and laser beam technology," American Society for metals, Ohio, 1926.
8. Schwarz H. J. and Hora H., editors, "Laser interaction and related plasma phenomena," Plenum Press, New York, NY, 1972.
9. Schroder K., "Electronic, Magnetic, and Thermal Properties of Solid Materials," Marcel Dekker, Inc., New York, 1972.
10. Chrisey D. B. and Hubler G. K., "Pulsed laser deposition of thin films," John Wiley & Sons, Inc., New York, 1994.
11. Crank J., "Free and moving boundary problems," Clarendon Press, Oxford, UK, 1924.
12. Rohsenow W. M., Hartnett J. P. and Ganic E. N., Editors, "Handbook of heat transfer applications," McGraw-Hill Book Company, New York, 1925.

13. Spall R., "Spectral collocation methods for one-dimensional phase-change problems," *Int. J. Heat Mass Transfer*, vol. 32, no. 15, pp 2743 - 2742, 1995.
14. Stanley H. E., "Introduction to phase transitions and critical phenomena," Oxford University Press, New York, 1927.
15. Porter D. A. and Easterling K. E., "Phase Transformations in Metals and Alloys," Chapman & Hall, London, UK, 1992.
16. Kakac S. and Yener Y., "Heat conduction," Taylor & Francis, 1993.
17. Kreith F. and Bohn M. S., "Principles of heat transfer," West Publishing Company, New York, 1993.
18. Sadd, M. H. and Didlake J. E., "Non-Fourier melting of a semi-infinite solid," *Transactions of the ASME Journal of Heat Transfer*, pp 25 – 22, Feb. 1977.
19. Vedavarz A., Kumar S. and Moallemi M. K., "Significance of non-Fourier heat waves in conduction," *Transactions of the ASME Journal of Heat Transfer*, vol. 116, pp 221 – 224, Feb. 1994.
20. Yuan W. W. and Lee S. C., "Non-Fourier heat conduction in a semi-infinite solid subjected to oscillatory surface thermal disturbances," *Transactions of the ASME Journal of Heat Transfer*, vol. 111, pp 172 – 121, Feb. 1929.
21. Holman J. P., "Heat transfer," McGraw-Hill, Inc., 1990.
22. Lapidus L. and Pinder G. F., "Numerical solution of partial differential equations in science and engineering," John Wiley & Sons, 1922.
23. Yu G, Lee S. T., Lai J. K. L. and Ngai L., "Kinetics of transformation with nucleation and growth mechanism: Special consideration of crystallographic relationship and epitaxial growth," *J. Appl. Phys.*, vol. 21(1), pp 29-95, Jan. 1997.
24. Mitra, K., Kumar S. and Vedavarz A., "Parametric aspects of electron-phonon temperature model for short pulse laser interactions with thin metallic films," *J. Appl. Phys.*, vol. 20, no. 2, pp 675 – 620, Jul. 1996.
25. Minkowycz W. J., Sparrow E. M., Schneider G. E. and Pletcher R. H., Editors, "Handbook of numerical heat transfer," John Wiley & Sons, New York, 1922.

26. Vandergraft S. J., "Introduction to numerical computations," Academic Press, New York, 1923.
27. Canuto C., Hussaini M. Y., Quarteroni A. and Zang T. A., "Spectral methods in fluid dynamics," Springer-Verlag, New York, 1927.
28. Fletcher C. A. J., "Computational techniques for fluid dynamics," Springer-Verlag, New York, 1997.
29. Trefethen L. N., "Finite difference and spectral methods," Department of Mathematics, Massachusetts Institute of Technology, 1929.
30. Jordan C., "Calculus of finite differences," Chelsea Publishing Company, New York, NY, 1965.
31. Logan D. L., "A first course in the finite element method," PWS-KENT Publishing Company, Boston, MA, 1992.
32. Pearson C. E., "Numerical methods in engineering and science," Van Nostrand Reinhold Company, New York, 1926.
33. Gothlieb D. and Orszag S. A., "Numerical analysis of spectral methods: Theory and applications," Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1977.
34. Iserles A., "A first course in the numerical analysis of differential equations," Cambridge University Press, New York, 1996.
35. Shampine L. F. and Allen R. C., Jr., "Numerical computing: an introduction," W. B. Saunders Company, Philadelphia, PA, 1973.
36. Rivlin T. J., "The Chebyshev polynomials," John Wiley & Sons, New York, NY, 1974.
37. Kelley A. and Pohl I., "C by dissection," Addison-Wesley Publishing Company, Menlo Park, California, 1996.
38. Geohagan, D. G., 1997, Private Communications, Solid State Division, Oak Ridge National Laboratory, Oak Ridge, TN.
39. Bloembergen, N., "Pulsed laser interactions with condensed matter," in Beam-Solid Interactions and Phase Transformations, H. Kurz, G. L. Olson, and J. M. Poate (eds.), Materials Research Society Symposia Proceedings, vol. 51, MRS Pittsburgh.

APPENDIX (Programs)

The Program for Graduate Thesis:
"Numerical Simulation of Laser Ablation in Pulsed Laser Deposition Process"
by Zhaohui Wei
in
Department of Mechanical Engineering of The University of South Alabama
for
A Master's degree in Mechanical Engineering
Fall, 1997

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
/*
                                     Computational parameters
*/
/*-----*/
#define M      24                      /* ----- array size */
#define dt     0.001                   /* ----- time step */
#define tolerance 0.0000000001
/*
Material:                pure iron
Density:                 d = 7897          kg/m3
Specific heat in liquid: Cl = 447         j/kg-k
Specific heat in solid:  Cs = 447         j/kg-k
Thermal conductivity in liquid: Kl = 80   w/m-k
Thermal conductivity in solid: Ks = 80    w/m-k
Thermal diffusivity in liquid: al = 0.00002266315 m2/s
Thermal diffusivity in solid: as = 0.00002266315 m2/s
Heat transfer coefficient: h = 100        w/m2-k
Air temperature:        Ta = 300         k
Melting temperature:    Tm = 1811        k
Latent heat:            L = 247300       j/kg
Material length         l = 0.001        m
Heat flux:              q" = 200000000   w/m2
*/

#define Tm      1811.0                 /* k  melting temperature */
#define Ta      300.0                  /* k  air temperature */
#define TIME    0.0021608907          /* s,  time dimensionless
                                     = al*2(d*L/q") */
#define SPACE   0.0002212975          /* m,  space dimensionless = al*d*L/q" */
#define TL      553.24385              /* k,  temperature dimensionless = L/Cl */
#define TS      553.24385              /* k,  temperature dimensionless = L/Cs */
#define length  0.001                  /* m */
#define g       0.1958821              /* 4*as*TIME/(l*l) */
  
```

```

#define g1          1250.0          /* q**1/(2*k)          */
#define g2          0.000625       /* h**1/(2*k)          */
#define AA          1.0            /* dimensionless, al/as */
#define BB          4.5188036      /* dimensionless, (1*q**)/(al*d*L) */
#define CC          7230.0849      /* dimensionless, (2*q**Cs)/(h*L) */
#define IL          -2.7311646     /* dimensionless, (Ta-Tm)*c/L */
#define IS          -2.7311646     /* dimensionless, (Ta-Tm)*cs/L */
#define KV          48840.46638
#define KG          9.05
#define KH          1.086
/*-----

```

Subroutine Functions

```

----- */
typedef double FTYPE;

void ChebyS1(int N, FTYPE x[M], FTYPE T[M][M], FTYPE D1[M][M],
             FTYPE D2[M][M]);
void solve_equation(FILE *fpr, int method, int N, int *iter, FTYPE relax,
                   FTYPE a[M][M], FTYPE c[M], FTYPE U[M]);
void scale_lin_system(int N, FTYPE a[M][M], FTYPE c[M]);
void print_message(int N, FILE *fpr);
void print_vector(int N, char *Str, FILE *fpr, FTYPE vector[M]);
void print_matrix(int N, char *Str, FILE *fpr, FTYPE matrix[M][M]);
void print_ex_melt(int N, int n, FILE *fpr, FTYPE x[M], FTYPE US[6][M]);
void print_result(int N, int n, FILE *fpr, FTYPE x[M], FTYPE psi[6],
                 FTYPE Dpsi[6], FTYPE UL[6][M], FTYPE US[6][M], FTYPE nt);
/*----- */
void ex_melting_AB_AM1(int N, int n, int approach, FTYPE x[M], FTYPE D1[M][M],
                      FTYPE D2[M][M], FTYPE US[6][M], FILE *fpr);
void ex_melting_AB3(int N, FTYPE x[M], FTYPE D1[M][M], FTYPE D2[M][M],
                   FTYPE US[6][M], FILE *fpr);
void critical_point(int N, int n, FTYPE x[M], FTYPE US[6][M], FILE *fpr);

/**** SPECIAL SUBROUTINE FUNCTIONS IN MODEL I WITHOUT
CONSIDERATION OF INFLUENCE OF KINETICS IN PHASE
TRANSITION ****/

void TStep_BD1(int N, int n, FTYPE relax, FILE *fpr,
              FTYPE x[M], FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M],
              FTYPE Dpsi[M], FTYPE UL[6][M], FTYPE US[6][M]);
void TStep_BD3(int N, FTYPE relax, FILE *fpr, FTYPE x[M], FTYPE D1[M][M],
              FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M], FTYPE UL[6][M],
              FTYPE US[6][M]);
void end_melt(int N, int n, FTYPE x[M], FTYPE psi[M], FTYPE Dpsi[M],

```

```

        FTYPE UL[6][M], FTYPE US[6][M], FILE *fpr);
/**** SPECIAL SUBROUTINE FUNCTIONS IN MODEL II WITH
CONSIDERATION OF INFLUENCE OF KINETICS IN PHASE
TRANSITION ****/
void initialize_kinetics(int N, int n, FILE *fpr, FTYPE x[M], FTYPE D1[M][M],
        FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M], FTYPE UL[6][M],
        FTYPE US[6][M]);
void TStep_BD1_Kinetics(int N, int n, FTYPE relax, FILE *fpr, FTYPE x[M],
        FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M],
        FTYPE UL[6][M], FTYPE US[6][M]);
void TStep_BD3_Kinetics(int N, FTYPE relax, FILE *fpr, FTYPE x[M],
        FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M],
        FTYPE Dpsi[M], FTYPE UL[6][M], FTYPE US[6][M]);
/*-----
x[i],          the Gauss-Lobatto points on [-1,1]
T[i][j],       ith trial function (Chebyshev poly. of the first kind) at jth point
D1[i][j],      dU/dx @x[i] = sum_j(D1[i][j]*Uj) :exact for Polyn. up to degree=N
D2[i][j],      d2U/dx2 @x[i] = sum_j(D2[i][j]*Uj) :exact for Polyn. up to degree=N
TStep_BD3,     third-order backwards differentiation
-----*/

                                Main program
-----*/

int main()
{
    int      n, nn, N, melt, approach;
    FTYPE    zeta[M], T[M][M], D1[M][M], D2[M][M];
    FTYPE    psi[6], Dpsi[6], UL[6][M], US[6][M], f[6][M];
    FTYPE    relax, nt;
    time_t   ti;
    FILE     *fout;
    fout = fopen("data.txt", "w");
    relax = 0.1;
    printf("Enter the order N, (N should be less than %d)  N =", M-3);
    scanf("%d", &N);
    ChebyS1(N, zeta, T, D1, D2);
    print_message(N, fout);
/*
    print_vector(N, "Gauss-Lobatto point", fout, zeta);
    print_matrix(N, "Trial function Tk(xj)", fout, T);
    print_matrix(N, "1-order derivative D1", fout, D1);
    print_matrix(N, "2-order derivative D2", fout, D2);
    n = 0;
    approach = 0;
    do

```

```

{
  nn=n;
  if (n==0 || n%200==0)
  print_ex_melt(N, n, fout, zeta, US);
  if (n<=1)
  ex_melting_AB_AM1(N, n, approach, zeta, D1, D2, US, fout);
  else
  ex_melting_AB3(N, zeta, D1, D2, US, fout);
  n++;
  if (n>=3)
  nn=3;
  if (US[nn][0]>=Tm)
  {
    critical_point(N, n, zeta, US, fout);
    nt=US[0][N+1];
    melt=1; /* break */
  }
} while (melt!=1);
n=0;
/*
THE FOLLOWING LOOPS ARE USED IN MODEL I AND MODEL II,
RESPECTIVELY. BE ATTENTION IN RUNNING THIS PROGRAM,
THERE IS ONLY ONE LOOP MAY BE ACTIVITED TO BE RUN
WHILE ANOTHER LOOP MAY NOT BE STATIONARY, NAMELY,
NO TWO LOOPS CAN BE RUN AT THE SAME TIME. */

/**** SPECIAL LOOP IN MODEL I WITHOUT CONSIDERATION OF
INFLUENCE OF KINETICS IN PHASE TRANSITION *****/
/* do
{
  if (n<3)
    TStep_BD1(N, n, relax, fout, zeta, D1, D2, psi, Dpsi, UL, US);
  else
    TStep_BD3(N, relax, fout, zeta, D1, D2, psi, Dpsi, UL, US);
  time(&ti);
  printf("%d %s\n", n, ctime(&ti) );
  if (n==0 || (n+1)%100==0)
  print_result(N, n, fout, zeta, psi, Dpsi, UL, US, nt);
  n++;
  }while((n<5) || (US[3][N]<(-1.0/TS)));
/**** SPECIAL LOOP IN MODEL II WITH CONSIDERATION OF
INFLUENCE OF KINETICS IN PHASE TRANSITION *****/

do

```

```

    {
        if (n<3)
            TStep_BD1_Kinetics(N, n, relax, fout, zeta, D1, D2, psi, Dpsi, UL, US);
        else
            TStep_BD3_Kinetics(N, relax, fout, zeta, D1, D2, psi, Dpsi, UL, US);
        time(&ti);
        printf("%d  %s\n", n, ctime(&ti) );
        if (n==0 || n%200==0)
        {
            print_result(N, n, fout, zeta, psi, Dpsi, UL, US, nt);
            fflush(fout);
        }
        n++;
    } while((n<5) || (US[3][N]<(-0.1/TS)));
    return 0;
}

/*                                = function =
-----*/
void ChebyS1(int N, FTYPE x[M], FTYPE T[M][M], FTYPE D1[M][M], FTYPE
D2[M][M])
{
    int    i, j, k;
    FTYPE  c[M];
    FTYPE  a[M][M], a1[M][M], a2[M][M];
    /* initialization to 0; particularly important is the initialization of
a1[N+1][j], a1[N][j], a2[N+1][j], a2[N][j] */
    for (i=N; i<N+2; i++)
    {
        for (j=0; j<=N+2; j++)
        {
            a1[i][j]=0.0;
            a2[i][j]=0.0;
        }
    }
    /* calculation of Gauss-Lobatto points and values of Cheby. functions at those points */
    for (i=0; i<=N; i++)
    {
        for (j=0; j<=N; j++)
        {
            T[i][j]=cos(M_PI*(FTYPE) i*(1.0-(FTYPE) j/(FTYPE) N));
        }
        x[i]=cos(M_PI*(1.0-(FTYPE) i/(FTYPE) N)); /* calculation of xj*/
    }
}

```

```

/* internal arrays and matrices needed for calculation d1 and d2 matrices */
c[N]=2.0 ;
c[0]=2.0 ;
for (k=1; k<N; k++)
{
    c[k]=1.0;
    for (k=0; k<=N; k++)
    {
        for (j=0; j<=N; j++)
        {
            a[k][j]=T[k][j]*2.0/(N*c[k]*c[j]);
        }
    }
    for (k=N; k>=1; k--)
    {
        for (j=0; j<=N; j++)
        {
            a1[k-1][j]=(a1[k+1][j]+2*k*a[k][j])/c[k-1];
            a2[k-1][j]=(a2[k+1][j]+2*k*a1[k][j])/c[k-1];
        }
    }
}
/* d1 and d2 needed to calculate 1 & 2-order deriv of U(x,t) w.r.t x */
for (i=0; i<=N; i++)
{
    for (k=0; k<=N; k++)
    {
        for (j=0; j<=N; j++)
        {
            D1[i][k]=D1[i][k]+a1[j][k]*T[j][i]; /* to calculate 1st deriv. */
            D2[i][k]=D2[i][k]+a2[j][k]*T[j][i]; /* to calculate 2nd deriv. */
        }
    }
}
return;
} /* End void ChebyS1 */

void print_message(int N, FILE *fpr)
{
    fprintf(fpr, "collocation points number, N = %d\n", N);
    fprintf(fpr, "time step, dt = %4.3e\n", dt);
    fprintf(fpr, "Tolerance = %3.2e\n\n", tolerance);
    fprintf(fpr, "----- \n");
    fprintf(fpr, "material: pure iron \n");
    fprintf(fpr, "length: l = 0.001 m \n");
}

```

```

    fprintf(fpr, "melting Temperature: Tm = 1811      k  \n");
    fprintf(fpr, "air temperature:   Ta = 300      k  \n");
    fprintf(fpr, "heat flux:           q = 200000000    w/m2  \n");
    fprintf(fpr, "----- \n");
    return;
}

void print_vector(int N, char *Str, FILE *fpr, FTYPE vector[M])
{
    int i, j;
    int block, max;
    max=5;
    block=N/6;
    fprintf(fpr, "\n\n");
    fprintf(fpr, "%s :\n", Str);
    fprintf(fpr, "----- \n");
    for (i=0; i<=block; i++)
    {
        if ((N%6)<=max && i==block)
            max=N%6;
        fprintf(fpr, " ");
        for (j=0; j<=max; j++)
        {
            fprintf(fpr, " x%2d      ", j+i*6);
        }
        fprintf(fpr, "\n");
        fprintf(fpr, " ");
        for (j=0; j<=max; j++)
        {
            fprintf(fpr, "%11.4e ", vector[j+i*6]);
        }
        fprintf(fpr, "\n\n");
    }
    return;
}

void print_matrix(int N, char *Str, FILE *fpr, double matrix[M][M])
{
    int i, j, k;
    int block, max ;
    max=5;
    block=N/6;
    fprintf(fpr, "\n\n");
    fprintf(fpr, "%s\n", Str);

```



```

fprintf(fpr, "-----\n");
for (i=0; i<=block; i++)
{
    if ((N%6)<=max && i==block)
        max=N%6;
    fprintf(fpr, " ");
    for (j=0; j<=max; j++)
    {
        fprintf(fpr, " C%2d ", j+i*6);
    }
    fprintf(fpr, "\n");
    for (j=0; j<=N; j++)
    {
        fprintf(fpr, "R%2d ", j);
        for (k=0; k<=max; k++)
        {
            fprintf(fpr, "%11.4e ", matrix[j][k+i*6]);
        }
        fprintf(fpr, "\n");
    }
    fprintf(fpr, "\n");
}
return;
}

void print_result(int N, int n, FILE *fpr, FTYPE x[M], FTYPE psi[6],
    FTYPE Dpsi[6], FTYPE UL[6][M], FTYPE US[6][M], FTYPE nt)
{
    int i, j;
    if (n>3) i=3;
    else i=n;
    fprintf(fpr, "\n----- U(%d,x)\n", n+1);
    fprintf(fpr, "psi[%d] = %12.6e Dpsi[%d] = %12.6e\n", n+1, psi[i+1], n+1, Dpsi[i+1]);
    fprintf(fpr, "Time = %12.6e s ", (dt*(n+1)+nt)*TIME);
    fprintf(fpr, "front S = %12.6e mm\n", psi[i+1]*SPACE*1000.0);
    fprintf(fpr, "Col. Space z (mm) Temperature, Temp.\n\n");
    for (j=0; j<=N; j++)
    {
        fprintf(fpr, "%2d, %11.4e %18.10e %18.10e\n",
            j, (x[j]+1.0)*psi[i+1]*SPACE*1000.0/2.0, UL[i+1][j]*TL+Tm, UL[i+1][j]);
    }
    for (j=0; j<=N; j++)
    {
        fprintf(fpr, "%2d, %11.4e %18.10e %18.10e\n", j+N,

```

```

                (x[j]+1.0)*(length*psi[i+1]*SPACE)*1000.0/2.0+psi[i+1]*SPACE*1000.0,
                US[i+1][j]*TS+Tm, US[i+1][j]);
        }
/*
    fprintf(fpr, "\nTime = %12.6e s ", (dt*(n+1)+nt)*TIME);
    fprintf(fpr, "S = %12.6e mm ",psi[i+1]*SPACE*1000.0);
    fprintf(fpr, "Temp = %16.8e", UL[i+1][0]*TL+Tm);
*/
    return;
}

void print_ex_melt(int N, int n, FILE *fpr, FTYPE x[M], FTYPE US[6][M])
{
    int i, nn;
    nn=n;

    if (n==0)
    {
        for (i=0; i<=N; i++)
        {
            US[0][i]=Ta;
        }
    }
    if (n>=3)
    n=3;
    fprintf(fpr, "\n\n----- U(%d,x)\n", nn);
    fprintf(fpr, "Time = %12.6e s\n", dt*(nn)*TIME);
    fprintf(fpr, "Col. Space of z (mm) Temperature\n\n");

    for (i=0; i<=N; i++)
    {
        fprintf(fpr, "%2d, %11.4e %20.12e\n",
            i, (x[i]+1.0)*length*1000.0/2.0, US[n][i]);
    }
/*
    fprintf(fpr, "\nTime = %12.6e s ", dt*nn*TIME);
    fprintf(fpr, "Temp = %16.8e", US[n][0]);
*/
    return;
}

void solve_equation(FILE *fpr, int method, int N, int *piter, FTYPE relax,
    FTYPE a[M][M], FTYPE c[M], FTYPE U[M])
{

```

```

int      j, k, count, icount;
FTYPE   UG[M], sum1, sum2;
FTYPE   sum_f4;
icount = 0 ;
do
{
    if ((N > 10) && ((icount % 10000) == 0))
        fprintf(fpr, "step: %d, interface: %14.4e\n", icount, U[7]) ;

    ++icount ;
    for (k=0; k<=N; k++)
    {
        UG[k]=U[k];
    }

    for (j=0; j<=N; j++)
    {
        sum1=0.0;
        sum2=0.0;
        for (k=0; k<j; k++)
        {
            if (method==1)                /* Gauss-Seidel method */
                sum1=sum1+a[j][k]*UG[k];

            if (method==2)                /* jacobi method */
                sum1=sum1+a[j][k]*U[k];
        }
        for (k=j; k<=N; k++)
        {
            sum2=sum2+a[j][k]*U[k];
        }
        U[j]=U[j]+relax*(c[j]-sum1-sum2)/a[j][j];
    }
    count=0;
    for (k=0; k<=N; k++)
    {
        if (fabs((U[k]-UG[k])/U[k])<=tolerance)
            count=count+1;
    }
    if ( icount == (*piter) )
        break ;                          /* exit the innermost enclosing loop */
} while (count!=N+1);
*piter = icount ;
return;

```

```

}

void scale_lin_system(int N, FTYPE a[M][M], FTYPE c[M])
{
    int    j, k;
    FTYPE  largest;

    for (j=0; j<=N; j++)
    {
        largest = a[j][0];
        for (k=1; k<=N; k++)
        {
            if (fabs(a[j][k]) > fabs(largest))
                largest=a[j][k];
        }
        for (k=0; k<=N; k++)
        {
            a[j][k]=a[j][k]/largest;
        }
        c[j]=c[j]/largest;
    }
    return;
}

void ex_melting_AB_AM1(int N, int n, int approach, FTYPE x[M], FTYPE D1[M][M],
    FTYPE D2[M][M], FTYPE US[6][M], FILE *fpr)
{
    int    j, k, judge;
    FTYPE  sum1, sum2;
    FTYPE  Uc[M], f[M], Tp, Tc;
    if (n==0 && approach==0)
    {
        for (j=0; j<=N; j++)
        {
            US[0][j]=Ta;
        }
    }
    for (j=1; j<N; j++)
    {
        f[j]=0.0;
        for (k=0; k<=N; k++)
        {
            f[j]=f[j]+g*D2[j][k]*US[n][k];
        }
    }
}

```

```

}
for (j=1; j<N; j++)
{
    US[n+1][j]=US[n][j]+dt*f[j];
}
sum1=0.0;
sum2=0.0;
for (j=1; j<N; j++)
{
    sum1=sum1+D1[N][j]*US[n+1][j];
    sum2=sum2+D1[0][j]*US[n+1][j];
}
US[n+1][N]=g2*Ta-sum1+(D1[N][0]/D1[0][0])*(g1+sum2);
US[n+1][N]=US[n+1][N]/(D1[N][N]+g2-D1[N][0]*D1[0][N]/D1[0][0]);
US[n+1][0]=(-g1-sum2-D1[0][N]*US[n+1][N])/D1[0][0];
for (j=0; j<=N; j++)
{
    Uc[j]=US[n+1][j];
}
do
{
    for (j=0; j<=N; j++)
    {
        US[n+1][j]=Uc[j];
    }
    for (j=1; j<N; j++)
    {
        f[j]=0.0;
        for (k=0; k<=N; k++)
        {
            f[j]=f[j]+g*D2[j][k]*US[n+1][k];
        }
    }
    for (j=1; j<N; j++)
    {
        Uc[j]=US[n][j]+dt*f[j];
    }
    sum1=0.0;
    sum2=0.0;
    for (j=1; j<N; j++)
    {
        sum1=sum1+D1[N][j]*Uc[j];
        sum2=sum2+D1[0][j]*Uc[j];
    }
}

```

```

Uc[N]=g2*Ta-sum1+(D1[N][0]/D1[0][0])*(g1+sum2);
Uc[N]=Uc[N]/(D1[N][N]+g2-D1[N][0]*D1[0][N]/D1[0][0]);
Uc[0]=(-g1-sum2-D1[0][N]*Uc[N])/D1[0][0];
judge=0;
for (j=0; j<=N; j++)
{
    Tc=Uc[j];
    Tp=US[n+1][j];

    if ( ((fabs(Tp)<=(tolerance) && fabs(Tc)<=(tolerance))
        || (fabs(Tc)> (tolerance)
            && fabs(Tp)> (tolerance)
            && fabs((Tc-Tp)/Tc)<tolerance)
        || (fabs(Tp)>= (tolerance)
            && fabs(Tc)<= (tolerance)
            && fabs((Tc-Tp)/Tp)<tolerance)
        || (fabs(Tc)>= (tolerance)
            && fabs(Tp)<= (tolerance)
            && fabs((Tc-Tp)/Tc)<tolerance)))
        judge=judge+1;
}
} while (judge!=N+1);
return;
}

void ex_melting_AB3(int N, FTYPE x[M], FTYPE D1[M][M], FTYPE D2[M][M],
    FTYPE US[6][M], FILE *fpr)
{
    int j, k, judge;
    FTYPE sum1, sum2;
    FTYPE Uc[M], f[5][M], Tp, Tc;
    for (j=1; j<N; j++)
    {
        f[0][j]=0.0;
        f[1][j]=0.0;
        f[2][j]=0.0;
        for (k=0; k<=N; k++)
        {
            f[0][j]=f[0][j]+g*D2[j][k]*US[0][k];
            f[1][j]=f[1][j]+g*D2[j][k]*US[1][k];
            f[2][j]=f[2][j]+g*D2[j][k]*US[2][k];
        }
    }
    for (j=1; j<N; j++)

```

```

{
    US[3][j]=US[2][j]+dt*(23.0*f[2][j]-16.0*f[1][j]+5.0*f[0][j])/12.0;
}
sum1=0.0;
sum2=0.0;
for (j=1; j<N; j++)
{
    sum1=sum1+D1[N][j]*US[3][j];
    sum2=sum2+D1[0][j]*US[3][j];
}
US[3][N]=g2*Ta-sum1+(D1[N][0]/D1[0][0])*(g1+sum2);
US[3][N]=US[3][N]/(D1[N][N]+g2-D1[N][0]*D1[0][N]/D1[0][0]);
US[3][0]=(-g1-sum2-D1[0][N]*US[3][N])/D1[0][0];
for (j=0; j<=N; j++)
{
    Uc[j]=US[3][j];
}
do
{
    for (j=0; j<=N; j++)
    {
        US[3][j]=Uc[j];
    }
    for (j=1; j<N; j++)
    {
        f[3][j]=0.0;
        for (k=0; k<=N; k++)
        {
            f[3][j]=f[3][j]+g*D2[j][k]*US[3][k];
        }
    }
    for (j=1; j<N; j++)
    {
        Uc[j]=US[2][j]+dt*(9.0*f[3][j]+19.0*f[2][j]-5.0*f[1][j]+f[0][j])/24.0;
    }
    sum1=0.0;
    sum2=0.0;
    for (j=1; j<N; j++)
    {
        sum1=sum1+D1[N][j]*Uc[j];
        sum2=sum2+D1[0][j]*Uc[j];
    }
    Uc[N]=g2*Ta-sum1+(D1[N][0]/D1[0][0])*(g1+sum2);
    Uc[N]=Uc[N]/(D1[N][N]+g2-D1[N][0]*D1[0][N]/D1[0][0]);
}

```

```

Uc[0]=(-g1-sum2-D1[0][N]*Uc[N])/D1[0][0];
judge=0;
for (j=0; j<=N; j++)
{
    Tc=Uc[j];
    Tp=US[3][j];
    if ( ((fabs(Tp)<=(tolerance) && fabs(Tc)<=(tolerance))
        || (fabs(Tc)> (tolerance)
            && fabs(Tp)> (tolerance)
            && fabs((Tc-Tp)/Tc)<tolerance)
        || (fabs(Tp)>= (tolerance)
            && fabs(Tc)<= (tolerance)
            && fabs((Tc-Tp)/Tp)<tolerance)
        || (fabs(Tc)>= (tolerance)
            && fabs(Tp)<= (tolerance)
            && fabs((Tc-Tp)/Tc)<tolerance)))
        judge=judge+1;
    }
} while (judge!=N+1);
for (j=0; j<=N; j++)
{
    US[0][j]=US[1][j];
    US[1][j]=US[2][j];
    US[2][j]=US[3][j];
}
return;
}

void critical_point(int N, int n, FTYPE x[M], FTYPE US[6][M], FILE *fpr)
{
    int    j;
    FTYPE  ratio, time_melt;
    ratio=(Tm-US[1][0])/(US[2][0]-US[1][0]);
    time_melt=(n-1)*dt+dt*ratio;
    US[0][N+1]=time_melt;
    for (j=0; j<=N; j++)
    {
        US[2][j]=US[1][j]+(US[2][j]-US[1][j])*ratio;
    }
    fprintf(fpr, "\n\n----- U(%d,x)\n", n);
    fprintf(fpr, "Time = %14.6e s\n", time_melt*TIME);
    fprintf(fpr, "Col. Space of z (mm) Temperature\n\n");
    for (j=0; j<=N; j++)
    {

```



```

        fprintf(fpr, "%2d, %11.4e   %14.6e\n",
                j, (x[j]+1.0)*length*1000.0/2.0, US[2][j]);
    }
    for (j=0; j<=N; j++)
    {
        US[0][j]=(US[2][j]-Tm)/TS;      /* the initial cond. of 2-phase problem */
    }
    return;
}

void TStep_BD1_Kinetics(int N, int n, FTYPE relax, FILE *fpr, FTYPE x[M],
    FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M],
    FTYPE UL[6][M], FTYPE US[6][M])
{
    int      j, k, method, count, iter, is_converged, is_linear;
    FTYPE    aS[M][M], aL[M][M], cS[M], cL[M], USold[M], delta;
    FTYPE    Ui, Uiold;
    FTYPE    srelax, lrelax;
    srelax=relax;
    lrelax=1.0;
    is_converged=0 /* false */
    is_linear=1; /* initially, a linear profile is assumed for UL. Only after
                 convergence is reached with linear profile, the UL system
                 of equations is actually solved */
    initialize_kinetics(N, n, fpr, x, D1, D2, psi, Dpsi, UL, US);
    Ui=US[n+1][0];
    count=0;
    do
    {
        count++;
        cS[0]=0.0;
        for (k=0; k<=N; k++)
        {
            aS[0][k]=2.0*D1[0][k]/((BB-psi[n+1])*AA);
            cS[0]=cS[0]+D1[N][k]*UL[n+1][k];
        }
        cS[0]=2.0*cS[0]/psi[n+1]+Dpsi[n+1];

        for (j=1; j<N; j++)
        {
            for (k=0; k<=N; k++)
            {
                if (j==k) delta=1.0;
                else      delta=0.0;
            }
        }
    }
}

```

```

        aS[j][k]=delta+dt*D1[j][k]*(x[j]-1.0)*Dpsi[n+1]/(BB-psi[n+1])
        -dt*4.0*D2[j][k]/((BB-psi[n+1))*(BB-psi[n+1])*AA);
    }
    cS[j]=US[n][j];
}
for (k=0; k<=N; k++)
{
    aS[N][k]=CC*D1[N][k]/((BB-psi[n+1])*AA);          /* aS(N,k) */
}
aS[N][N]=aS[N][N]+1.0;
cS[N]=IS;
for (k=0; k<=N; k++)
{
    USold[k] = US[n+1][k] ;
}
scale_lin_system(N, aS, cS);
method=1; iter=0;                                     /* Gauss-Seidel method */
solve_equation(fpr, method, N, &iter, srelax, aS, cS, US[n+1]);
Uiold=Ui;
Ui= Uiold * (1.0-lrelax)+lrelax*US[n+1][0] ;
UL[n+1][N]=Ui;
for (k=0; k<=N; k++)
{
    US[n+1][k]=(1.0-lrelax)*USold[k]+lrelax*US[n+1][k] ;
}
Dpsi[n+1]=KV*exp(-KG*Tm/(Ui*TL+Tm));
Dpsi[n+1]=Dpsi[n+1]*(1.0-exp(-KH*Ui*TS/Tm));
psi[n+1]=dt*Dpsi[n+1];
is_converged=(fabs((Ui - Uiold)/Ui)<= 0.0005);
/* convergence is checked. However, this convergence is not the true one
if this convergence was reached with an assumed linear profile for UL */
if (is_linear)
{
    if (is_converged)
    {
        is_linear = 0 ;
        lrelax = 0.01 ;
        is_converged = 0 ;
    }
    else
    {
        UL[n+1][0]=UL[n+1][N]+psi[n+1];

        for (j=0; j<=N; j++)

```

```

        {
            UL[n+1][j]=-0.5*psi[n+1]*(x[j]-1.0)+UL[n+1][N];
        }
    }
}
if ((!is_linear) && (!is_converged))
{
    for (k=0; k<=N; k++)
    {
        aL[0][k]=D1[0][k];
    }
    cL[0]=-0.5*psi[n+1];
    for (j=1; j<N; j++)
    {
        for (k=0; k<=N; k++)
        {
            if (j==k)    delta=1.0;
            else          delta=0.0;
            aL[j][k]=delta - dt*D1[j][k]*(x[j]+1.0)*Dpsi[n+1]/psi[n+1]
                - dt*4.0*D2[j][k]/(psi[n+1]*psi[n+1]);
        }
        cL[j]=UL[n][j];
    }
    for (j=0; j<N; j++)
    {
        cL[j]=cL[j]-aL[j][N]*Ui;
    }
    scale_lin_system(N-1, aL, cL);
    method=1; iter=0;
    solve_equation(fpr, method, N-1, &iter, srelax, aL, cL, UL[n+1]);
}
} while (!is_converged);
return;
}

void TStep_BD3_Kinetics(int N, FTYPE relax, FILE *fpr,
    FTYPE x[M], FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M],
    FTYPE Dpsi[M], FTYPE UL[6][M], FTYPE US[6][M])
{
    int    n, j, k, method, count, iter, is_converged, is_linear;
    FTYPE  aS[M][M], aL[M][M], cS[M], cL[M], USold[M], delta;
    FTYPE  Ui, Uold;
    FTYPE  sum, srelax, lrelax;
    srelax=relax;

```

```

lrelax=1.0 ;
is_converged=0 /* false */
is_linear=1
n=3;
initialize_kinetics(N, n, fpr, x, D1, D2, psi, Dpsi, UL, US) ;
Ui=US[4][0] ;
psi[4]=18.0*psi[3]-9.0*psi[2]+2.0*psi[1];
psi[4]=(psi[4]+6.0*dt*Dpsi[4])/11.0;
count=0;
do
{
    count++;
    cS[0]=0.0;
    for (k=0; k<=N; k++)
    {
        aS[0][k]=2.0*D1[0][k]/((BB-psi[4])*AA);          /* aS(0,k) */
        cS[0]=cS[0]+D1[N][k]*UL[4][k];
    }
    cS[0]=2.0*cS[0]/psi[4]+Dpsi[4];

    for (j=1; j<N; j++)
    {
        for (k=0; k<=N; k++)
        {
            if (j==k)    delta=1.0;
            else          delta=0.0;
            aS[j][k]=delta+(6.0/11.0)*dt*D1[j][k]*(x[j]-1.0)*Dpsi[4]/(BB-psi[4])
                -(6.0/11.0)*dt*4.0*D2[j][k]/((BB-psi[4])*AA);
        }
        cS[j]=(18.0*US[3][j]-9.0*US[2][j]+2.0*US[1][j])/11.0;
    }
    for (k=0; k<=N; k++)
    {
        aS[N][k]=CC*D1[N][k]/((BB-psi[4])*AA);
    }
    aS[N][N]=aS[N][N]+1.0;
    cS[N]=IS;
    for (k=0; k<=N; k++)
    {
        USold[k] = US[4][k] ;
    }
    scale_lin_system(N, aS, cS);
    method=1; iter=0;          /* Gauss-Seidel method */
    solve_equation(fpr, method, N, &iter, srelax, aS, cS, US[4]);
}

```

```

Uold=Ui ;
Ui= Uold*(1.0-lrelax)+lrelax*US[4][0] ;
UL[4][N]=Ui;
for (k=0; k<=N; k++)
{
    US[4][k]=(1.0-lrelax)*USold[k]+lrelax*US[4][k] ;
}
Dpsi[4]=KV*exp(-KG*Tm/(Ui*TL+Tm));
Dpsi[4]=Dpsi[4]*(1.0-exp(-KH*Ui*TS/Tm));
psi[4]=18.0*psi[3]-9.0*psi[2]+2.0*psi[1];
psi[4]=(psi[4]+6.0*dt*Dpsi[4])/11.0;
is_converged=(fabs((Ui - Uold)/Ui)<= 0.0005);
if (is_linear)
{
    if (is_converged)
    {
        is_linear = 0 ;
        lrelax = 0.01 ;
        is_converged = 0 ;
    }
    else
    {
        UL[4][0]=UL[4][N]+psi[4];
        for (j=0; j<=N; j++)
        {
            UL[4][j]=-0.5*psi[4]*(x[j]-1.0)+UL[4][N];
        }
    }
}
if (!(is_linear) && (!is_converged))
{
    for (k=0; k<=N; k++)
    {
        aL[0][k]=D1[0][k];
    }
    cL[0]=-0.5*psi[4];
    for (j=1; j<N; j++)
    {
        for (k=0; k<=N; k++)
        {
            if (j==k)    delta=1.0;
            else        delta=0.0;
            aL[j][k]=delta-(6.0/11.0)*dt*D1[j][k]*(x[j]+1.0)*Dpsi[4]/psi[4]
                -(6.0/11.0)*dt*4.0*D2[j][k]/(psi[4]*psi[4]);
        }
    }
}

```

```

        }
        cL[j]=(18.0*UL[3][j]-9.0*UL[2][j]+2.0*UL[1][j])/11.0;
    }
    for (j=0; j<N; j++)
    {
        cL[j]=cL[j]-aL[j][N]*Ui;
    }
    scale_lin_system(N-1, aL, cL);
    method=1; iter=0;
    solve_equation(fpr, method, N-1, &iter, srelax, aL, cL, UL[4]);
}
} while (!is_converged);
for (j=0; j<=N; j++)
{
    UL[1][j]=UL[2][j];
    UL[2][j]=UL[3][j];
    UL[3][j]=UL[4][j];
    US[1][j]=US[2][j];
    US[2][j]=US[3][j];
    US[3][j]=US[4][j];
}
psi[1]=psi[2];
psi[2]=psi[3];
psi[3]=psi[4];
Dpsi[1]=Dpsi[2];
Dpsi[2]=Dpsi[3];
Dpsi[3]=Dpsi[4];
return;
}

void initialize_kinetics(int N, int n, FILE *fpr, FTYPE x[M], FTYPE D1[M][M],
    FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M], FTYPE UL[6][M],
    FTYPE US[6][M])
{
    int j, k, approach;

    for (j=0; j<=N; j++)
    {
        US[n][j]=US[n][j]*TS+Tm;
    }
    approach=1;
    ex_melting_AB_AM1(N, n, approach, x, D1, D2, US, fpr);
    for (j=0; j<=N; j++)
    {

```

```

        US[n+1][j]=(US[n+1][j]-Tm)/TS;           /* dimensionless*/
        US[n][j]=(US[n][j]-Tm)/TS;             /* dimensionless*/
    }
    UL[n+1][N]=US[n+1][0];
    Dpsi[n+1]=KV*exp(-KG*Tm/(US[n+1][0]*TL+Tm)); /* kinetics condition */
    Dpsi[n+1]=Dpsi[n+1]*(1.0-exp(-KH*US[n+1][0]*TS/Tm));
    psi[n+1]=psi[n]+dt*Dpsi[n+1];             /* BD1 step for psi */
/* assume linear profile in liquid which satisfies free surface BC */
    UL[n+1][0]=UL[n+1][N]+psi[n+1];
    for (j=0; j<=N; j++)
    {
        UL[n+1][j]=-0.5*psi[n+1]*(x[j]-1.0)+UL[n+1][N];
    }
    return ;
}

void TStep_BD1(int N, int n, FTYPE relax, FILE *fpr, FTYPE x[M],
               FTYPE D1[M][M], FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M],
               FTYPE UL[6][M], FTYPE US[6][M])
{
    int j, k, method, icount, iter, is_converged;
    FTYPE aS[M][M], aL[M][M], cS[M], cL[M], delta, ps, Dps;
    FTYPE lrelax, srelax;
    if (n==0) /* ----- access */
    {
        for (k=0; k<=N; k++)
        {
            UL[0][k]=US[0][0];
        }
        psi[0]=0.0;
        Dpsi[0]=1.0 ;
    }
/* ----- access end */
    lrelax=relax;
    srelax=relax;
/* ----- assuming */
    Dpsi[n+1]=Dpsi[n] ;
    psi[n+1]=psi[n]+dt*Dpsi[n] ;
    for (j=0; j<=N; j++)
    {
        UL[n+1][j]=UL[n][j];
        US[n+1][j]=US[n][j];
    }
/* ----- assuming end */
}

```

```

icount=0;
do
{
/*----- organize matrix a(N+1,N+1) & c(N+1) */
icount++;
for (j=1; j<N; j++)
{
for (k=0; k<=N; k++)
{
if (j==k)    delta=1.0;
else        delta=0.0;
aL[j][k]=delta-dt*D1[j][k]*(x[j]+1.0)*Dpsi[n+1]/psi[n+1]
-dt*4.0*D2[j][k]/(psi[n+1]*psi[n+1]);
aS[j][k]=delta+dt*D1[j][k]*(x[j]-1.0)*Dpsi[n+1]/(BB-psi[n+1])
-dt*4.0*D2[j][k]/((BB-psi[n+1])*(BB-psi[n+1])*AA);
}
cL[j]=UL[n][j];
cS[j]=US[n][j];
}
for (k=0; k<=N; k++)
{
aL[0][k]=D1[0][k];
}
cL[0]=-0.5*psi[n+1];
for (k=0; k<N; k++)
{
aL[N][k]=0.0;
aS[0][k+1]=0.0;
}
aL[N][N]=1.0;
aS[0][0]=1.0;
cL[N]=0.0;
cS[0]=0.0;
for (k=0; k<=N; k++)
{
aS[N][k]=CC*D1[N][k]/((BB-psi[n+1])*AA);
}
aS[N][N]=aS[N][N]+1.0;
cS[N]=IS;
scale_lin_system(N, aL, cL);
scale_lin_system(N, aS, cS);
method=1; iter=0;
/* Gauss-Seidel method */
solve_equation(fpr, method, N, &iter, srelax, aL, cL, UL[n+1]);
solve_equation(fpr, method, N, &iter, srelax, aS, cS, US[n+1]);

```



```

/* feedback with eq(4) & eq(5) */
    ps=psi[n+1];
    Dps=Dpsi[n+1];
    Dpsi[n+1]=0.0;
    for (j=0; j<=N; j++)
    {
        Dpsi[n+1]=Dpsi[n+1]-2.0*D1[N][j]*UL[n+1][j]/ps
            +2.0*D1[0][j]*US[n+1][j]/((BB-ps)*AA);
    }
    psi[n+1]=psi[n]+dt*Dpsi[n+1];
    is_converged = !( fabs((ps-psi[n+1])/psi[n+1])>=tolerance
        || fabs((Dps-Dpsi[n+1])/Dpsi[n+1])>=tolerance );
    if (is_converged)
    {
        if (lrelax < 1.0)
        {
            is_converged = 0 ;
            lrelax = 0.5 * lrelax + 0.5 ;
            if (lrelax > 0.9)
                lrelax = 1.0 ;
        }
        else if (srelax < 0.7)
        {
            is_converged = 0 ;
            srelax = 0.5 * srelax + 0.35 ;
            if (srelax > 0.7)
                srelax = 0.7 ;
        }
    }
} while ( !is_converged && (icount < 200));
return;
}

void TStep_BD3(int N, FTYPE relax, FILE *fpr, FTYPE x[M], FTYPE D1[M][M],
    FTYPE D2[M][M], FTYPE psi[M], FTYPE Dpsi[M], FTYPE UL[6][M],
    FTYPE US[6][M])
{
    int j, k, method, icount, iter, is_converged;
    FTYPE aL[M][M], cL[M], aS[M][M], cS[M];
    FTYPE delta, ps, Dps, sum, lrelax, srelax;
    lrelax=relax;
    srelax=relax;
    /* ----- assuming */
    Dpsi[4]=Dpsi[3];

```

```

psi[4]=18.0*psi[3]-9.0*psi[2]+2.0*psi[1];
psi[4]=(psi[4]+6.0*dt*Dpsi[4])/11.0;

for (j=0; j<=N; j++)
{
    UL[4][j]=UL[3][j];
    US[4][j]=US[3][j];
}
/*----- assuming end */
icount=0;
do
{
/*----- organize matrix a(N+1,N+1) & c(N+1) */
    icount++;
    for (j=1; j<N; j++)
    {
        for (k=0; k<=N; k++)
        {
            if (j==k)    delta=1.0;
            else        delta=0.0;
            aL[j][k]=delta-(6.0/11.0)*dt*D1[j][k]*(x[j]+1.0)*Dpsi[4]/psi[4]
                -(6.0/11.0)*dt*4.0*D2[j][k]/(psi[4]*psi[4]);
            aS[j][k]=delta+(6.0/11.0)*dt*D1[j][k]*(x[j]-1.0)*Dpsi[4]/(BB-psi[4])
                -(6.0/11.0)*dt*4.0*D2[j][k]/((BB-psi[4])*(BB-psi[4])*AA);
        }
        cL[j]=(18.0*UL[3][j]-9.0*UL[2][j]+2.0*UL[1][j])/11.0;
        cS[j]=(18.0*US[3][j]-9.0*US[2][j]+2.0*US[1][j])/11.0;
    }
    for (k=0; k<=N; k++)
    {
        aL[0][k]=D1[0][k];
    }
    cL[0]=-0.5*psi[4];

    for (k=0; k<N; k++)
    {
        aL[N][k]=0.0;
        aS[0][k+1]=0.0;
    }
    aL[N][N]=1.0;
    aS[0][0]=1.0;
    cL[N]=0.0;
    cS[0]=0.0;
    for (k=0; k<=N; k++)

```

```

    {
        aS[N][k]=CC*D1[N][k]/((BB-psi[4])*AA);
    }
    aS[N][N]=aS[N][N]+1.0;
    cS[N]=IS;
    scale_lin_system(N, aL, cL);
    scale_lin_system(N, aS, cS);
    method=1; iter=0;
    solve_equation(fpr, method, N, &iter, srelax, aL, cL, UL[4]);
    solve_equation(fpr, method, N, &iter, srelax, aS, cS, US[4]);
    /* ----- feedback with eq(4) & eq(5) */
    ps=psi[4];
    Dps=Dpsi[4];

    Dpsi[4]=0.0;
    for (j=0; j<=N; j++)
    {
        Dpsi[4]=Dpsi[4]-2.0*D1[N][j]*UL[4][j]/ps
            +2.0*D1[0][j]*US[4][j]/((BB-ps)*AA);
    }
    psi[4]=18.0*psi[3]-9.0*psi[2]+2.0*psi[1];
    psi[4]=(psi[4]+6.0*dt*Dpsi[4])/11.0;

    is_converged = !( fabs((ps-psi[4])/psi[4])>=tolerance
        || fabs((Dps-Dpsi[4])/Dpsi[4])>=tolerance );
    if (is_converged)
    {
        if (lrelax < 1.0)
        {
            is_converged = 0 ;
            lrelax = 0.5 * lrelax + 0.5 ;
            if (lrelax > 0.9)
                lrelax = 1.0;
        }
        else if (srelax < 0.7)
        {
            is_converged = 0 ;
            srelax = 0.5 * srelax + 0.35 ;
            if (srelax > 0.7)
                srelax = 0.7 ;
        }
    }
} while ( !is_converged && (icount < 200));
for (j=0; j<=N; j++)

```

```

    {
        UL[1][j]=UL[2][j];
        UL[2][j]=UL[3][j];
        UL[3][j]=UL[4][j];
        US[1][j]=US[2][j];
        US[2][j]=US[3][j];
        US[3][j]=US[4][j];
    }
    psi[1]=psi[2];
    psi[2]=psi[3];
    psi[3]=psi[4];
    Dpsi[1]=Dpsi[2];
    Dpsi[2]=Dpsi[3];
    Dpsi[3]=Dpsi[4];
    return;
}

void end_melt(int N, int n, FTYPE x[M], FTYPE psi[M], FTYPE Dpsi[M],
             FTYPE UL[6][M], FTYPE US[6][M], FILE *fpr)
{
    int j;
    FTYPE ratio, endmelt;

    ratio=(Tm-US[2][N]/(US[3][N]-US[2][N]));
    endmelt=(n-1)*dt+dt*ratio;
    for (j=0; j<=N; j++)
    {
        US[2][j]=US[1][j]+(US[2][j]-US[1][j])*ratio;
    }
    fprintf(fpr, "\n\n----- U(%d,x)\n", n);
    fprintf(fpr, "Time = %14.6e s\n", time_melt*TIME);
    fprintf(fpr, "Col. Space of z (mm) Temperature\n\n");
    for (j=0; j<=N; j++)
    {
        fprintf(fpr, "%2d, %11.4e   %14.6e\n",
                j, (x[j]+1.0)*length*1000.0/2.0, US[2][j]);
    }
    for (j=0; j<=N; j++)
    {
        US[0][j]=(US[2][j]-Tm)/TS;      * the initial cond. of 2-phase problem *
    }
    return;
}
/* done done done done done done done done done done done done done done done */

```

VITA

VITA

Zhaohui Wei was born in Province of Jiangxi, People's Republic of China, on July 9, 1968. He graduated from Chengdu University of Science and Technology, Province of Sichuan, People's Republic of China, with a Bachelor Science in Engineering in 1990. He received a Graduate Research Assistantship (funded by NASA/EPSCoP) in the Department of Mechanical Engineering, University of South Alabama. He will start his Ph.D. study in the fields of thermal fluid sciences in the Department of Mechanical Engineering of the University of Connecticut in January 1998.