

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A NUMERICAL STUDY OF HEAT TRANSFER BEHAVIOR IN WELDING

by

Yasar Vehbi Isiklar

June, 1998

Thesis Advisor:

Ashok Gopinath

Approved for public release; distribution is unlimited.

19980803 032

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 1998		3. REPORT TYPE AND DATES COVERED Master's Thesis
4. TITLE AND SUBTITLE A NUMERICAL STUDY OF HEAT TRANSFER BEHAVIOR IN WELDING			5. FUNDING NUMBERS	
6. AUTHOR (S) Isiklar, Yasar V.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) NAVSEA Carderock, Maryland			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (<i>maximum 200 words</i>) A numerical model has been developed for three-dimensional transient conduction based temperature calculations in underwater wet welding on a thick rectangular plate. The numerical scheme is based on a fully implicit finite volume method. A variable mesh size centered around the moving heat source, and temperature dependent thermal properties have been used in the calculations. Convective, radiative and boiling surface thermal conditions have also been included. The weld pool region itself has been modeled as a solid region of thermal conductivity higher than the surrounding unmelted region. The validity of the results was checked by comparison with Rosenthal's three-dimensional solution for a moving point heat source, and other results in the literature.				
14. SUBJECT TERMS Underwater Wet Welding, Heat Transfer, Finite Volume Method			15. NUMBER OF PAGES 125	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

Approved for public release; distribution is unlimited.

A NUMERICAL STUDY OF HEAT TRANSFER BEHAVIOR IN WELDING

Yasar V. Isiklar
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1992

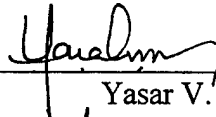
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

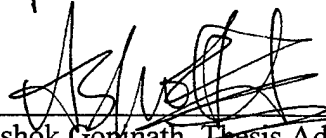
NAVAL POSTGRADUATE SCHOOL
June 1998

Author:

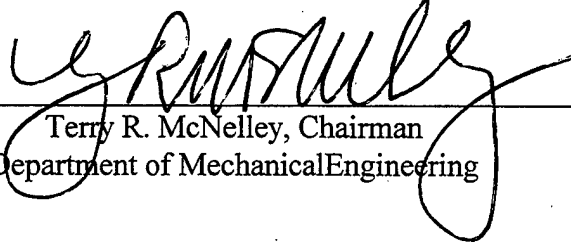


Yasar V. Isiklar

Approved by:



Ashok Gopinath, Thesis Advisor



Terry R. McNelley, Chairman
Department of Mechanical Engineering

ABSTRACT

A numerical model has been developed for three-dimensional transient conduction based temperature calculations in underwater wet welding on a thick rectangular plate. The numerical scheme is based on a fully implicit finite volume method. A variable mesh size centered around the moving heat source, and temperature dependent thermal properties have been used in the calculations. Convective, radiative and boiling surface thermal conditions have also been included. The weld pool region itself has been modeled as a solid region of thermal conductivity higher than the surrounding unmelted solid region. The validity of the results was checked by comparison with Rosenthal's three-dimensional solution for a moving point heat source, and other results in the literature.

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	BACKGROUND.....	3
	A. PREVIOUS STUDIES.....	3
	B. FINITE VOLUME METHOD.....	7
III.	MODEL DEVELOPMENT.....	13
	A. DEFINING THE WORKPIECE.....	13
	B. BOUNDARY CONDITIONS.....	14
	1. Top Face.....	14
	2. Other Faces.....	15
	3. Simulating The Arc.....	16
	4. Boiling Heat Transfer.....	16
	a. Free Convection Regime.....	17
	b. Nucleate Boiling Regime.....	18
	c. Transition Boiling Regime.....	19
	d. Film Boiling Regime.....	20
	C. COEFFICIENTS USED IN THE EQUATIONS.....	22
	D. DERIVATION OF THE EQUATIONS.....	23
IV.	RESULTS AND DISCUSSION.....	45
V.	CONCLUSIONS AND RECOMMENDATIONS.....	61
	APPENDIX A: PROGRAM STRUCTURE.....	63

APPENDIX B: PROGRAM CODES.....	71
LIST OF REFERENCES.....	107
INITIAL DISTRIBUTION LIST.....	111

LIST OF SYMBOLS

SYMBOL	DESCRIPTION	UNITS
c_p	specific heat	J/kg K
$C_{s,f}$	empirical constant	
d	exponential factor	
g	gravitational acceleration	m/s ²
h	convective heat transfer coefficient;	W/m ² K
$\bar{h}$	combined heat transfer coefficient	W/m ² K
$\bar{h}_c$	local heat transfer coeff. convection conductance	W/m ² K
$\bar{h}_r$	average heat transfer coeff. for radiation	W/m ² K
h_{fg}	latent heat of vaporization	J/kg
k	thermal conductivity	W/m K
L	characteristic length	m
q	heat flux	W/m ²
q_o	volumetric heat energy generation rate	W/m ³
Q	heat input to the workpiece	W
r_o	radius of the heat input distribution	m
R	radial distance from the origin	m
t	time	s
T	temperature	K or C
U	welding speed	m/s
V	volume	m ³
α	thermal diffusivity	m ² /s
β	volumetric thermal expansion coefficient	K ⁻¹
δ	distance between two neighboring grid points	m
Δ	difference between values	
ε	surface emissivity	
Θ	finite volume method coefficient	
μ	absolute viscosity	N s/m ²
ν	kinematic viscosity	m ² /s
ρ	density	kg/m ³
σ	Stefan-Boltzman constant	W/m ² K ⁴
σ	surface tension	N/m

ACKNOWLEDGEMENTS

The author would like to gratefully acknowledge the support of NAVSEA for this project.

The author would like especially to express his appreciation to *Prof. Ashok Gopinath* for his expert assistance and creative influence throughout the course of this research. The author also wishes to thank his colleague, *Ltjg. Ibrahim Girgin* for various forms of assistance and computational support.

Finally, the author would like to express his sincere gratitude to his wife *Gönül* and his family for their patience, understanding and encouragement during the preparation of this thesis research.

I. INTRODUCTION

To improve the quality of the underwater welding and to accomplish a reliable, permanent underwater wet welding capability has a great importance in today's industrial and military facilities. With the development of underwater wet welding techniques, the time and the money required for permanent and temporary repairs of ships and other underwater structures can be minimized. Today, the use of hyperbaric welding process obtains a limited quality of welding especially for the construction and repairs of underwater pipelines. In this process a large pressure chamber is used to keep the water away from the workpiece. But, operating this kind of chamber is very expensive and due to the limited geometric size, only a few joint configurations can be enclosed in a chamber [Ref.1]. The other welding techniques such as double shielding and flux shielding also use the water removing theory from the arc area during welding. But, the working area must be completely prevented from water for satisfactory welding results [Ref. 2].

Currently, underwater wet welding is used for the temporary repair needs. Because of their poor quality compared to surface (air) welds (they obtain 80% of the tensile strength and 50% ductility of the surface welds [Ref. 3].), they must be replaced as soon as possible. Therefore, the development of a more efficient wet welding technique is the only solution to this problem.

The surrounding water environment during wet welding causes rapid cooling and steep temperature gradients in the weld area behind the arc [Ref.4]. Because of the

extremely complex nature of the heat transfer phenomena between heated surface and the water environment, a numerical model simulation is necessary.

In the present study, a numerical model has been developed for transient, three-dimensional conduction heat transfer in underwater welding process on a thick rectangular plate. The numerical scheme was based on fully implicit finite volume model, including convection, radiation and boiling surface thermal boundary conditions. The different regimes of boiling were accounted for on the surface. A variable mesh size centered around an arc source moving at constant speed was used to determine temperature variations inside and around the weld pool. The weld pool region itself has been modeled as a solid region of thermal conductivity higher than the surrounding unmelted solid region. The input data from the previous studies based on different methods were used to check the accuracy and the validity of the numerical method.

II. BACKGROUND

A. PREVIOUS STUDIES

Rosenthal did the most important early work on the theory of the effect of moving sources of heat in the late of 1930s. He studied the fundamentals of this theory and derived equations for two-dimensional and three-dimensional heat conduction in a solid when a moving source is in use [Ref. 5,6]. The analytical solution derived by Rosenthal was based on the principle of a quasi-stationary thermal state. The quasi-stationary thermal state represents a steady thermal response of the weldment with respect to the moving coordinates. In other words, the origin moves with the heat source, and for an observer at origin, the temperature distribution and the pool geometry do not change with time [Ref. 6,7]. Rosenthal's solution for three-dimensional heat flow during welding is as follows [Ref. 6]:

$$\frac{2\pi(T - T_0)k_s R}{Q} = \exp\left(\frac{-U(R - x)}{2\alpha_s}\right) \quad (2.1)$$

where

R = radial distance from origin $(x^2 + y^2 + z^2)^{1/2}$

T = temperature,

T_0 = workpiece temperature before welding,

k_s = thermal conductivity of solid,

Q = heat input to the workpiece,

U = welding speed,
 x = the distance that the heat source traveled,
 α_s = thermal diffusivity of solid (i.e., $k_s / \rho C_s$, where ρ and C_s are density and specific heat of solid, respectively).

The assumptions used by Rosenthal to derive the equation above are as follows

[Ref. 7]:

1. Point heat source.
2. No melting and negligible heat of fusion.
3. Constant thermal properties.
4. No heat loss from the workpiece surface.
5. Infinitely wide workpiece.

Although Rosenthal's assumptions help to simplify the mathematical analysis involved, there are some significant deviations between theoretical and experimental results. For instance, as a result of the point heat source assumption, the temperature at the weld centerline goes to infinity even though the power of the heat source is finite. Also, the values of thermal properties change with the temperature and neglecting the heat fusion gives considerable errors. [Ref. 7]

Tanaka did the first studies on the application of the mathematical analysis of non-stationary heat flow to the practical problems. Naka and Masubuchi studied the mathematical analysis of non-stationary heat flow. They used dimensional expressions to make the numerical calculations simple. Nippes and Savage studied the cooling rates of heat affected zones by using a graphical approach method. Suzuki found an analytical-empirical method of studying the effects of welding parameters and determined the cooling rate from welding conditions with the help of a monograph. [Ref.6]

Adams derived new equations from Rosenthal's equation by using the fusion line as a boundary condition and calculated the peak temperature at a distance from the fusion boundary at the weldment surface [Ref. 7]. His equation for three-dimensional heat flow is as follows:

$$\frac{1}{T_p - T_0} = \frac{5.44\pi k_s \alpha_s}{QU} \left[2 + \left(\frac{UY}{2\alpha_s} \right)^2 \right] + \frac{1}{T_m - T_0} \quad (2.2)$$

where all the terms are defined in equation (2.1) except

T_p = peak temperature at a distance Y from the fusion boundary,
 T_m = melting temperature,

Christensen also used analytical-empirical approach and derived dimensionless equations based on Rosenthal's three-dimensional equation. He explained the relationships between the welding conditions and the weld bead geometry. [Ref. 7]

Recently, numerical analysis methods and computer programs have been commonly used to develop the previous assumptions. In 1965, The Battelle Institute Geneva Laboratory in Switzerland conducted a computer-aided study about analyzing of heat flow in weldments [Ref. 6]. The University of Wisconsin and McDonnell Douglas Aircraft Company also conducted similar studies in the same year [Ref. 6]. In 1970, M.I.T. researchers studied heat flow during underwater welding. They also developed finite element programs on heat flow during welding [Ref. 6]. Oreper and Szekeley examined stationary, axisymmetric TIG (tungsten-inert-gas) welding process with a

moving boundary by using finite difference method. Their formulation contained the affects of transient conduction, electromagnetic, buoyancy and surface tension forces [Ref. 8]. Kou and Wang performed computational studies of the GTA welding process. They presented a computer simulation of three-dimensional convection for an arc source moving at constant speed. They considered electromagnetic, buoyancy and surface tension forces on the pool surface. They found very good agreement between the calculated and observed fusion boundaries [Ref. 9]. They also studied the computer simulation of three-dimensional convection in laser melting by considering the buoyancy force and the surface tension gradient at the weld pool surface [Ref. 10]. Correa and Sundell studied axisymmetric stationary arc source by using different grid sizes for computation of flow and electromagnetic fields [Ref. 11]. Saedi and Unkel developed a thermal-fluid model of the weld pool. Their model was based on the stationary arc. To describe the weld pool geometry, they matched the convective and the conductive heat fluxes at the weld boundary by using an iterative calculation method [Ref. 12]. Zacharia et al. made three-dimensional calculations on the effects of the heat source in the stationary GTAW process. He indicated that a depressed area formed at the weld pool center because of an outward fluid flow caused by surface tension force [Ref. 13]. Kim and Na developed a model on heat and mass flow for stationary, GTAW process with electromagnetic, buoyancy and surface tension forces. They used numerical mapping method for calculations [Ref. 14]. Ramanan and Korpela compared the effects of thermo-capillary and Lorentz forces on the flow pattern in a stationary weld pool with buoyancy forces. They used multi-grid methods and a local grid refinement technique with

axisymmetric stationary arc source [Ref. 14]. Ule, Joshi and Sedy determined three-dimensional transient temperature variations in the GTAW process by using the explicit finite difference method. They used different mesh sizes and temperature dependent thermal properties. They also considered convective and radiative surface thermal conditions during calculations [Ref. 16]. Kanouff and Greif studied the unsteady development of an axisymmetric arc weld pool in GTAW process. They used moving grids to follow the phase change boundary and considered the effects of Marangoni, Lorentz and buoyancy forces in the calculations [Ref. 17]. Joshi, Dutta, Schupp and Espinosa developed a three-dimensional numerical model to describe the flow circulation phenomena in aluminum weld pools under non-axisymmetric Lorentz force field [Ref. 18,19].

B. FINITE VOLUME METHOD

The finite volume method is one of the simple and well-established Computational Fluid Dynamics (CFD) techniques that were originally developed as a special finite difference method [Ref. 19]. The stages of the numerical algorithm in this method are as follows [Ref.19]:

1. Formal integration of the governing equations of fluid flow over all the finite control volumes of the solution domain.
2. Discretisation involves the substitution of a variety of finite difference type approximations for the terms in the integrated equation representing flow process such as convection, diffusion and sources. This converts the integral equations into a system of algebraic equations.
3. Solution of the algebraic equations by an iterative method.

In the control volume integration, the calculation domain is divided into discrete control volumes. There is only one control volume surrounding each grid point and the boundaries of the control volumes are positioned side to side in the middle of the distance between the grid points. Integrating the governing differential equation over each control volume derives the resulting discretised equations. The discretised equations express the conservation of quantities such as mass, momentum and energy for each control volume. This characteristic exists for any number of grid points. The numerical solution insures the validity of the conservation principle over the whole calculation domain for the related quantities. Versteeg and Malalasekera [Ref. 20] wrote " This clear relationship between the numerical algorithm and the underlying physical conservation principle forms makes finite volume method much simpler to understand by engineers than finite element and spectral methods". To understand the finite volume method better an illustrative example can be given for one-dimensional steady state heat conduction situation. [Ref. 20,21,22]

1. One dimensional steady state heat conduction

The governing equation for one-dimensional steady state heat conduction is

$$\frac{d}{dx} \left(k \frac{dT}{dx} \right) + S = 0 \quad (2.3)$$

where

k = thermal conductivity,

T = temperature,

S = the rate of heat generation per unit volume.

The domain is divided into small and nonoverlapping control volumes. A part of one-dimensional grids generated is shown in Figure 2.1. Here, The grid point under construction is denoted as P and the neighboring nodes to the east and west are denoted as E and W respectively. The lower case letters e and w denotes the east and west faces of the control volume. The distance between the nodes W and P is denoted by δx_{WP} and between P and E is denoted by δx_{PE} . The distances between w and P and between P and e are given by δx_{wP} and δx_{Pe} respectively. The control volume width is shown as $\Delta x = \delta x_{we}$

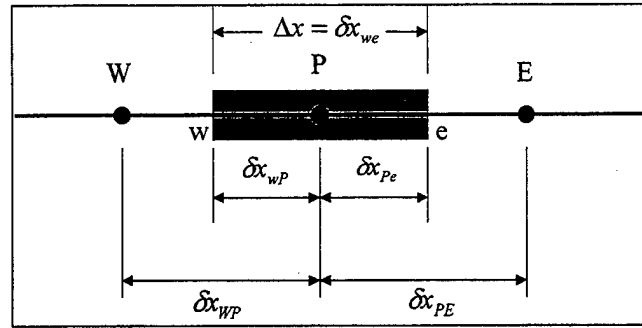


Figure 2.1 One-dimensional grid [Ref. 20]

Integrating equation (2.3) over the control volume gives,

$$\left(k \frac{dT}{dx} \right)_e - \left(k \frac{dT}{dx} \right)_w + \int_w^e S dx = 0 \quad (2.4)$$

By assuming a piecewise-linear profile assumption (Figure 2.2), equation (2.4) can be evaluated as

$$k_e \left(\frac{T_E - T_P}{\delta x_{PE}} \right) - k_w \left(\frac{T_P - T_W}{\delta x_{WP}} \right) + \bar{S} \Delta x = 0 \quad (2.5)$$

where $\bar{S}$ is the average value of S over the control volume.

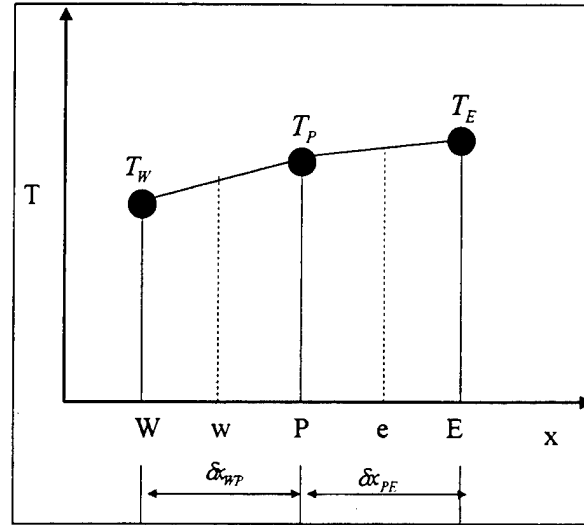


Figure 2.2 Piecewise –linear Profile [Ref. 22]

By arranging the terms, the final discretised equation can be written as

$$a_P T_P = a_E T_E + a_W T_W + b \quad (2.6)$$

where

$$a_E = \frac{k_e}{\delta x_{PE}} \quad (2.7a)$$

$$a_w = \frac{k_w}{\delta x_{WP}} \quad (2.7b)$$

$$a_p = a_E + a_w \quad (2.7c)$$

$$b = \bar{S} \Delta x \quad (2.7d)$$

III. MODEL DEVELOPMENT

A. DEFINING THE WORKPIECE

The workpiece has been defined as a thick rectangular plate (Figure 3.1).

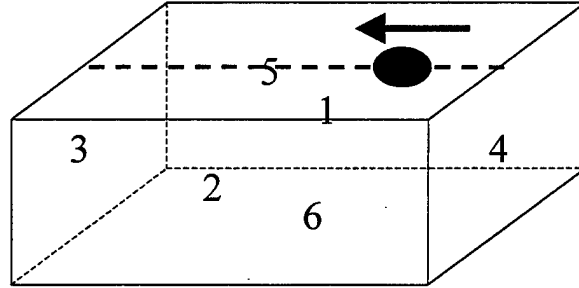


Figure 3.1 The workpiece

where

1. Right Lateral Face (East)
2. Left Lateral Face (West)
3. Back Face (North)
4. Front Face (South)
5. Top Face (Top)
6. Bottom Face (Bottom)

Convective heat transfer coefficients for each face have been defined as follows:

- Right-Lateral Face: h_e
Left-Lateral Face : h_w
Back Face : h_n
Front Face : h_s
Top Face : h_t
Bottom Face : h_b

The same method has been also used to define thermal conductivity and heat flux terms for each face.

B. BOUNDARY CONDITIONS

1. Top Face

By adding radiative and convective heat loss from the top face, boundary condition equation can be defined as

$$-k \frac{dT}{dx} = h(T_{wall} - T_{\infty}) + q'' + \sigma \varepsilon (T_{wall}^4 - T_{sur}^4) \quad (3.1)$$

In equation (3.1), for the radiation term, T_{sur} can be neglected. ($T_{wall}^4 \gg T_{sur}^4$). The distance between the node point P and the face can be taken as $\Delta x / 2$, where the distance between the node point P and the neighboring nodes is Δx . Again, for the radiation term, because of the small distance between the node point P and the face, by assuming $T_{wall} \cong T_P$, we have

$$-k \frac{(T_{wall} - T_P)}{\Delta x / 2} = h(T_{wall} - T_{\infty}) + q'' + \sigma \varepsilon T_P^4 \quad (3.2)$$

Equation (3.2) can be opened as

$$-\frac{2k}{\Delta x} T_{wall} + \frac{2k}{\Delta x} T_P = h T_{wall} - h T_{\infty} + q'' + \sigma \varepsilon T_P^4 \quad (3.3)$$

Equation (3.3) can be arranged as

$$T_{wall} \left(\frac{2k}{\Delta x} + h \right) = \frac{2k}{\Delta x} T_p + hT_{\infty} - q'' + \sigma \epsilon T_p^4 \quad (3.4)$$

Equation can be written as

$$T_{wall} = \frac{\frac{2k}{\Delta x} T_p + hT_{\infty} - q'' + \sigma \epsilon T_p^4}{\frac{2k}{\Delta x} + h} \quad (3.5)$$

The unknown temperature for the top face points is found as

$$T_{wall} = \frac{2kT_p + h\Delta x T_{\infty} - q'' \Delta x - \sigma \epsilon \Delta x T_p^4}{2k + h\Delta x} \quad (3.6)$$

where

$$q''_{top} = q''_{source} + q''_i + q''_{boiling} \quad (3.7)$$

q''_{source} = arc source heat flux,

q''_i = a constant arbitrary heat flux which may be applied to the top face,

$q''_{boiling}$ = boiling heat flux,

2. Other Faces

By using the same method from the top face (without radiation),

$$T_{wall} = \frac{2kT_p + h\Delta x T_\infty - q'' \Delta x}{2k + h\Delta x} \quad (3.8)$$

3. Simulating the Arc

To simulate the heat input from the arc to the workpiece, it is assumed that the heat input distribution of the arc have a Gaussian distribution on the top face of the workpiece. The general equation is

$$Q = q_0 \int_0^\infty e^{-\frac{d}{r_0^2} r^2} 2\pi r dr \quad (3.9)$$

where

Q = the total heat input into the workpiece,
 q_0 = the volumetric energy generation rate,
 r_0 = the radius of the heat input distribution,
 d = the exponential factor,

By solving equation (3.9), the volumetric energy generation rate can be found as

$$q_0 = \frac{Qd}{\pi r_0^2} \quad (3.10)$$

4. Boiling Heat Transfer

Modes or regimes of boiling and the related equations can be classified as follows

(where $\Delta T_e = T_s - T_{sat}$):

a. Free Convection Regime ($\Delta T_e \leq 5^\circ C$)

In this regime, natural convection effects determine the heat transfer between the heating surface and surrounding liquid. Recommended correlations for upper surface of heated plate are [Ref. 25]

$$\overline{Nu}_L = 0.54 Ra_L^{1/4} \quad (10^4 \leq Ra_L \leq 10^7) \quad (3.11)$$

$$\overline{Nu}_L = 0.15 Ra_L^{1/3} \quad (10^7 \leq Ra_L \leq 10^{11}) \quad (3.12)$$

where the Rayleigh number ,

$$Ra_L = \frac{g\beta(T_s - T_\infty)L^3}{\nu\alpha} \quad (3.13)$$

here

g = gravitational acceleration, m/s^2

β = volumetric thermal expansion coefficient, K^{-1}

ν = kinematic viscosity, m^2/s

α = thermal diffusivity, m^2/s

L = characteristic length, $L \equiv \text{Plate surface area } (A_s) / \text{Perimeter } (P)$

and

$$\bar{h} = \frac{\overline{Nu}_L k}{L} \quad (3.14)$$

and the value of heat flux is,

$$q'' = \bar{h}(T_s - T_\infty) \quad (3.15)$$

b. Nucleate Boiling Regime ($5^\circ\text{C} < \Delta T_e \leq 30^\circ\text{C}$)

The most useful nucleate pool boiling correlating equation was developed by Rohsenow [Ref. 23, 24, 25,],

$$q_s'' = \mu_l h_{fg} \left[\frac{g(\rho_l - \rho_v)}{\sigma} \right]^{1/2} \left(\frac{c_{p,l} \Delta T_e}{C_{s,f} h_{fg} \text{Pr}_l^n} \right)^3 \quad (3.16)$$

where the subscripts s, l, and v express surface, saturated liquid state and vapor state. The definition of each term in equation (3.16) are as follows:

- μ_l = viscosity of the liquid, kg/ms
- h_{fg} = latent heat of vaporization, J/kg
- g = gravitational acceleration, m/s^2
- ρ_l = density of the saturated liquid, kg/m^3
- ρ_v = density of the saturated vapor, kg/m^3
- σ = surface tension of the liquid-to-vapor interface, N/m
- $c_{p,l}$ = specific heat of saturated liquid, J/kgK
- $\Delta T_e = T_s - T_{sat}$
- Pr_l = Prandtl number of the saturated liquid
- n = 1.0 for water, 1.7 for other fluids
- $C_{s,f}$ = empirical constant that depends on the nature of the heating surface fluid combination and whose numerical value varies from system to system

But, in underwater welding, the surrounding water temperature is below the saturation temperature (between 0°C and 30°C). This is called as the heat transfer to a subcooled liquid. For the subcooled boiling, the heat flux can be estimated as [Ref. 23]

$$q'' = q_s'' \left\{ 1 + \left[\frac{2k_l (T_{sat} - T_{liquid})}{\sqrt{\pi \alpha_l \tau}} \right] \frac{24}{\pi h_{fg} \rho_v} \left[\frac{\rho_v^2}{\sigma g (\rho_l - \rho_v)} \right]^{1/4} \right\} \quad (3.17)$$

where

k_l = thermal conductivity of the liquid, W/m.K

α_l = thermal diffusivity of the liquid ($k / \rho c_p$), m^2/s and,

$$\tau = \frac{\pi}{3} \sqrt{2\pi} \left[\frac{\sigma}{g (\rho_l - \rho_v)} \right]^{1/2} \left[\frac{\rho_v^2}{\sigma g (\rho_l - \rho_v)} \right]^{1/4} \quad (3.18)$$

c. Transition Boiling Regime ($30^\circ C < \Delta T_e \leq 120^\circ C$)

For the transition-boiling regime, no sufficient theory has been derived.

This regime is between the maximum and minimum heat fluxes where [Ref. 23,25],

$$q''_{max} = 0.149 h_{fg} \rho_v \left[\frac{\sigma g (\rho_l - \rho_v)}{\rho_v^2} \right]^{1/4} \quad (3.19)$$

$$q''_{min} = 0.09 \rho_v h_{fg} \left[\frac{g \sigma (\rho_l - \rho_v)}{(\rho_l + \rho_v)^2} \right]^{1/4} \quad (3.20)$$

By assuming a linear heat flux distribution in the transition boiling regime

(Figure 3.2), it can be written,

$$\frac{\log(q''_{max}) - \log(q''_{min})}{\log 30 - \log 120} = \frac{\log(q'') - \log(q''_{min})}{\log \Delta T_e - \log 120} \quad (3.21)$$

by arranging equation (3.21),

$$\log(q'') = \frac{\log\left(\frac{q''_{\max}}{q''_{\min}}\right)}{\log\left(\frac{30}{120}\right)} \log\left(\frac{\Delta T_e}{120}\right) + \log(q''_{\min}) \quad (3.22)$$

and the heat flux at a point in transition boiling regime can be found as

$$q'' = 10^{\log(q'')} \quad (3.23)$$

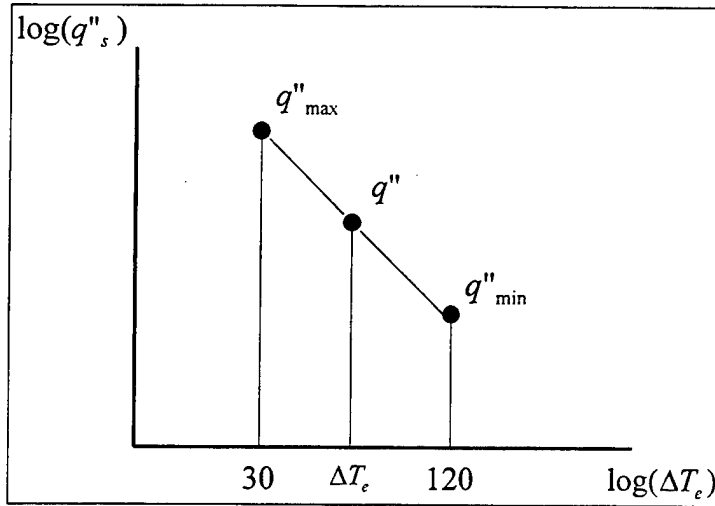


Figure 3.2 Linear Distribution of Heat Flux in Transition Regime

d. Film Boiling Regime ($\Delta T_e > 120^\circ\text{C}$)

In film boiling regime for flat horizontal surfaces, Westwater and Breen recommended the following correlation for conduction heat transfer coefficient [Ref.23],

$$\bar{h}_c = 0.59 \left\{ \frac{g(\rho_l - \rho_v) \rho_v k_v^3 [h_{fg} + 0.68 c_{pv} \Delta T_e]}{\lambda \mu_v \Delta T_e} \right\}^{1/4} \quad (3.24)$$

where

μ_v = viscosity of the vapor, kg/ms

k_v = thermal conductivity of vapor, W/mK

c_{pv} = specific heat of saturated vapor, J/kgK and,

$$\lambda = 2\pi \left[\frac{\sigma}{g(\rho_l - \rho_v)} \right]^{1/2} \quad (3.25)$$

Bromley suggested combining conduction and radiation heat transfer coefficients

[Ref. 23],

$$\bar{h}_{total} = \bar{h}_c + 0.75 \bar{h}_r \quad (3.26)$$

where

$$\bar{h}_r = \sigma \varepsilon_s \left(\frac{T_s^4 - T_{sat}^4}{T_s - T_{sat}} \right) \quad (3.27)$$

here

ε_s = surface emissivity

T_s = absolute surface temperature

and the resulting heat flux in this regime is found as

$$q'' = \bar{h}_{total} \Delta T_e \quad (3.28)$$

C. COEFFICIENTS USED IN THE EQUATIONS

The coefficients used deriving the discretised equations are as follows:

$$a_E = \frac{k_e A_e}{\delta x_{PE}} \quad a_W = \frac{k_w A_w}{\delta x_{WP}} \quad a_N = \frac{k_n A_n}{\delta y_{PN}}$$

$$a_S = \frac{k_s A_s}{\delta y_{SP}} \quad a_T = \frac{k_t A_t}{\delta z_{PT}} \quad a_B = \frac{k_b A_b}{\delta z_{BP}}$$

$$a_P^0 = \rho c \frac{\Delta V}{\Delta t} \quad a_P = \Theta(a_E + a_W + a_N + a_S + a_T + a_B) + a_P^0$$

$$coeff_e = \frac{h_e \delta x_{PE}}{2k_e + h_e \delta x_{PE}} \quad coeff_w = \frac{h_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}}$$

$$coeff_n = \frac{h_n \delta y_{PN}}{2k_n + h_n \delta y_{PN}} \quad coeff_s = \frac{h_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}}$$

$$coeff_t = \frac{h_t \delta z_{PT}}{2k_t + h_t \delta z_{PT}} \quad coeff_b = \frac{h_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}}$$

$$fluxcoeff_e = \frac{\delta x_{PE}}{2k_e + h_e \delta x_{PE}} \quad fluxcoeff_w = \frac{\delta x_{WP}}{2k_w + h_w \delta x_{WP}}$$

$$fluxcoeffn = \frac{\delta y_{PN}}{2k_n + h_n \delta y_{PN}}$$

$$fluxcoeffs = \frac{\delta y_{SP}}{2k_s + h_s \delta y_{SP}}$$

$$fluxcoefft = \frac{\delta z_{PT}}{2k_t + h_t \delta z_{PT}}$$

$$fluxcoeffb = \frac{\delta z_{BP}}{2k_b + h_b \delta z_{BP}}$$

$$radcoeff = \frac{\sigma \epsilon \delta z_{PT}}{2k_t + h_t \delta z_{PT}}$$

D. DERIVATION OF THE EQUATIONS

The problem is governed by the equation

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) + \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) + \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) \quad (3.29)$$

1. Interior Nodes

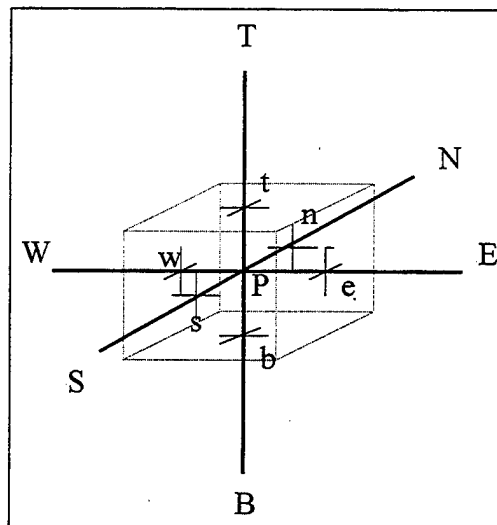


Figure 3.3 Interior Nodes

Integration of equation (3.29) over the control volume and over a time interval from t to $t + \Delta t$ gives

$$\begin{aligned} \int_t^{t+\Delta t} \int_{CV} \rho c \frac{\partial T}{\partial t} dV dt &= \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) dV dt + \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) dV dt \\ &+ \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) dV dt \end{aligned} \quad (3.30)$$

This may be written as

$$\begin{aligned} \int_t^{t+\Delta t} \int_{CV} \rho c \frac{\partial T}{\partial t} dV dt &= \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial x} \left(k \frac{\partial T}{\partial x} \right) A dx dt + \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial y} \left(k \frac{\partial T}{\partial y} \right) A dy dt \\ &+ \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial z} \left(k \frac{\partial T}{\partial z} \right) A dz dt \end{aligned} \quad (3.31)$$

where A is the face area of the control volume and dx , dy , dz are the dimensions of the control volume. Equation (3.31) may be written as

$$\begin{aligned} \int_{CV} \left[\int_t^{t+\Delta t} \rho c \frac{\partial T}{\partial t} dt \right] dV &= \int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial x} \right)_e - \left(kA \frac{\partial T}{\partial x} \right)_w \right] dt + \int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial y} \right)_n - \left(kA \frac{\partial T}{\partial y} \right)_s \right] dt \\ &+ \int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial z} \right)_t - \left(kA \frac{\partial T}{\partial z} \right)_b \right] dt \end{aligned} \quad (3.32)$$

By assuming the grid point value of the temperature at a node to prevail over the whole control volume, the left hand side of equation (3.32) can be written as

$$\int_{CV} \left[\int_t^{t+\Delta t} \rho c \frac{\partial T}{\partial t} dt \right] dV = \rho c (T_p - T_p^0) \Delta V \quad (3.33)$$

where T_p^0 refers to temperatures at time t and T_p refers to temperatures at time $t + \Delta t$.

By applying central differencing to the diffusion terms on the right hand side equation (3.32) can be written as

$$\begin{aligned} \rho c (T_p - T_p^0) \Delta V = & \int_t^{t+\Delta t} \left[\left(k_e A_e \frac{T_E - T_p}{\delta_{PE}} \right) - \left(k_w A_w \frac{T_p - T_W}{\delta_{WP}} \right) \right] dt + \int_t^{t+\Delta t} \left[\left(k_n A_n \frac{T_N - T_p}{\delta_{PN}} \right) - \left(k_s A_s \frac{T_p - T_S}{\delta_{SP}} \right) \right] dt \\ & + \int_t^{t+\Delta t} \left[\left(k_t A_t \frac{T_T - T_p}{\delta_{PT}} \right) - \left(k_b A_b \frac{T_p - T_B}{\delta_{BP}} \right) \right] dt \end{aligned} \quad (3.34)$$

The values of $T_p, T_E, T_W, T_N, T_S, T_T$ and T_B vary with time. The time integral can be calculated by using temperatures at time t or at time $t + \Delta t$ or, a combination of temperatures at time t and $t + \Delta t$. This approach may be generalized by defining a weighting parameter Θ between 0 and 1.

$$I_T = \int_t^{t+\Delta t} T_p dt = [\Theta T_p + (1 - \Theta) T_p^0] \Delta t \quad (3.35)$$

In equation (3.35) for $\Theta=0$ the temperature at time level t is used (Explicit scheme); for $\Theta=0.5$ the temperatures at t and $t + \Delta t$ are equally weighted (Crank-Nicholson scheme); for $\Theta=1$ the temperature at time level $t + \Delta t$ is used (Fully implicit scheme). By using equation (3.35) and dividing Δt throughout, equation (3.34) may be arranged as

$$\begin{aligned}
\rho \frac{(T_P - T_P^0)}{\Delta t} \Delta V = & \Theta \left[k_e A_e \frac{(T_E - T_P)}{\delta x_{PE}} - k_w A_w \frac{(T_P - T_W)}{\delta x_{WP}} + k_n A_n \frac{(T_N - T_P)}{\delta y_{PN}} - k_s A_s \frac{(T_P - T_S)}{\delta y_{SP}} \right. \\
& + k_t A_t \frac{(T_T - T_P)}{\delta z_{PT}} - k_b A_b \frac{(T_P - T_B)}{\delta z_{BP}} \left. \right] + (1 - \Theta) \left[k_e A_e \frac{(T_E^0 - T_P^0)}{\delta x_{PE}} \right. \\
& - k_w A_w \frac{(T_P^0 - T_W^0)}{\delta x_{WP}} + k_n A_n \frac{(T_N^0 - T_P^0)}{\delta y_{PN}} - k_s A_s \frac{(T_P^0 - T_S^0)}{\delta y_{SP}} \\
& \left. + k_t A_t \frac{(T_T^0 - T_P^0)}{\delta z_{PT}} - k_b A_b \frac{(T_P^0 - T_B^0)}{\delta z_{BP}} \right] \quad (3.36)
\end{aligned}$$

Equation (3.36) may be re-arranged as

$$\begin{aligned}
\left[\rho \frac{\Delta V}{\Delta t} + \Theta \left(\frac{k_e A_e}{\delta x_{PE}} + \frac{k_w A_w}{\delta x_{WP}} + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right) \right] T_P = & \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E \\
& + (1 - \Theta) T_E^0] + \frac{k_w A_w}{\delta x_{WP}} [\Theta T_W + (1 - \Theta) T_W^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1 - \Theta) T_N^0] + \frac{k_s A_s}{\delta y_{SP}} [\Theta T_S + (1 - \Theta) T_S^0] \\
& + \frac{k_t A_t}{\delta z_{PT}} [\Theta T_T + (1 - \Theta) T_T^0] + \frac{k_b A_b}{\delta z_{BP}} [\Theta T_B + (1 - \Theta) T_B^0] + \left[\rho \frac{\Delta V}{\Delta t} - (1 - \Theta) \left(\frac{k_e A_e}{\delta x_{PE}} + \frac{k_w A_w}{\delta x_{WP}} \right. \right. \\
& \left. \left. + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right) \right] T_P^0 \quad (3.37)
\end{aligned}$$

By putting the defined coefficients from the previous section, equation (3.37) may be written as

$$\begin{aligned}
 & \left[a_P^0 + \Theta(a_E + a_W + a_N + a_S + a_T + a_B) \right] T_P = a_E \left[\Theta T_E + (1 - \Theta) T_E^0 \right] \\
 & + a_W \left[\Theta T_W + (1 - \Theta) T_W^0 \right] + a_N \left[\Theta T_N + (1 - \Theta) T_N^0 \right] + a_S \left[\Theta T_S + (1 - \Theta) T_S^0 \right] \\
 & + a_T \left[\Theta T_T + (1 - \Theta) T_T^0 \right] + a_B \left[\Theta T_B + (1 - \Theta) T_B^0 \right] \\
 & + \left[a_P^0 - (1 - \Theta)(a_E + a_W + a_N + a_S + a_T + a_B) \right] T_P^0
 \end{aligned} \tag{3.38}$$

Finally, by grouping the known and the unknown terms at each side, the discretised equation is found as

$$\begin{aligned}
 & a_P T_P - \Theta a_E T_E - \Theta a_W T_W - \Theta a_N T_N - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) \\
 & \left[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0 \right] + \left[a_P^0 - (1 - \Theta)(a_E + a_W \right. \\
 & \left. + a_N + a_S + a_T + a_B) \right] T_P^0
 \end{aligned} \tag{3.39}$$

2. Left-Front-Top Corner

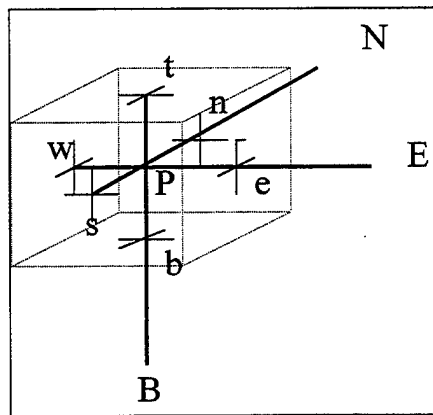


Figure 3.4 Left-Front-Top Corner

By using the same method from equations (3.30-3.33), equation (3.29) can be written as

$$\begin{aligned} \rho(T_P - T_P^0)\Delta V = & \int_t^{t+\Delta t} \left[\left(k_e A_e \frac{T_E - T_P}{\delta x_{PE}} \right) - \left(k_w A_w \frac{T_P - T_W}{\delta x_{WP}} \right) \right] dt + \int_t^{t+\Delta t} \left[\left(k_n A_n \frac{T_N - T_P}{\delta y_{PN}} \right) - \left(k_s A_s \frac{T_P - T_S}{\delta y_{SP}} \right) \right] dt \\ & + \int_t^{t+\Delta t} \left[\left(k_t A_t \frac{T_T - T_P}{\delta z_{PT}} \right) - \left(k_b A_b \frac{T_P - T_B}{\delta z_{BP}} \right) \right] dt \end{aligned} \quad (3.40)$$

In equation (3.40), T_W and T_S take the value of equation (3.8) and T_T takes the value of equation (3.6). By using these values and equation (3.35), we have

$$\begin{aligned} \rho \frac{(T_P - T_P^0)}{\Delta t} \Delta V = & \Theta \left[\frac{k_e A_e}{\delta x_{PE}} (T_E - T_P) - \frac{2k_w A_w}{\delta x_{WP}} \left(T_P - \frac{2k_w T_P + h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\ & + \frac{k_n A_n}{\delta y_{PN}} (T_N - T_P) - \frac{2k_s A_s}{\delta y_{SP}} \left(T_P - \frac{2k_s T_P + h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \\ & + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{2k_t T_P + h_t \delta z_{PT} T_\infty - q''_{top} \delta z_{PT} - \sigma \varepsilon \delta z_{PT} T_P^4}{2k_t + h_t \delta z_{PT}} - T_P \right) - \frac{k_b A_b}{\delta z_{BP}} (T_P - T_B) \Big] \\ & + (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} (T_E^0 - T_P^0) - \frac{2k_w A_w}{\delta x_{WP}} \left(T_P^0 - \frac{2k_w T_P^0 + h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\ & + \frac{k_n A_n}{\delta y_{PN}} (T_N^0 - T_P^0) - \frac{2k_s A_s}{\delta y_{SP}} \left(T_P^0 - \frac{2k_s T_P^0 + h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \\ & + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{2k_t T_P^0 + h_t \delta z_{PT} T_\infty - q''_{top} \delta z_{PT} - \sigma \varepsilon \delta z_{PT} T_P^4}{2k_t + h_t \delta z_{PT}} - T_P^0 \right) - \frac{k_b A_b}{\delta z_{BP}} (T_P^0 - T_B^0) \Big] \end{aligned} \quad (3.41)$$

Equation (3.41) may be arranged as

$$\begin{aligned}
& \left[\rho \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(1 - \frac{2k_w}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} + \frac{2k_s A_s}{\delta y_{SP}} \left(1 - \frac{2k_s}{2k_s + h_s \delta y_{SP}} \right) \right. \right. \\
& \left. \left. + \frac{2k_t A_t}{\delta z_{PT}} \left(1 - \frac{2k_t}{2k_t + h_t \delta z_{PT}} \right) + \frac{k_b A_b}{\delta z_{BP}} \right] T_P = \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1-\Theta) T_E^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1-\Theta) T_N^0] \right. \\
& \left. + \frac{k_b A_b}{\delta z_{BP}} [\Theta T_B + (1-\Theta) T_B^0] + \left[\rho \frac{\Delta V}{\Delta t} - (1-\Theta) \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(1 - \frac{2k_w}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} \right. \right. \right. \\
& \left. \left. + \frac{2k_s A_s}{\delta y_{SP}} \left(1 - \frac{2k_s}{2k_s + h_s \delta y_{SP}} \right) + \frac{2k_t A_t}{\delta z_{PT}} \left(1 - \frac{2k_t}{2k_t + h_t \delta z_{PT}} \right) + \frac{k_b A_b}{\delta z_{BP}} \right] T_P^0 \right. \\
& \left. + \Theta \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right. \right. \\
& \left. \left. + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{h_t \delta z_{PT} T_\infty - q''_{iop} \delta z_{PT} - \sigma \varepsilon \delta z_{PT} T_P^4}{2k_t + h_t \delta z_{PT}} \right) + (1-\Theta) \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \right. \right. \\
& \left. \left. + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{h_t \delta z_{PT} T_\infty - q''_{iop} \delta z_{PT} - \sigma \varepsilon \delta z_{PT} T_P^4}{2k_t + h_t \delta z_{PT}} \right) \right] \right] \quad (3.42)
\end{aligned}$$

Equation (3.42) may be re-arranged as

$$\begin{aligned}
& \left[\rho \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right. \right. \\
& \left. \left. + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{h_t \delta z_{PT}}{2k_t + h_t \delta z_{PT}} \right) + \frac{k_b A_b}{\delta z_{BP}} \right] T_P = \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1-\Theta) T_E^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1-\Theta) T_N^0] \right. \\
& \left. + \frac{k_b A_b}{\delta z_{BP}} [\Theta T_B + (1-\Theta) T_B^0] + \left[\rho \frac{\Delta V}{\Delta t} - (1-\Theta) \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \right. \right. \\
& \left. \left. + \frac{k_n A_n}{\delta y_{PN}} + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{h_t \delta z_{PT}}{2k_t + h_t \delta z_{PT}} \right) + \frac{k_b A_b}{\delta z_{BP}} \right] T_P^0 + \Theta \left[\frac{2k_w A_w}{\delta x_{WP}} \right. \right. \\
& \left. \left(\frac{h_w \delta x_{WP} T_\infty}{2k_w + h_w \delta x_{WP}} - \frac{q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty}{2k_s + h_s \delta y_{SP}} - \frac{q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{2k_t A_t}{\delta z_{PT}} \right.
\end{aligned}$$

$$\begin{aligned}
& \left(\frac{h_i \delta z_{PT} T_\infty}{2k_i + h_i \delta z_{PT}} - \frac{q''_{top} \delta z_{PT}}{2k_i + h_i \delta z_{PT}} - \frac{\sigma \varepsilon \delta z_{PT} T_p^4}{2k_i + h_i \delta z_{PT}} \right) + (1 - \Theta) \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty}{2k_w + h_w \delta x_{WP}} \right. \right. \\
& \left. \left. - \frac{q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty}{2k_s + h_s \delta y_{SP}} - \frac{q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{2k_t A_t}{\delta z_{PT}} \left(\frac{h_t \delta z_{PT} T_\infty}{2k_t + h_t \delta z_{PT}} \right. \right. \\
& \left. \left. - \frac{q''_{top} \delta z_{PT}}{2k_t + h_t \delta z_{PT}} - \frac{\sigma \varepsilon \delta z_{PT} T_p^4}{2k_t + h_t \delta z_{PT}} \right) \right] \quad (3.43)
\end{aligned}$$

By putting the defined coefficients from the previous section, equation (3.43) may be arranged as

$$\begin{aligned}
& [a_p^0 + \Theta[a_E + 2a_w(\text{coeff}w) + a_N + 2a_s(\text{coeff}s) + 2a_t(\text{coeff}t) + a_B]]T_p \\
& = a_E[\Theta T_E + (1 - \Theta)T_E^0] + a_N[\Theta T_N + (1 - \Theta)T_N^0] + a_B[\Theta T_B + (1 - \Theta)T_B^0] \\
& + [a_p^0 - (1 - \Theta)[a_E + 2a_w(\text{coeff}w) + a_N + 2a_s(\text{coeff}s) \\
& + 2a_t(\text{coeff}t) + a_B]]T_p^0 + \Theta[2a_w[(\text{coeff}w)T_\infty - (\text{fluxcoeff}w)q''_w] \\
& + 2a_s[(\text{coeff}s)T_\infty - (\text{fluxcoeff}s)q''_s] \\
& + 2a_t[(\text{coeff}t)T_\infty - (\text{fluxcoeff}t)q''_{top} - (\text{radcoeff}t)T_p^4] \\
& + (1 - \Theta)[2a_w[(\text{coeff}w)T_\infty - (\text{fluxcoeff}w)q''_w] \\
& + 2a_s[(\text{coeff}s)T_\infty - (\text{fluxcoeff}s)q''_s] \\
& + 2a_t[(\text{coeff}t)T_\infty - (\text{fluxcoeff}t)q''_{top} - (\text{radcoeff}t)T_p^4]] \quad (3.44)
\end{aligned}$$

by putting heat flux terms from equation (3.7), equation (3.42) may be written as

$$\begin{aligned}
& [a_p^0 + \Theta[a_E + 2a_w(\text{coeff}w) + a_N + 2a_s(\text{coeff}s) + 2a_t(\text{coeff}t) + a_B]]T_p \\
& = a_E[\Theta T_E + (1 - \Theta)T_E^0] + a_N[\Theta T_N + (1 - \Theta)T_N^0] + a_B[\Theta T_B + (1 - \Theta)T_B^0] \\
& + [a_p^0 - (1 - \Theta)[a_E + 2a_w(\text{coeff}w) + a_N + 2a_s(\text{coeff}s) \\
& + 2a_t(\text{coeff}t) + a_B]]T_p^0 + [2a_w(\text{coeff}w) + 2a_s(\text{coeff}s) + 2a_t(\text{coeff}t)]T_\infty \\
& - 2a_w(\text{fluxcoeff}w)q''_w - 2a_s(\text{fluxcoeff}s)q''_s - 2a_t(\text{fluxcoeff}t)q''_i
\end{aligned}$$

$$-2a_T(\text{fluxcoeff})q''_{\text{source}} - 2a_T(\text{fluxcoeff})q''_{\text{boiling}} - 2a_T(\text{radcoeff})T_p^4 \quad (3.45)$$

and, the discretised equation may be written as

$$\begin{aligned} & \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + 2a_T(\text{coeff}t) + a_B] \right] T_p \\ & - \Theta a_E T_E - \Theta a_N T_N - \Theta a_B T_B = (1 - \Theta) [a_E T_E^0 + a_N T_N^0 + a_B T_B^0] \\ & + \left[a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + 2a_T(\text{coeff}t) + a_B] \right] T_p^0 \\ & + [2a_W(\text{coeff}w) + 2a_S(\text{coeff}s) + 2a_T(\text{coeff}t)] T_\infty - 2a_W(\text{fluxcoeff}w)q''_w \\ & - 2a_S(\text{fluxcoeff}s)q''_s - 2a_T(\text{fluxcoeff}t)q''_t - 2a_T(\text{fluxcoeff})q''_{\text{source}} \\ & - 2a_T(\text{fluxcoeff})q''_{\text{boiling}} - 2a_T(\text{radcoeff})T_p^4 \end{aligned} \quad (3.46)$$

3. Front-Left Edge

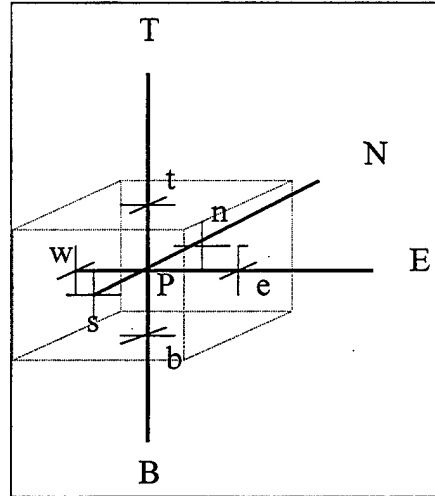


Figure 3.5 Front-Left Edge

By using equations (3.28-3.31), equation (3.27) can be written as

$$\begin{aligned}
\rho c(T_P - T_P^0)\Delta V = & \int_t^{t+\Delta t} \left[\left(k_e A_e \frac{T_E - T_P}{\delta x_{PE}} \right) - \left(k_w A_w \frac{T_P - T_W}{\delta x_{WP}} \right) \right] dt + \int_t^{t+\Delta t} \left[\left(k_n A_n \frac{T_N - T_P}{\delta y_{PN}} \right) - \left(k_s A_s \frac{T_P - T_S}{\delta y_{SP}} \right) \right] dt \\
& + \int_t^{t+\Delta t} \left[\left(k_t A_t \frac{T_T - T_P}{\delta z_{PT}} \right) - \left(k_b A_b \frac{T_P - T_B}{\delta z_{BP}} \right) \right] dt
\end{aligned} \tag{3.47}$$

In equation (3.47), T_W and T_S take the value of equation (3.8). By using these values and equation (3.35), we have

$$\begin{aligned}
\rho c \frac{(T_P - T_P^0)}{\Delta t} \Delta V = & \Theta \left[\frac{k_e A_e}{\delta x_{PE}} (T_E - T_P) - \frac{2k_w A_w}{\delta x_{WP}} \left(T_P - \frac{2k_w T_P + h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\
& + \frac{k_n A_n}{\delta y_{PN}} (T_N - T_P) - \frac{2k_s A_s}{\delta y_{SP}} \left(T_P - \frac{2k_s T_P + h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{k_t A_t}{\delta z_{PT}} (T_T - T_P) \\
& \left. - \frac{k_b A_b}{\delta z_{BP}} (T_P - T_B) \right] + (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} (T_E^0 - T_P^0) - \frac{2k_w A_w}{\delta x_{WP}} \left(T_P^0 - \frac{2k_w T_P^0 + h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\
& + \frac{k_n A_n}{\delta y_{PN}} (T_N^0 - T_P^0) - \frac{2k_s A_s}{\delta y_{SP}} \left(T_P^0 - \frac{2k_s T_P^0 + h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \\
& \left. + \frac{k_t A_t}{\delta z_{PT}} (T_T^0 - T_P^0) - \frac{k_b A_b}{\delta z_{BP}} (T_P^0 - T_B^0) \right]
\end{aligned} \tag{3.48}$$

Equation (3.48) may be arranged as

$$\begin{aligned}
& \left[\rho c \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(1 - \frac{2k_w}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} \right. \right. \\
& \left. \left. + \frac{2k_s A_s}{\delta y_{SP}} \left(1 - \frac{2k_s}{2k_s + h_s \delta y_{SP}} \right) + \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right] \right] T_P = \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1 - \Theta) T_E^0]
\end{aligned}$$

$$\begin{aligned}
& + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1 - \Theta) T_N^0] + \frac{k_t A_t}{\delta z_{PT}} [\Theta T_T + (1 - \Theta) T_T^0] + \frac{k_b A_b}{\delta z_{BP}} [\Theta T_B + (1 - \Theta) T_B^0] \\
& + \left[\rho c \frac{\Delta V}{\Delta t} - (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(1 - \frac{2k_w}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} \right. \right. \\
& + \left. \frac{2k_s A_s}{\delta y_{SP}} \left(1 - \frac{2k_s}{2k_s + h_s \delta y_{SP}} \right) + \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right] T_P^0 + \Theta \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\
& + \left. \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right] + (1 - \Theta) \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty - q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\
& + \left. \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty - q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right] \quad (3.49)
\end{aligned}$$

Equation (3.49) may be re-arranged as

$$\begin{aligned}
& \left[\rho c \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right. \right. \\
& + \left. \left. \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right] \right] T_P = \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1 - \Theta) T_E^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1 - \Theta) T_N^0] \\
& + \frac{k_t A_t}{\delta z_{PT}} [\Theta T_T + (1 - \Theta) T_T^0] + \frac{k_b A_b}{\delta z_{BP}} [\Theta T_B + (1 - \Theta) T_B^0] + \left[\rho c \frac{\Delta V}{\Delta t} - (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} \right. \right. \\
& + \left. \frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{k_n A_n}{\delta y_{PN}} + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) + \frac{k_t A_t}{\delta z_{PT}} + \frac{k_b A_b}{\delta z_{BP}} \right] T_P^0 \\
& + \Theta \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty}{2k_w + h_w \delta x_{WP}} - \frac{q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) + \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty}{2k_s + h_s \delta y_{SP}} \right. \right. \\
& - \left. \left. \frac{q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right] + (1 - \Theta) \left[\frac{2k_w A_w}{\delta x_{WP}} \left(\frac{h_w \delta x_{WP} T_\infty}{2k_w + h_w \delta x_{WP}} - \frac{q''_w \delta x_{WP}}{2k_w + h_w \delta x_{WP}} \right) \right. \\
& + \left. \frac{2k_s A_s}{\delta y_{SP}} \left(\frac{h_s \delta y_{SP} T_\infty}{2k_s + h_s \delta y_{SP}} - \frac{q''_s \delta y_{SP}}{2k_s + h_s \delta y_{SP}} \right) \right] \quad (3.50)
\end{aligned}$$

By putting the coefficients from the previous section, equation (3.50) may be written as

$$\begin{aligned}
& \left[a_P^0 + \Theta[a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + a_T + a_B] \right] T_P \\
&= a_E \left[\Theta T_E + (1 - \Theta) T_E^0 \right] + a_N \left[\Theta T_N + (1 - \Theta) T_N^0 \right] + a_T \left[\Theta T_T + (1 - \Theta) T_T^0 \right] \\
&+ a_B \left[\Theta T_B + (1 - \Theta) T_B^0 \right] + \left[a_P^0 - (1 - \Theta)[a_E + 2a_W(\text{coeff}w) + a_N \right. \\
&+ 2a_S(\text{coeff}s) + a_T + a_B] T_P^0 + \Theta[2a_W[(\text{coeff}w)T_\infty - (\text{fluxcoeff}w)q''_w] \\
&+ 2a_S[(\text{coeff}s)T_\infty - (\text{fluxcoeff}s)q''_s] + (1 - \Theta)[2a_W[(\text{coeff}w)T_\infty - (\text{fluxcoeff}w)q''_w] \\
&+ 2a_S[(\text{coeff}s)T_\infty - (\text{fluxcoeff}s)q''_s] \quad (3.51)
\end{aligned}$$

Equation (3.51) may be arranged as

$$\begin{aligned}
& \left[a_P^0 + \Theta[a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + a_T + a_B] \right] T_P \\
&= a_E \left[\Theta T_E + (1 - \Theta) T_E^0 \right] + a_N \left[\Theta T_N + (1 - \Theta) T_N^0 \right] + a_T \left[\Theta T_T + (1 - \Theta) T_T^0 \right] \\
&+ a_B \left[\Theta T_B + (1 - \Theta) T_B^0 \right] + \left[a_P^0 - (1 - \Theta)[a_E + 2a_W(\text{coeff}w) + a_N \right. \\
&+ 2a_S(\text{coeff}s) + a_T + a_B] T_P^0 + [2a_W(\text{coeff}w) + 2a_S(\text{coeff}s)] T_\infty \\
&- 2a_W(\text{fluxcoeff}w)q''_w - 2a_S(\text{fluxcoeff}s)q''_s \quad (3.52)
\end{aligned}$$

Finally, the discretised equation may be written as

$$\begin{aligned}
& \left[a_P^0 + \Theta[a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + a_T + a_B] \right] T_P - \Theta a_E T_E - \Theta a_N T_N \\
&- \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta)[a_E T_E^0 + a_N T_N^0 + a_T T_T^0 + a_B T_B^0] + \left[a_P^0 - (1 - \Theta)[a_E \right. \\
&+ 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + a_T + a_B] T_P^0 + [2a_W(\text{coeff}w) + 2a_S(\text{coeff}s)] T_\infty \\
&- 2a_W(\text{fluxcoeff}w)q''_w - 2a_S(\text{fluxcoeff}s)q''_s \quad (3.53)
\end{aligned}$$

4. Bottom Face

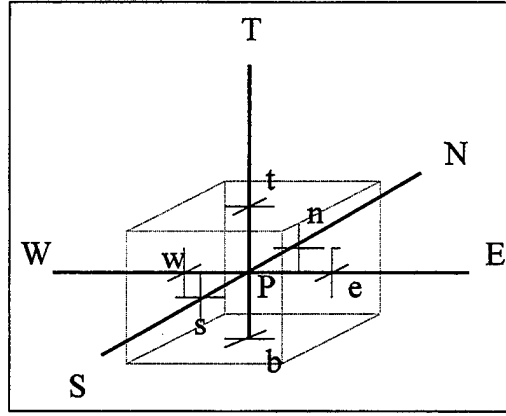


Figure 3.6 Bottom Face

By using equation (3.30-3.33), equation (3.29) may be written as

$$\begin{aligned} \rho c (T_P - T_P^0) \Delta V = & \int_t^{t+\Delta t} \left[\left(k_e A_e \frac{T_E - T_P}{\delta x_{PE}} \right) - \left(k_w A_w \frac{T_P - T_W}{\delta x_{WP}} \right) \right] dt + \int_t^{t+\Delta t} \left[\left(k_n A_n \frac{T_N - T_P}{\delta y_{PN}} \right) - \left(k_s A_s \frac{T_P - T_S}{\delta y_{SP}} \right) \right] dt \\ & + \int_t^{t+\Delta t} \left[\left(k_t A_t \frac{T_T - T_P}{\delta z_{PT}} \right) - \left(k_b A_b \frac{T_P - T_B}{\delta z_{BP}} \right) \right] dt \end{aligned} \quad (3.54)$$

In equation (3.54), T_B takes the value of equation (3.8). By applying this and equation (3.35) to equation (3.54), we have

$$\rho c \frac{(T_P - T_P^0)}{\Delta t} \Delta V = \Theta \left[\frac{k_e A_e}{\delta x_{PE}} (T_E - T_P) - \frac{k_w A_w}{\delta x_{WP}} (T_P - T_W) + \frac{k_n A_n}{\delta y_{PN}} (T_N - T_P) \right]$$

$$\begin{aligned}
& -\frac{k_s A_s}{\delta y_{SP}}(T_P - T_S) + \frac{k_t A_t}{\delta z_{PT}}(T_T - T_P) - \frac{2k_b A_b}{\delta z_{BP}} \left(T_P - \frac{2k_b T_P + h_b \delta z_{BP} T_\infty - q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \Bigg] \\
& + (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}}(T_E^0 - T_P^0) - \frac{k_w A_w}{\delta x_{WP}}(T_P^0 - T_W^0) + \frac{k_n A_n}{\delta y_{PN}}(T_N^0 - T_P^0) - \frac{k_s A_s}{\delta y_{SP}}(T_P^0 - T_S^0) \right. \\
& \left. + \frac{k_t A_t}{\delta z_{PT}}(T_T^0 - T_P^0) - \frac{2k_b A_b}{\delta z_{BP}} \left(T_P^0 - \frac{2k_b T_P^0 + h_b \delta z_{BP} T_\infty - q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] \quad (3.55)
\end{aligned}$$

Equation (3.55) may be arranged as

$$\begin{aligned}
& \left[\rho c \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{k_w A_w}{\delta x_{WP}} + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{2k_b A_b}{\delta z_{BP}} \left(1 - \frac{2k_b}{2k_b + h_b \delta y_{BP}} \right) \right] T_P \right. \\
& = \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1 - \Theta) T_E^0] + \frac{k_w A_w}{\delta x_{WP}} [\Theta T_W + (1 - \Theta) T_W^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1 - \Theta) T_N^0] \\
& + \frac{k_s A_s}{\delta y_{SP}} [\Theta T_S + (1 - \Theta) T_S^0] + \frac{k_t A_t}{\delta z_{PT}} [\Theta T_T + (1 - \Theta) T_T^0] + \left[\rho c \frac{\Delta V}{\Delta t} - (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} \right. \right. \\
& + \frac{k_w A_w}{\delta x_{WP}} + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{2k_b A_b}{\delta z_{BP}} \left(1 - \frac{2k_b}{2k_b + h_b \delta y_{BP}} \right) \Bigg] T_P^0 \\
& \left. + \Theta \left[\frac{2k_b A_b}{\delta z_{BP}} \left(\frac{h_b \delta z_{BP} T_\infty - q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] + (1 - \Theta) \left[\frac{2k_b A_b}{\delta z_{BP}} \left(\frac{h_b \delta z_{BP} T_\infty - q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] \right] \quad (3.56)
\end{aligned}$$

Equation (3.56) may be re-arranged as

$$\left[\rho c \frac{\Delta V}{\Delta t} + \Theta \left[\frac{k_e A_e}{\delta x_{PE}} + \frac{k_w A_w}{\delta x_{WP}} + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{2k_b A_b}{\delta z_{BP}} \left(\frac{h_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] \right] T_P$$

$$\begin{aligned}
&= \frac{k_e A_e}{\delta x_{PE}} [\Theta T_E + (1 - \Theta) T_E^0] + \frac{k_w A_w}{\delta x_{WP}} [\Theta T_W + (1 - \Theta) T_W^0] + \frac{k_n A_n}{\delta y_{PN}} [\Theta T_N + (1 - \Theta) T_N^0] \\
&+ \frac{k_s A_s}{\delta y_{SP}} [\Theta T_S + (1 - \Theta) T_S^0] + \frac{k_t A_t}{\delta z_{PT}} [\Theta T_T + (1 - \Theta) T_T^0] + \left[\rho c \frac{\Delta V}{\Delta t} - (1 - \Theta) \left[\frac{k_e A_e}{\delta x_{PE}} \right. \right. \\
&+ \frac{k_w A_w}{\delta x_{WP}} + \frac{k_n A_n}{\delta y_{PN}} + \frac{k_s A_s}{\delta y_{SP}} + \frac{k_t A_t}{\delta z_{PT}} + \frac{2k_b A_b}{\delta z_{BP}} \left(\frac{h_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \left. \right] T_P^0 + \Theta \left[\frac{2k_b A_b}{\delta z_{BP}} \right. \\
&\left. \left(\frac{h_b \delta z_{BP} T_\infty}{2k_b + h_b \delta z_{BP}} - \frac{q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] + (1 - \Theta) \left[\frac{2k_b A_b}{\delta z_{BP}} \left(\frac{h_b \delta z_{BP} T_\infty}{2k_b + h_b \delta z_{BP}} \right. \right. \\
&\left. \left. - \frac{q''_b \delta z_{BP}}{2k_b + h_b \delta z_{BP}} \right) \right] \quad (3.57)
\end{aligned}$$

By using the defined coefficients, equation (3.57) may be written as

$$\begin{aligned}
&[a_P^0 + \Theta[a_E + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb})]]T_P = a_E[\Theta T_E + (1 - \Theta)T_E^0] \\
&+ a_W[\Theta T_W + (1 - \Theta)T_W^0] + a_N[\Theta T_N + (1 - \Theta)T_N^0] + a_S[\Theta T_S + (1 - \Theta)T_S^0] \\
&+ a_T[\Theta T_T + (1 - \Theta)T_T^0] + [a_P^0 - (1 - \Theta)[a_E + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb})]]T_P^0 \\
&+ \Theta[2a_B[(\text{coeffb})T_\infty] - ((\text{fluxcoeffb})q''_b)] \\
&+ (1 - \Theta)[2a_B[(\text{coeffb})T_\infty] - ((\text{fluxcoeffb})q''_b)] \quad (3.58)
\end{aligned}$$

And, the discretised equation may be written as

$$\begin{aligned}
&[a_P^0 + \Theta[a_E + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb})]]T_P - \Theta a_E T_E - \Theta a_W T_W - \Theta a_N T_N \\
&- \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta)[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0] + [a_P^0 \\
&- (1 - \Theta)[a_E + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb})]]T_P^0 + [2a_B(\text{coeffb})]T_\infty \\
&- 2a_B(\text{fluxcoeffb})q''_b \quad (3.59)
\end{aligned}$$

The resulting discretised equations for the other parts of the workpiece are as follows

5. Left-Back-Top Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + 2a_N(\text{coeff}n) + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P - \Theta a_E T_E \\
& - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta) [a_E T_E^0 + a_S T_S^0 + a_B T_B^0] + [a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) \\
& + 2a_N(\text{coeff}n) + a_S + 2a_T(\text{coeff}t) + a_B] T_P^0 + [2a_W(\text{coeff}w) + 2a_N(\text{coeff}n) \\
& + 2a_T(\text{coeff}t)] T_\infty - 2a_W(\text{fluxcoeff}w) q''_w - 2a_N(\text{fluxcoeff}n) q''_n - 2a_T(\text{fluxcoeff}t) q''_t \\
& - 2a_T(\text{fluxcoeff}t) q''_{\text{source}} - 2a_T(\text{fluxcoeff}t) q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff}) T_P^4
\end{aligned} \tag{3.60}$$

6. Right-Back-Top Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeff}e) + a_W + 2a_N(\text{coeff}n) + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P - \Theta a_W T_W \\
& - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta) [a_W T_W^0 + a_S T_S^0 + a_B T_B^0] + [a_p^0 - (1 - \Theta) [2a_E(\text{coeff}e) \\
& + a_W + 2a_N(\text{coeff}n) + a_S + 2a_T(\text{coeff}t) + a_B] T_P^0 + [2a_E(\text{coeff}e) + 2a_N(\text{coeff}n) \\
& + 2a_T(\text{coeff}t)] T_\infty - 2a_E(\text{fluxcoeff}e) q''_e - 2a_N(\text{fluxcoeff}n) q''_n - 2a_T(\text{fluxcoeff}t) q''_t \\
& - 2a_T(\text{fluxcoeff}t) q''_{\text{source}} - 2a_T(\text{fluxcoeff}t) q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff}) T_P^4
\end{aligned} \tag{3.61}$$

7. Right-Front-Top Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeff}e) + a_W + a_N + 2a_S(\text{coeff}s) + 2a_T(\text{coeff}t) + a_B] \right] T_P - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_B T_B = (1 - \Theta) [a_W T_W^0 + a_N T_N^0 + a_B T_B^0] + [a_p^0 - (1 - \Theta) [2a_E(\text{coeff}e) \\
& + a_W + a_N + 2a_S(\text{coeff}s) + 2a_T(\text{coeff}t) + a_B] T_P^0 + [2a_E(\text{coeff}e) + 2a_S(\text{coeff}s) \\
& + 2a_T(\text{coeff}t)] T_\infty - 2a_E(\text{fluxcoeff}e) q''_e - 2a_S(\text{fluxcoeff}s) q''_s - 2a_T(\text{fluxcoeff}t) q''_t \\
& - 2a_T(\text{fluxcoeff}t) q''_{\text{source}} - 2a_T(\text{fluxcoeff}t) q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff}) T_P^4
\end{aligned} \tag{3.62}$$

8. Left-Back-Bottom Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + 2a_N(\text{coeff}n) + a_S + a_T + 2a_B(\text{coeff}b)] \right] T_P - \Theta a_E T_E \\
& - \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta) [a_E T_E^0 + a_S T_S^0 + a_T T_T^0] + [a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) \\
& + 2a_N(\text{coeff}n) + a_S + a_T + 2a_B(\text{coeff}b)] T_P^0 \\
& + [2a_W(\text{coeff}w) + 2a_N(\text{coeff}n) + 2a_B(\text{coeff}b)] T_\infty - 2a_W(\text{fluxcoeff}w) q''_w \\
& - 2a_N(\text{fluxcoeff}n) q''_n - 2a_B(\text{fluxcoeff}b) q''_b
\end{aligned} \tag{3.63}$$

9. Right-Back-Bottom Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeff}e) + a_W + 2a_N(\text{coeff}n) + a_S + a_T + 2a_B(\text{coeff}b)] \right] T_P - \Theta a_W T_W \\
& - \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta) [a_W T_W^0 + a_S T_S^0 + a_T T_T^0] + [a_p^0 - (1 - \Theta) [2a_E(\text{coeff}e) \\
& + a_W + 2a_N(\text{coeff}n) + a_S + a_T + 2a_B(\text{coeff}b)] T_P^0 \\
& + [2a_E(\text{coeff}e) + 2a_N(\text{coeff}n) + 2a_B(\text{coeff}b)] T_\infty - 2a_E(\text{fluxcoeff}e) q''_e \\
& - 2a_N(\text{fluxcoeff}n) q''_n - 2a_B(\text{fluxcoeff}b) q''_b
\end{aligned} \tag{3.64}$$

10. Left-Front-Bottom Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + a_N + 2a_S(\text{coeff}s) + a_T + 2a_B(\text{coeff}b)] \right] T_P - \Theta a_E T_E \\
& - \Theta a_N T_N - \Theta a_T T_T = (1 - \Theta) [a_E T_E^0 + a_N T_N^0 + a_T T_T^0] + [a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) \\
& + a_N + 2a_S(\text{coeff}s) + a_T + 2a_B(\text{coeff}b)] T_P^0 \\
& + [2a_W(\text{coeff}w) + 2a_S(\text{coeff}s) + 2a_B(\text{coeff}b)] T_\infty - 2a_W(\text{fluxcoeff}w) q''_w \\
& - 2a_S(\text{fluxcoeff}s) q''_s - 2a_B(\text{fluxcoeff}b) q''_b
\end{aligned} \tag{3.65}$$

11. Right-Front-Bottom Corner

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeff}e) + a_W + a_N + 2a_S(\text{coeff}s) + a_T + 2a_B(\text{coeff}b)] \right] T_P - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_T T_T = (1 - \Theta) [a_W T_W^0 + a_N T_N^0 + a_T T_T^0] + [a_p^0 - (1 - \Theta) [2a_E(\text{coeff}e) \\
& + a_W + a_N + 2a_S(\text{coeff}s) + a_T + 2a_B(\text{coeff}b)] T_P^0 \\
& + [2a_E(\text{coeff}e) + 2a_S(\text{coeff}s) + 2a_B(\text{coeff}b)] T_\infty - 2a_E(\text{fluxcoeff}e) q''_e \\
& - 2a_S(\text{fluxcoeff}s) q''_s + 2a_B(\text{fluxcoeff}b) q''_b
\end{aligned} \tag{3.66}$$

12. Top Face

$$\begin{aligned}
& \left[a_p^0 + \Theta[a_E + a_W + a_N + a_S + 2a_T(\text{coeffi}) + a_B] \right] T_P - \Theta a_E T_E - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta) \left[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_B T_B^0 \right] \\
& + \left[a_p^0 - (1 - \Theta)[a_E + a_W + a_N + a_S + 2a_T(\text{coeffi}) + a_B] \right] T_P^0 \\
& + [2a_T(\text{coeffi})] T_\infty - 2a_T(\text{fluxcoeffi})q''_i - 2a_T(\text{fluxcoeffi})q''_{\text{source}} \\
& - 2a_T(\text{fluxcoeffi})q''_{\text{boiling}} - 2a_T(\text{radcoeff})T_P^4
\end{aligned} \tag{3.67}$$

13. Front Face

$$\begin{aligned}
& \left[a_p^0 + \Theta[a_E + a_W + a_N + 2a_S(\text{coeffs}) + a_T + a_B] \right] T_P - \Theta a_E T_E - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) \left[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_T T_T^0 + a_B T_B^0 \right] \\
& + \left[a_p^0 - (1 - \Theta)[a_E + a_W + a_N + 2a_S(\text{coeffs}) + a_T + a_B] \right] T_P^0 \\
& + [2a_S(\text{coeffs})] T_\infty - 2a_S(\text{fluxcoeffs})q''_s
\end{aligned} \tag{3.68}$$

14. Back Face

$$\begin{aligned}
& \left[a_p^0 + \Theta[a_E + a_W + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P - \Theta a_E T_E - \Theta a_W T_W \\
& - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) \left[a_E T_E^0 + a_W T_W^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0 \right] \\
& + \left[a_p^0 - (1 - \Theta)[a_E + a_W + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P^0 \\
& + [2a_N(\text{coeffn})] T_\infty - 2a_N(\text{fluxcoeffn})q''_n
\end{aligned} \tag{3.69}$$

15. Left-Lateral-Face

$$\begin{aligned}
& \left[a_p^0 + \Theta[a_E + 2a_W(\text{coeffw}) + a_N + a_S + a_T + a_B] \right] T_P - \Theta a_E T_E - \Theta a_N T_N \\
& - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) \left[a_E T_E^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0 \right] \\
& + \left[a_p^0 - (1 - \Theta)[a_E + 2a_W(\text{coeffw}) + a_N + a_S + a_T + a_B] \right] T_P^0 \\
& + [2a_W(\text{coeffw})] T_\infty - 2a_W(\text{fluxcoeffw})q''_w
\end{aligned} \tag{3.70}$$

16. Right-Lateral-Face

$$\begin{aligned}
& [a_p^0 + \Theta[2a_E(\text{coeffe}) + a_W + a_N + a_S + a_T + a_B]]T_P - \Theta a_W T_W - \Theta a_N T_N \\
& - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta)[a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0] \\
& + [a_p^0 - (1 - \Theta)[2a_E(\text{coeffe}) + a_W + a_N + a_S + a_T + a_B]]T_P^0 \\
& + [2a_E(\text{coeffe})]T_\infty - 2a_E(\text{fluxcoeffe})q''_e
\end{aligned} \tag{3.71}$$

17. Front-Top Edge

$$\begin{aligned}
& [a_p^0 + \Theta[a_E + a_W + a_N + 2a_S(\text{coeffs}) + 2a_T(\text{coefft}) + a_B]]T_P - \Theta a_E T_E \\
& - \Theta a_W T_W - \Theta a_N T_N - \Theta a_B T_B = (1 - \Theta)[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_B T_B^0] \\
& + [a_p^0 - (1 - \Theta)[a_E + a_W + a_N + 2a_S(\text{coeffs}) + 2a_T(\text{coefft}) + a_B]]T_P^0 \\
& + [2a_S(\text{coeffs}) + 2a_T(\text{coefft})]T_\infty - 2a_S(\text{fluxcoeffs})q''_s - 2a_T(\text{fluxcoefft})q''_t \\
& - 2a_T(\text{fluxcoefft})q''_{\text{source}} - 2a_T(\text{fluxcoefft})q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff})T_P^4
\end{aligned} \tag{3.72}$$

18. Front-Bottom Edge

$$\begin{aligned}
& [a_p^0 + \Theta[a_E + a_W + a_N + 2a_S(\text{coeffs}) + a_T + 2a_B(\text{coeffb})]]T_P - \Theta a_E T_E \\
& - \Theta a_W T_W - \Theta a_N T_N - \Theta a_T T_T = (1 - \Theta)[a_E T_E^0 + a_W T_W^0 + a_N T_N^0 + a_T T_T^0] \\
& + [a_p^0 - (1 - \Theta)[a_E + a_W + a_N + 2a_S(\text{coeffs}) + a_T + 2a_B(\text{coeffb})]]T_P^0 \\
& + [2a_S(\text{coeffs}) + 2a_B(\text{coeffb})]T_\infty \\
& - 2a_S(\text{fluxcoeffs})q''_s - 2a_B(\text{fluxcoeffb})q''_b
\end{aligned} \tag{3.73}$$

19. Back-Top Edge

$$\begin{aligned}
& [a_p^0 + \Theta[a_E + a_W + 2a_N(\text{coeffn}) + a_S + 2a_T(\text{coefft}) + a_B]]T_P - \Theta a_E T_E \\
& - \Theta a_W T_W - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta)[a_E T_E^0 + a_W T_W^0 + a_S T_S^0 + a_B T_B^0] \\
& + [a_p^0 - (1 - \Theta)[a_E + a_W + 2a_N(\text{coeffn}) + a_S + 2a_T(\text{coefft}) + a_B]]T_P^0 \\
& + [2a_N(\text{coeffn}) + 2a_T(\text{coefft})]T_\infty - 2a_N(\text{fluxcoeffn})q''_n - 2a_T(\text{fluxcoefft})q''_t \\
& - 2a_T(\text{fluxcoefft})q''_{\text{source}} - 2a_T(\text{fluxcoefft})q''_{\text{boiling}} - 2a_T(\text{radcoeff})T_P^4
\end{aligned} \tag{3.74}$$

20. Back-Bottom Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + a_W + 2a_N(\text{coeffn}) + a_S + a_T + 2a_B(\text{coeffb})] \right] T_P - \Theta a_E T_E \\
& - \Theta a_W T_W - \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta) [a_E T_E^0 + a_W T_W^0 + a_S T_S^0 + a_T T_T^0] \\
& + \left[a_p^0 - (1 - \Theta) [a_E + a_W + 2a_N(\text{coeffn}) + a_S + a_T + 2a_B(\text{coeffb})] \right] T_P^0 \\
& + [2a_N(\text{coeffn}) + 2a_B(\text{coeffb})] T_\infty \\
& - 2a_N(\text{fluxcoeffn})q''_n - 2a_B(\text{fluxcoeffb})q''_b
\end{aligned} \tag{3.75}$$

21. Front-Right Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeffe}) + a_W + a_N + 2a_S(\text{coeffs}) + a_T + a_B] \right] T_P - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) [a_W T_W^0 + a_N T_N^0 + a_T T_T^0 + a_B T_B^0] \\
& + \left[a_p^0 - (1 - \Theta) [2a_E(\text{coeffe}) + a_W + a_N + 2a_S(\text{coeffs}) + a_T + a_B] \right] T_P^0 \\
& + [2a_E(\text{coeffe}) + 2a_S(\text{coeffs})] T_\infty \\
& - 2a_E(\text{fluxcoeffe})q''_e - 2a_S(\text{fluxcoeffs})q''_s
\end{aligned} \tag{3.76}$$

22. Back-Left Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeffw}) + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P - \Theta a_E T_E \\
& - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) [a_E T_E^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0] \\
& + \left[a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeffw}) + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P^0 \\
& + [2a_W(\text{coeffw}) + 2a_N(\text{coeffn})] T_\infty \\
& - 2a_W(\text{fluxcoeffw})q''_w - 2a_N(\text{fluxcoeffn})q''_n
\end{aligned} \tag{3.77}$$

23. Back-Right Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeffe}) + a_W + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P - \Theta a_W T_W \\
& - \Theta a_S T_S - \Theta a_T T_T - \Theta a_B T_B = (1 - \Theta) [a_W T_W^0 + a_S T_S^0 + a_T T_T^0 + a_B T_B^0] \\
& + \left[a_p^0 - (1 - \Theta) [2a_E(\text{coeffe}) + a_W + 2a_N(\text{coeffn}) + a_S + a_T + a_B] \right] T_P^0 \\
& + [2a_E(\text{coeffe}) + 2a_N(\text{coeffn})] T_\infty \\
& - 2a_E(\text{fluxcoeffe})q''_e - 2a_N(\text{fluxcoeffn})q''_n
\end{aligned} \tag{3.78}$$

24. Left-Lateral-Top Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + a_N + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P - \Theta a_E T_E \\
& - \Theta a_N T_N - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta) [a_E T_E^0 + a_N T_N^0 + a_S T_S^0 + a_B T_B^0] \\
& + \left[a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) + a_N + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P^0 \\
& + [2a_W(\text{coeff}w) + 2a_T(\text{coeff}t)] T_\infty - 2a_W(\text{fluxcoeff}w) q''_w - 2a_T(\text{fluxcoeff}t) q''_t \\
& - 2a_T(\text{fluxcoeff}t) q''_{\text{source}} - 2a_T(\text{fluxcoeff}t) q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff}) T_P^4
\end{aligned} \tag{3.79}$$

25. Left-Lateral-Bottom Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [a_E + 2a_W(\text{coeff}w) + a_N + a_S + a_T + 2a_B(\text{coeff}b)] \right] T_P - \Theta a_E T_E \\
& - \Theta a_N T_N - \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta) [a_E T_E^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0] \\
& + \left[a_p^0 - (1 - \Theta) [a_E + 2a_W(\text{coeff}w) + a_N + a_S + a_T + 2a_B(\text{coeff}b)] \right] T_P^0 \\
& + [2a_W(\text{coeff}w) + 2a_B(\text{coeff}b)] T_\infty \\
& - 2a_W(\text{fluxcoeff}w) q''_w - 2a_B(\text{fluxcoeff}b) q''_b
\end{aligned} \tag{3.80}$$

26. Right-Lateral-Top Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta [2a_E(\text{coeff}e) + a_W + a_N + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_S T_S - \Theta a_B T_B = (1 - \Theta) [a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_B T_B^0] \\
& + \left[a_p^0 - (1 - \Theta) [2a_E(\text{coeff}e) + a_W + a_N + a_S + 2a_T(\text{coeff}t) + a_B] \right] T_P^0 \\
& + [2a_E(\text{coeff}e) + 2a_T(\text{coeff}t)] T_\infty - 2a_E(\text{fluxcoeff}e) q''_e - 2a_T(\text{fluxcoeff}t) q''_t \\
& - 2a_T(\text{fluxcoeff}t) q''_{\text{source}} - 2a_T(\text{fluxcoeff}t) q''_{\text{boiling}} \\
& - 2a_T(\text{radcoeff}) T_P^4
\end{aligned} \tag{3.81}$$

27. Right-Lateral-Bottom Edge

$$\begin{aligned}
& \left[a_p^0 + \Theta \left[2a_E(\text{coeffe}) + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb}) \right] \right] - \Theta a_W T_W \\
& - \Theta a_N T_N - \Theta a_S T_S - \Theta a_T T_T = (1 - \Theta) \left[a_W T_W^0 + a_N T_N^0 + a_S T_S^0 + a_T T_T^0 \right] \\
& + \left[a_p^0 - (1 - \Theta) \left[2a_E(\text{coeffe}) + a_W + a_N + a_S + a_T + 2a_B(\text{coeffb}) \right] \right] T_p^0 \\
& + \left[2a_E(\text{coeffe}) + 2a_B(\text{coeffb}) \right] T_\infty \\
& - 2a_E(\text{fluxcoeffe})q''_e - 2a_B(\text{fluxcoeffb})q''_b
\end{aligned} \tag{3.82}$$

IV. RESULTS AND DISCUSSION

Results are presented for the different cases discussed in greater detail below. As noted earlier, a variable sized and moving numerical mesh was used in such a way that the arc was always positioned at the center of the mesh where the spacing is the finest. The goal is to be able to resolve the large temperature gradient features around the arc and yet not incur a large overhead of computer resources, which would be required, if the grid was uniformly fine all over. This strategy however did require that a separate mesh generating routine be used which did demand some extra computational resources. The weld pool region was also modeled as a solid region but with a thermal conductivity higher than the surrounding unmelted region to simulate the effects of weld pool convection. The discontinuity in the thermal conductivity boundaries was handled using the standard technique of employing harmonic averaging at the boundary. Since the coefficients of the system of equations depend on the temperature, an iterative solution technique was used to achieve convergence in such a way that the maximum temperature difference between two consecutive iterations at any grid point was no more than 0.1°C .

The numerical solution method was used to examine different cases in freshwater for a 40-mm-thick 70 x 90 mm workpiece with a moving heat source in the positive y-direction.

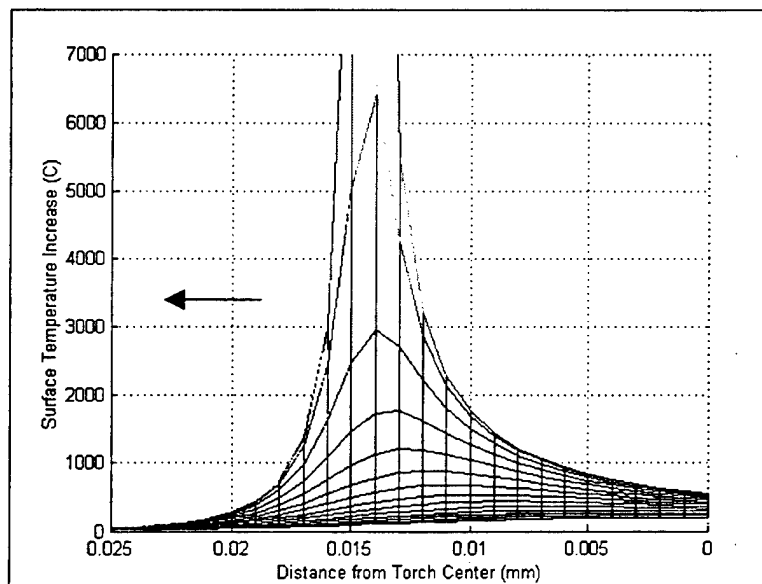
Case 1a:

The validity of the numerical model was compared to Rosenthal's three-dimensional solution for a moving heat source. At this point, convective, radiative and boiling surface thermal conditions were not considered. A constant thermal conductivity

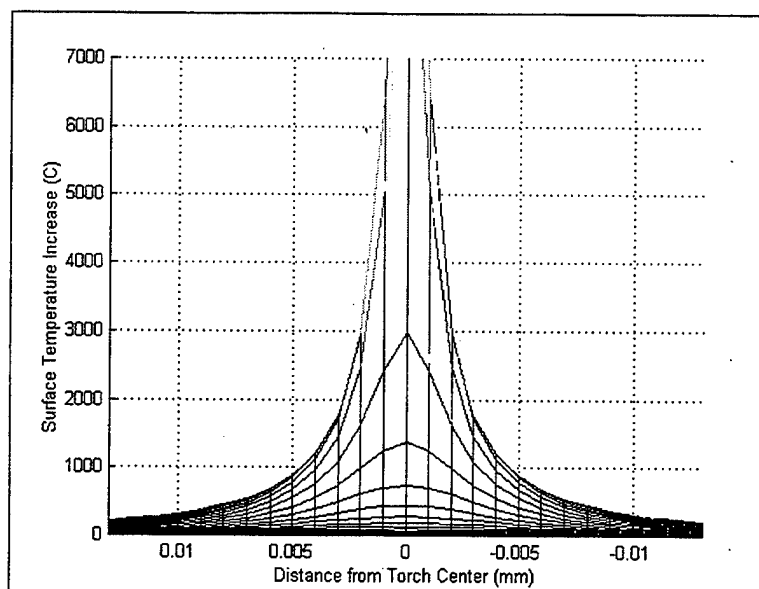
value and a point heat source were used in the calculations. The input data used in computations for case 1a are shown in Table 1. The calculations were made up to 3.6 seconds. The numerical results show excellent agreement with the analytical results of Rosenthal (Figure 4.1 and Figure 4.2).

Workpiece :	$Length = 70\text{ mm},$	$Width = 90\text{ mm},$	$Thickness = 40\text{ mm}$
Water temperature:	$T = 27\text{ }^{\circ}\text{C} (300.15\text{ K})$		
Power input into workpiece :	$Q = 2544\text{ W}$		
Arc torch speed :	$v_y = 4\text{ mm/s}$		
Radius of heat input distribution :	$r_o = 4.5\text{ mm}$		
Thermal conductivity :	$k = 53\text{ W/m K}$		
Density of steel :	$\rho = 7854\text{ kg/m}^3$		
Specific heat of steel :	$C_p = 509.3\text{ J/kgK}$		

Table 1. The input data used in computations for case 1a

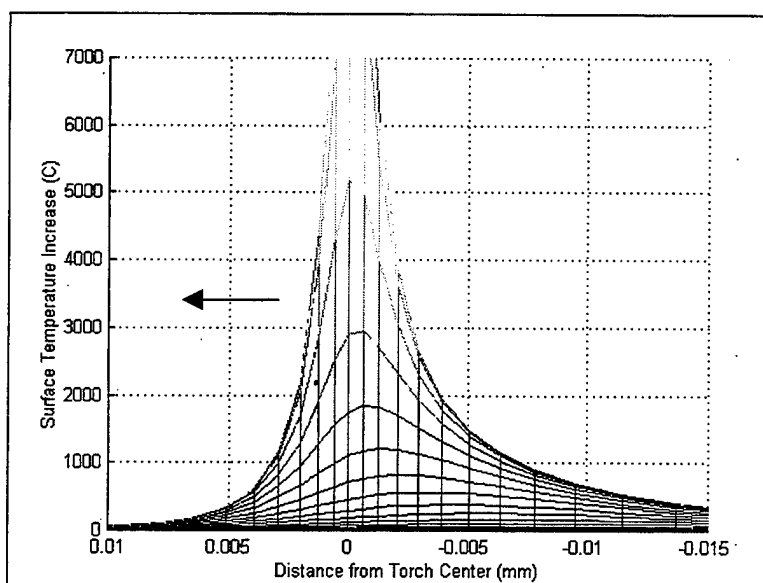


(a)

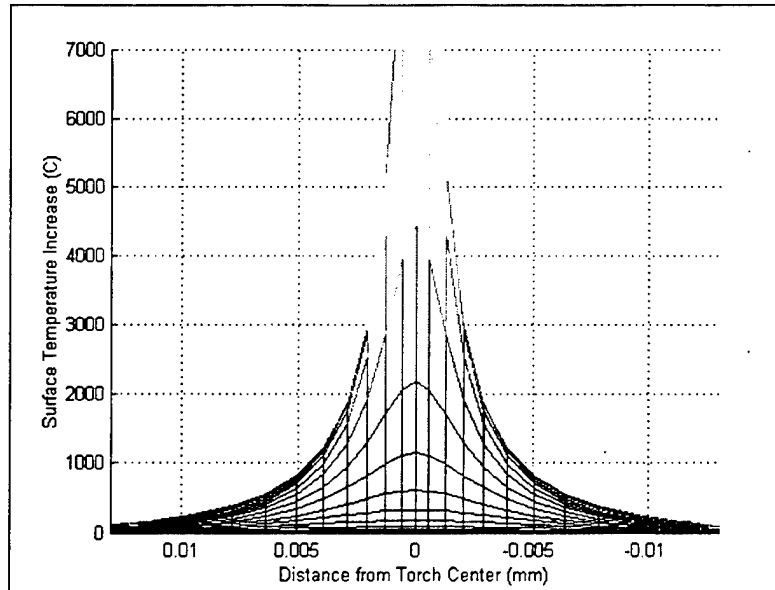


(b)

Figure 4.1 Rosenthal's point heat source solution:
(a) Profile; (b) Front-view.



(a)



(b)

**Figure 4.2 The numerical model point heat source solution:
(a) Profile; (b) Front-view.**

Case 1b :

In this case, the surface thermal boundary conditions of boiling heat transfer were now included. Other relevant data is given in Table 2.

Workpiece :	Length = 70 mm,	Width = 90mm,	Thickness = 40mm
Water temperature:	$T = 27\text{ }^{\circ}\text{C}$ (300.15 K)		
Saturation temperature:	$T = 100\text{ }^{\circ}\text{C}$ (373.15 K)		
Water depth :	$l = 0\text{ ft}$		
Total pressure:	$P = 101.325\text{ kPa}$		
Power input into workpiece :	$Q = 2544\text{ W}$		
Arc torch speed :	$v_y = 4\text{ mm/s}$		
Radius of heat input distribution :	$r_o = 4.5\text{ mm}$		
Thermal conductivity ; k (W/m K)	$53 - 0.04(T-300)$ $300 \leq T(K) \leq 1000$ $25 + 6.25 \times 10^{-3}(T-1000)$ $1000 \leq T(K) \leq 1800$ 125 $T(K) > 1800$		
Emissivity :	$\varepsilon = 0.82$		

Table 2. The input data used in computations for case 1b

The top surface temperature values at 0.5 seconds can be seen in Figure 4.3 and Figure 4.4. At 0.5 seconds, the temperature distribution is almost symmetric about the position of the arc. The temperature distribution around the arc center is above the melting temperature of the workpiece ($T_m = 1800$ K). To see the depth of the weld pool penetration, the melting temperature contours were plotted for different surfaces (Figure 4.5) and the weld pool depth was shown through the melting temperature points by using a curve-fit (Figure 4.6). Because of the brief reaction time, the arc heat input penetration to the workpiece is limited and a well-formed weld pool cannot be seen. [Ref. 16]

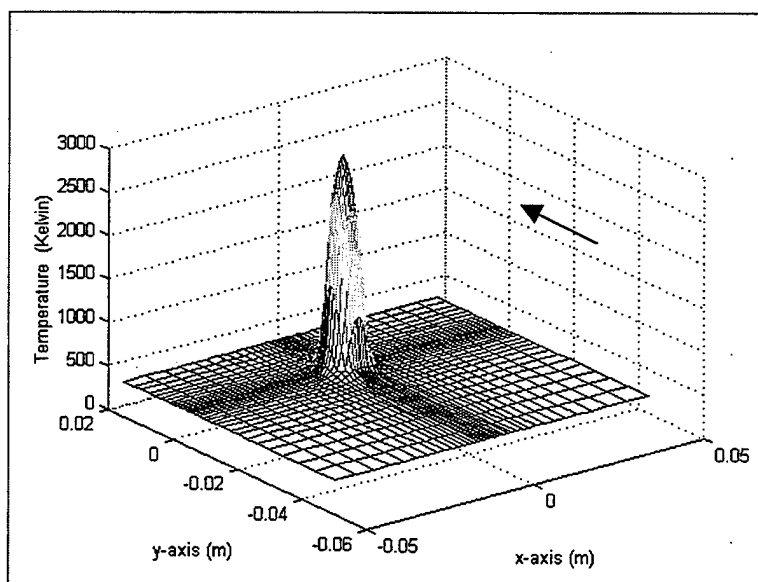


Figure 4.3 Top surface $t=0.5$ sec

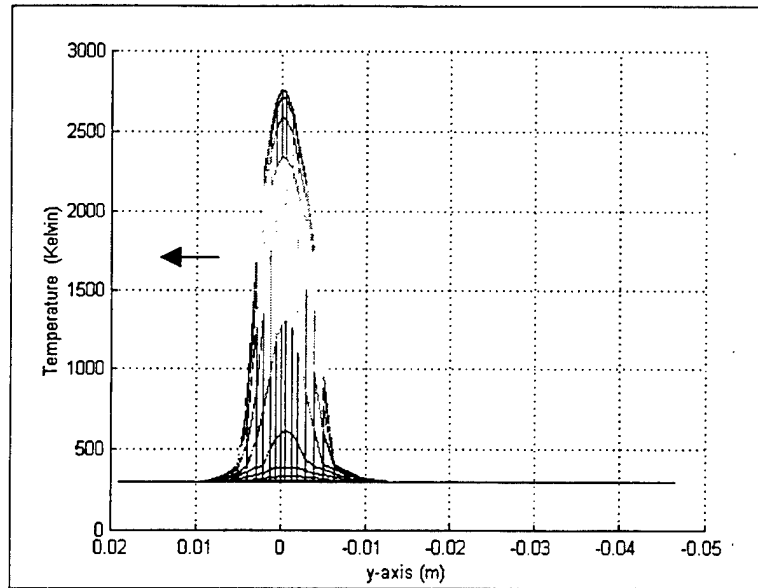


Figure 4.4 Top surface (profile) $t=0.5$ sec

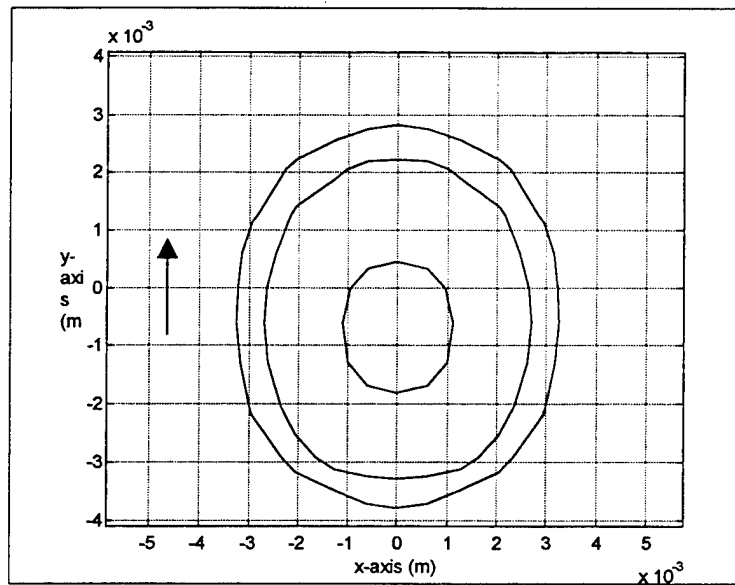


Figure 4.5 The weld pool surface characteristics by the $T=1800$ K contours $t=0.5$ sec

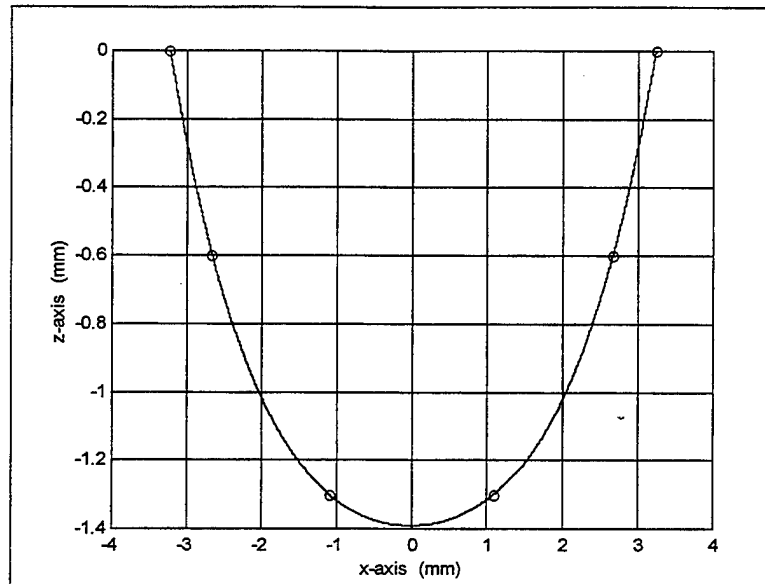


Figure 4.6 The weld pool (front-view) $t=0.5$ sec

At $t = 2.0$ sec., the results reveal that the temperature distribution has already reached steady state. Due to the moving heat source, the small change of slope of the temperature distribution curve can be observed clearly (Figure 4.7 and Figure 4.8). Because of the increased reaction time, the melting temperature contours are seen till the fourth surface from the top (Figure 4.9). The weld pool width increased to 8 mm and its depth to 2.1 mm (Figure 4.10).

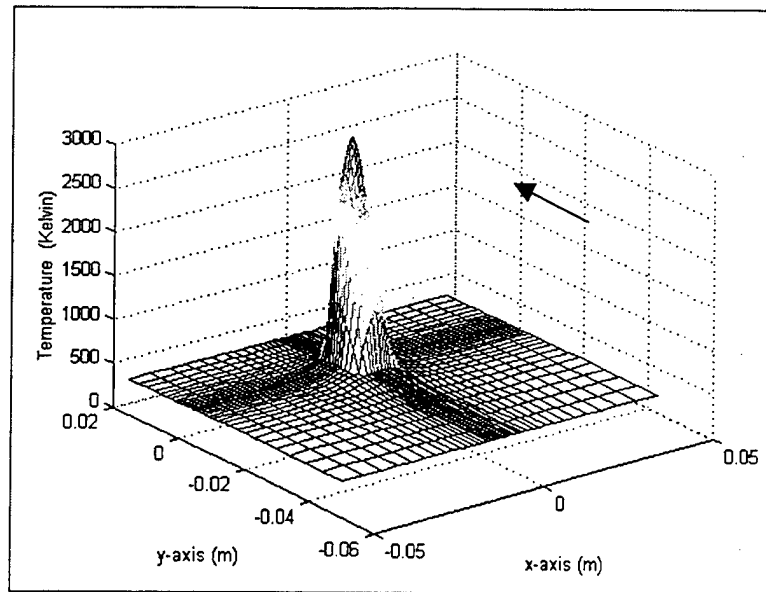


Figure 4.7 Top surface $t=2.0$ sec

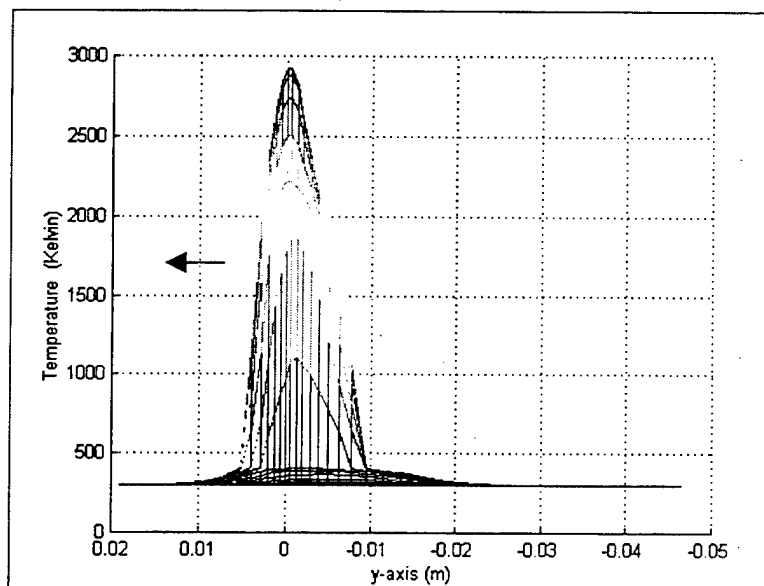


Figure 4.8 Top surface (profile) $t=2.0$ sec

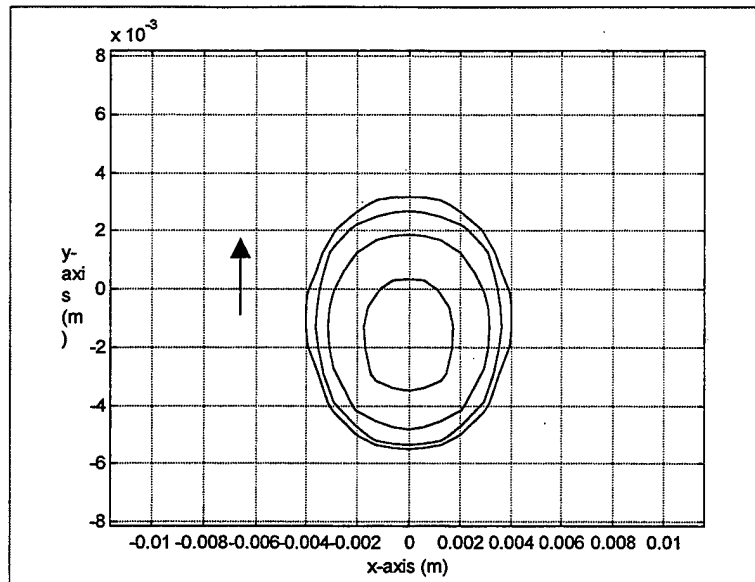


Figure 4.9 The weld pool surface characteristics by the $T=1800$ K contours $t=2.0$ sec

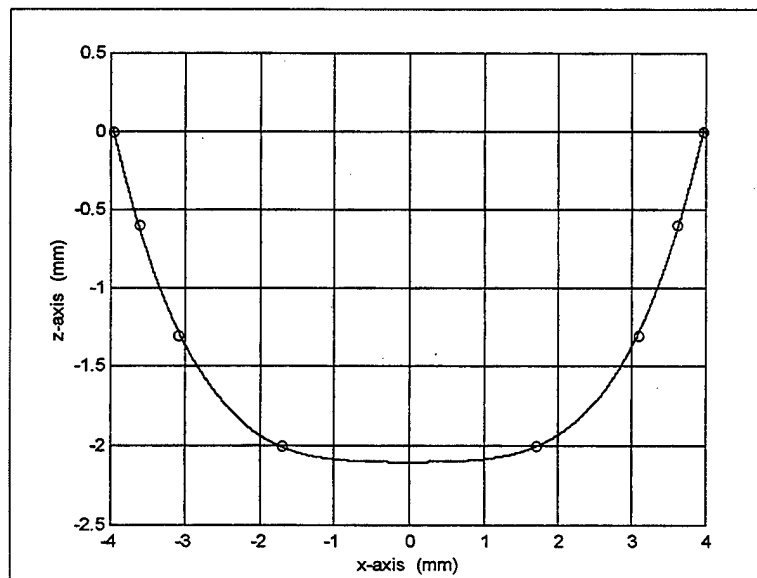


Figure 4.10 The weld pool (front-view) $t=2.0$ sec

At $t = 4.5$ sec., the change of slope of the temperature distribution behind the arc is more evident in Figure 4.11 and Figure 4.12. The melting temperature contours are still seen till the fourth surface from the top (Figure 4.13). But, the weld pool depth reached to 3 mm with a width of 8 mm (Figure 4.14). These results show a good agreement with the data from Ref.16.

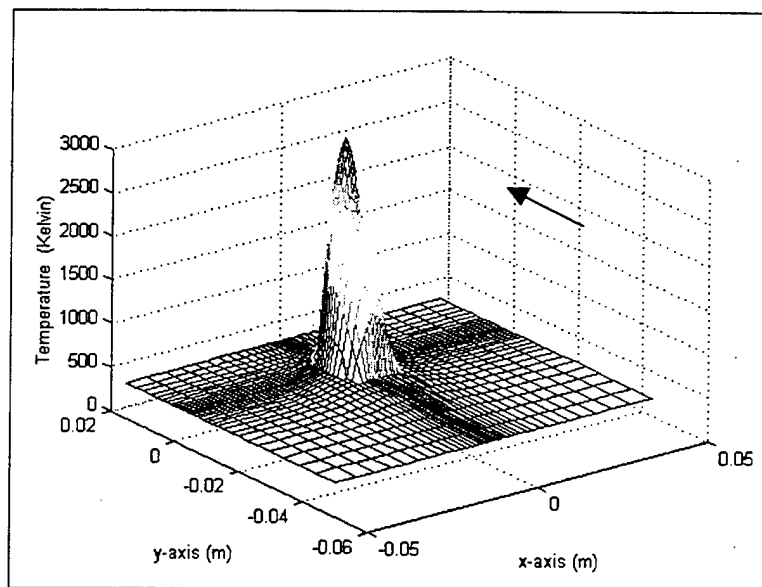


Figure 4.11 Top surface $t=4.5$ sec

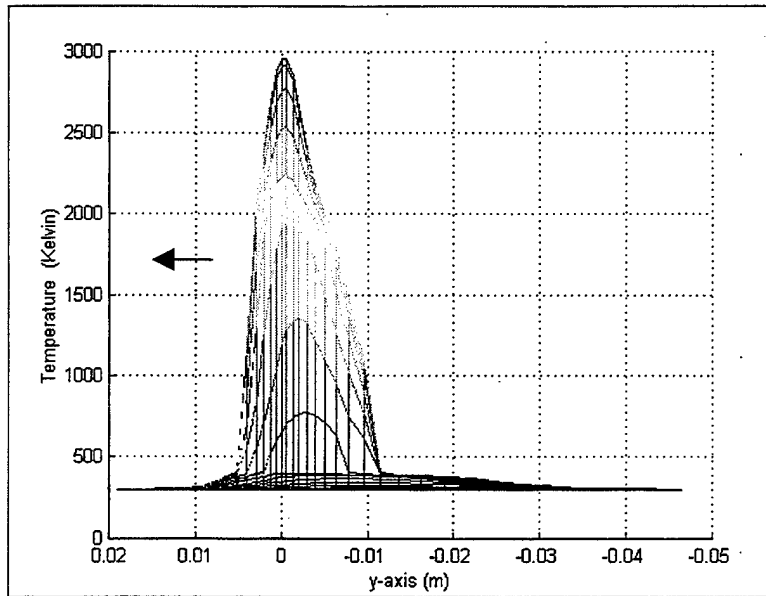


Figure 4.12 Top surface (profile) $t=4.5$ sec

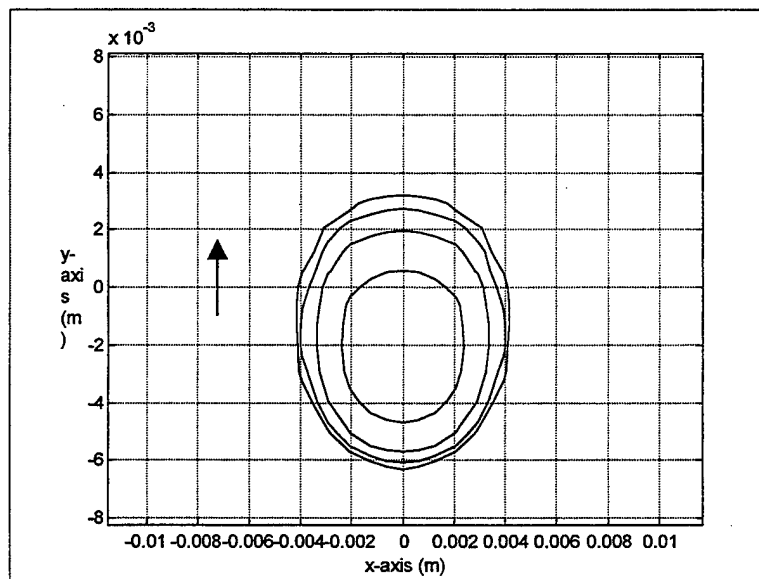


Figure 4.13 The weld pool surface characteristics by the $T=1800$ K contours $t=4.5$ sec

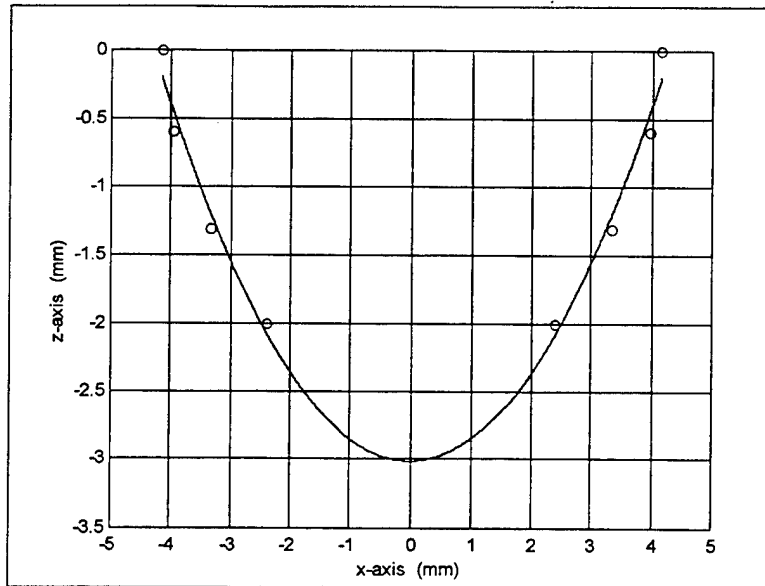


Figure 4.14 The weld pool (front-view) $t=4.5$ sec

Case 2, 3, 4 :

In the following cases we do not consider the temperature distribution. Instead, the cooling time that elapses between 800°C and 500°C is examined for freshwater at different temperatures and welding depths. The input data used in calculations are shown in Table 3, Table 4 and Table 5.

Workpiece :	Length = 70 mm,	Width = 90mm,	Thickness = 40mm
Water temperature:	$T = 3\text{ }^{\circ}\text{C}$ (276.15 K)		
Saturation temperature :	$T = 114.88\text{ }^{\circ}\text{C}$ (388.03 K)		
Water depth :	$l = 22\text{ ft}$ (6.706 m)		
Total Pressure :	$P = 168.75\text{ kPa}$		
Power input into workpiece :	$Q = 3900\text{ W}$		
Arc torch speed :	$v_y = 2.75\text{ mm/s}$		
Radius of heat input distribution :	$r_o = 4.5\text{ mm}$		
Thermal conductivity ; k (W/m K)	$53 - 0.04 (T-300)$ $300 \leq T(K) \leq 1000$ $25 + 6.25 \times 10^{-3} (T-1000)$ $1000 \leq T(K) \leq 1800$ 125 $T(K) > 1800$		
Emissivity :	$\varepsilon = 0.82$		

Table 3. The input data of calculations at $T = 3\text{ }^{\circ}\text{C}$ and $l = 22\text{ ft}$

Workpiece :	Length = 70 mm,	Width = 90mm,	Thickness = 40mm
Water temperature:	$T = 10\text{ }^{\circ}\text{C}$ (283.15 K)		
Saturation temperature :	$T = 112.62\text{ }^{\circ}\text{C}$ (385.77 K)		
Water depth :	$l = 18\text{ ft}$ (5.486 m)		
Total Pressure :	$P = 156.492\text{ kPa}$		
Power input into workpiece :	$Q = 3900\text{ W}$		
Arc torch speed :	$v_y = 2.75\text{ mm/s}$		
Radius of heat input distribution :	$r_o = 4.5\text{ mm}$		
Thermal conductivity ; k (W/m K)	$53 - 0.04 (T-300)$ $300 \leq T(K) \leq 1000$ $25 + 6.25 \times 10^{-3} (T-1000)$ $1000 \leq T(K) \leq 1800$ 125 $T(K) > 1800$		
Emissivity :	$\varepsilon = 0.82$		

Table 4. The input data of calculations at $T = 10\text{ }^{\circ}\text{C}$ and $l = 18\text{ ft}$.

Workpiece :	$Length = 70\text{ mm},$	$Width = 90\text{mm},$	$Thickness = 40\text{mm}$
Water temperature:	$T = 31\text{ }^{\circ}\text{C} \text{ (304.15 K)}$		
Saturation temperature :	$T = 115.44\text{ }^{\circ}\text{C} \text{ (388.59 K)}$		
Water depth :	$l = 24\text{ ft} \text{ (7.315 m)}$		
Total Pressure :	$P = 171.762\text{ kPa}$		
Power input into workpiece :	$Q = 3900\text{ W}$		
Arc torch speed :	$v_y = 2.75\text{ mm/s}$		
Radius of heat input distribution :	$r_o = 4.5\text{ mm}$		
Thermal conductivity ; k (W/m K)			
	$53 - 0.04 (T-300)$	$300 \leq T(K) \leq 1000$	
	$25 + 6.25 \times 10^{-3} (T-1000)$	$1000 \leq T(K) \leq 1800$	
	125	$T(K) > 1800$	
Emissivity :	$\varepsilon = 0.82$		

Table 5. The input data of calculations at $T = 31\text{ }^{\circ}\text{C}$ and $l = 24\text{ ft}$.

The resulting graphs of the water temperature and welding depth effects on the peak temperature and the cooling time (between $800\text{ }^{\circ}\text{C}$ and $500\text{ }^{\circ}\text{C}$) for a point initially 1.25 mm . behind the arc heat source are shown in Figure 4.15, Figure 4.16 and Figure 4.17. The cooling times for $800 - 500\text{ }^{\circ}\text{C}$ temperature range are 0.19 seconds ($T_{\text{water}}=3\text{ }^{\circ}\text{C}, l=22\text{ ft.}$), 0.20 seconds ($T_{\text{water}}=10\text{ }^{\circ}\text{C}, l=18\text{ ft.}$) and 0.33 seconds ($T_{\text{water}}=31\text{ }^{\circ}\text{C}, l=24\text{ ft.}$) respectively. These cooling times are significantly different from those in the previous studies in the literature. Based on the models in this case, it is seen that boiling phenomena must be considered for surface heat losses from the weld metal. For $800\text{ }^{\circ}\text{C}$ to $500\text{ }^{\circ}\text{C}$ range of the weld metal, film boiling exists and the surface is completely covered by a vapor blanket. A high amount of heat transfer from the weld metal to the water environment occurs by conduction across the vapor film. The other reason is that due to the thermal conductivity equations used in the model, for $800\text{ }^{\circ}\text{C}$ to $500\text{ }^{\circ}\text{C}$ ranges, thermal conductivity of steel is low. This causes a decrease in conduction heat

transfer from the arc to the studied point. But the high convection heat loss at this point causes a rapid cooling rate. Another possible reason is that because of the low welding speed and the high thermal conductivity in the weld pool, the workpiece, which has a large thermal capacity, absorbs most of the heat energy. After the arc has passed, temperature and thermal conductivity at that point drops. Due to the low thermal conductivity, the transfer of the absorbed heat energy from the inner part of the workpiece to the surface is not sufficient compared to the surface heat loss. This causes a rapid cooling rate on the surface.

During welding, the high heat input to the workpiece increases the degree of grain growth. The grain coarsening effect causes a decrease in the grain boundary area. Because of the large grain size and the very high cooling rates, the resulting phase is hard with $V_i > 450$, but with a brittle martensitic structure.

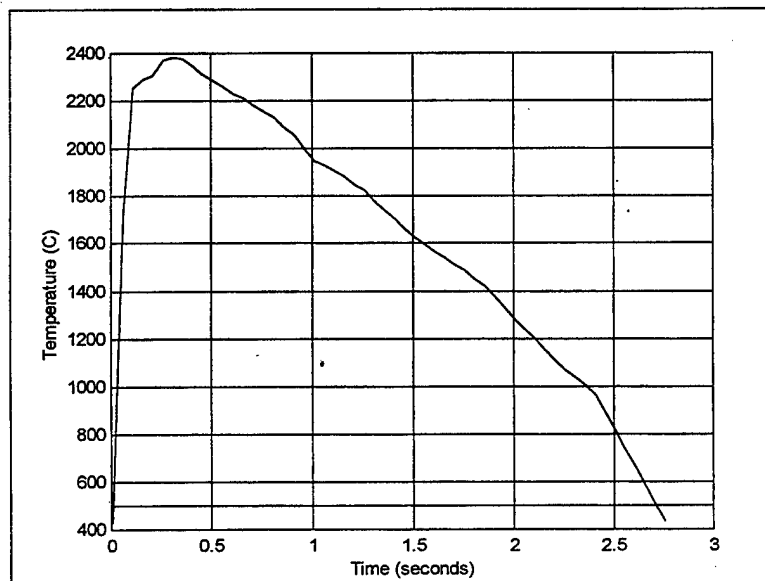


Figure 4.15 Cooling time and peak temperature for a point initially 1.25 mm behind the arc heat source. ($T_{\text{water}} = 3\text{ C}$, depth = 22 ft.)

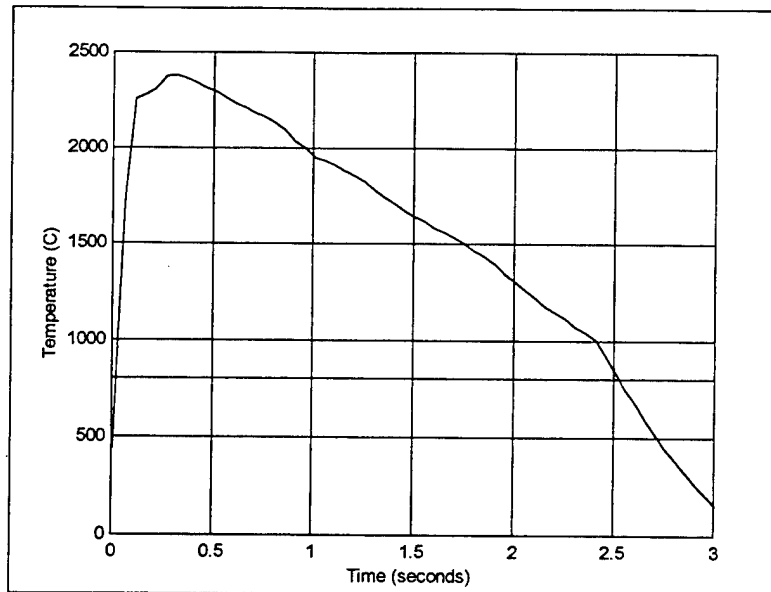


Figure 4.16 Cooling time and peak temperature for a point initially 1.25 mm behind the arc heat source.
($T_{\text{water}} = 10 \text{ C}$, depth = 18 ft.)

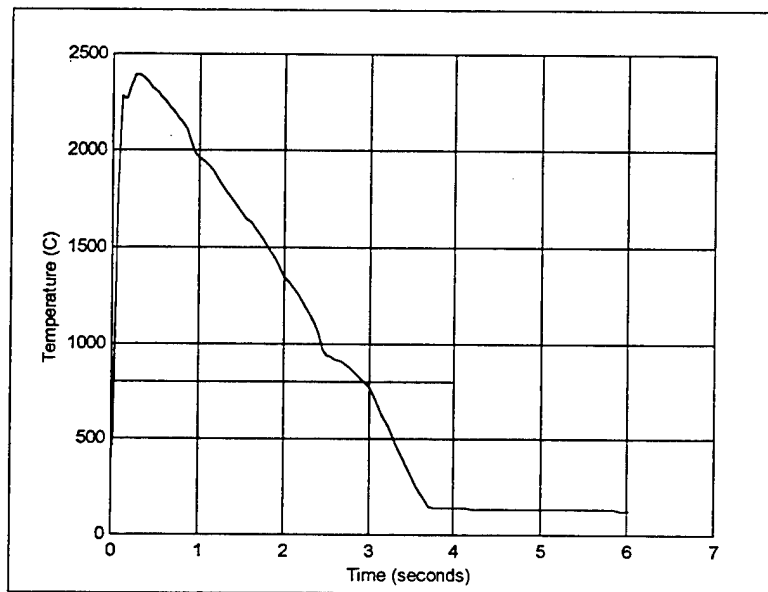


Figure 4.17 Cooling time and peak temperature for a point initially 1.25 mm behind the arc heat source.
($T_{\text{water}} = 31 \text{ C}$, depth = 24 ft.)

V. CONCLUSIONS AND RECOMMENDATIONS

Compared to experimental methods, numerical solution techniques are very fast, effective and economical alternatives. Therefore, the aim of this study was to develop a general numerical model for transient, three-dimensional conduction heat transfer phenomena in underwater welding process on a thick rectangular plate. Computations were presented for different cases. Comparisons of current predictions with results in the literature showed good agreement and validated the model. If the material, environment and the arc source properties are known, this program can be applied to the different types of metals under the wet welding process.

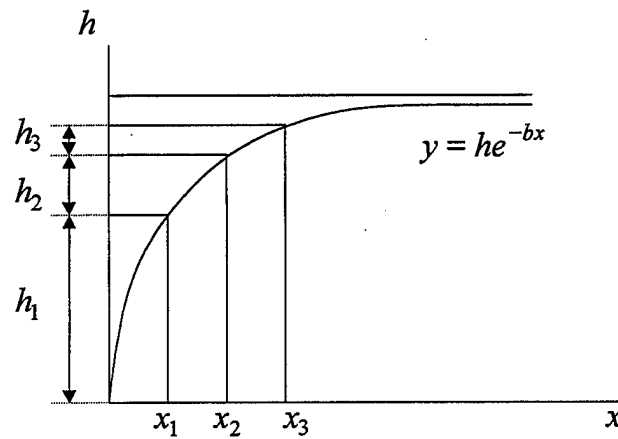
The numerical model gave important temperature-time data in the critical HAZ, which in turn determines material structure. The model also helped to make prediction of size of weld pool and its evolution with time.

The principal recommendations for the future studies may be summarized as follows.

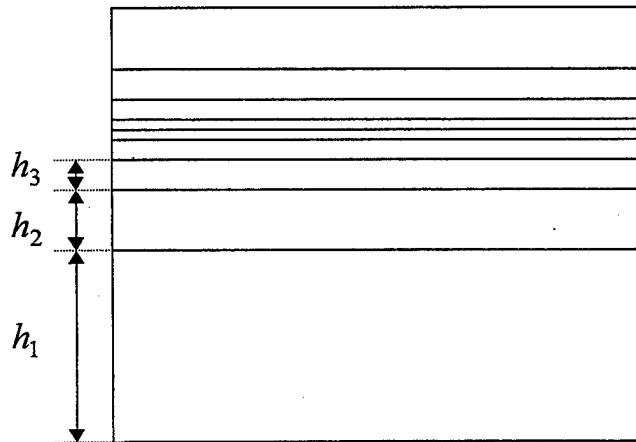
1. The weld pool region itself was modeled, as a solid region. But, owing to the presence of the temperature values higher than the melting temperature in the weld pool, the model must be considered as a liquid weld pool with the surrounding unmelted solid region. So, the numerical model needs to be modified to account for weld pool convection with the buoyancy force, the electromagnetic force and the surface tension gradient at the weld pool surface.
2. The chemical effects due to interaction of the arc with the surrounding water environment may need to be included.
3. In the model, boiling heat transfer phenomena was accounted for a solid surface. For a molten pool region, the role of boiling on a liquid substrate is unknown and needs to be solved.

APPENDIX A. PROGRAM STRUCTURE

In the main program <aath10.m>, non-uniform grid spacing was constructed by using the exponential function $y = he^{-bx}$ where b is the coefficient which affects the grid spacing distribution and h is the sum of the grid spacings when $x \rightarrow \infty$. The application of grid spacing to the workpiece was shown in Figure A1.



(a)



(b)

Figure A1 Applying the grid spacing to the workpiece by using the exponential function

The distance in front of the moving heat source was called as “front” and the distance behind the heat source was called as “back”. The width and the thickness of the plate were defined with the same terms. The variable “number” was used to define the number of grid points in the “back” region. Variables “b” and “number” can be changed to control the resolution of the grid spacing.

To simulate the heat input from the arc to the top surface, it was assumed that the arc source heat flux distribution had a Gaussian distribution on the top surface. The applied arc heat source was defined as a matrix, which its size was equal to the size of the top surface. The value of heat flux at each grid point was calculated by using the x-coordinate and the y-coordinate of that point. The resulting arc heat source matrix was applied to the top surface.

The discretised equations was represented by the matrix equation $[A] X = b$ where, $[A]$ is the coefficient matrix, X is the column matrix of the unknown temperature values of the grid points and b is the column matrix of the constants. In the coefficient matrix, the coefficients of the studied grid and the neighboring grids were written to the same row. An example for 5x5x2 workpiece was shown in Figure A2. In this example, the coefficient matrix may be written as

$$\begin{bmatrix} c1 & c2 & 0 & 0 & 0 & c6 & 0... & c26... \\ c1 & c2 & c3 & 0 & 0 & 0 & c7 & 0... & c27... \\ 0 & c2 & c3 & c4 & 0 & 0 & 0 & 0... & c28... \\ \\ \\ 0 & 0... & & & & c25... & & c45... & c49 & c50 \end{bmatrix}$$

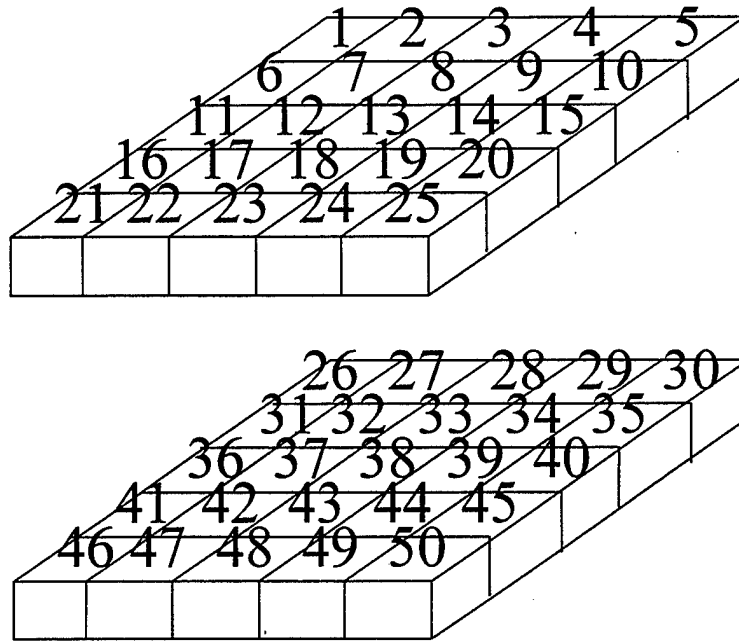


Figure A2 The example workpiece

Even the row number, the column number and the surface number were increased, the maximum number of coefficients for each row does not exceed 7. This is for an interior grid point of a workpiece with a surface number 3 or more. But, the increasing size of the coefficient matrix increases the computing time and the required computing memory and storage capacity. To prevent this problem, tridiagonal part of the general matrix (Figure A3) which contains T_W , T_P and T_E coefficients was formed as a new $a \times 3$ matrix (a = row $\times$ column $\times$ surface). The coefficient values and T_N , T_S , T_T and T_B were also formed separately and called as <north>, <south>, <top> and <bottom> matrices respectively. The size of each matrix is $a \times 1$ (a = row $\times$ column $\times$ surface). The constant values in the right side of the equation were called as <right> matrix.

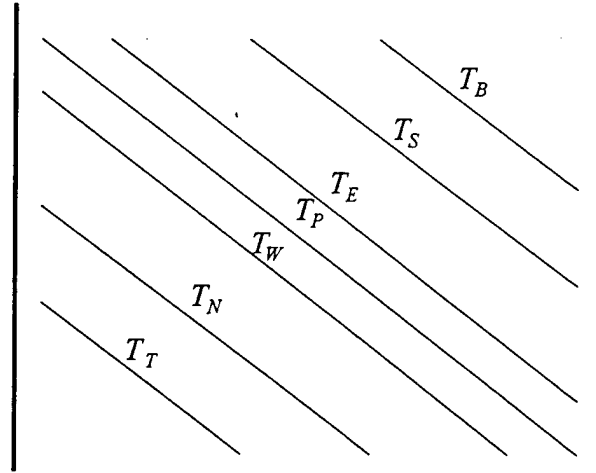


Figure A3 The coefficient matrix

Beginning from the first surface, the function program <aasolve10> solves the matrix produced by the main program by using row-by-row iterative sweeping method. The representation of row-by-row method was shown in Figure A4 [Ref. 21,22]. The statements <index1> and <index2> defines the first and the last element of each row.

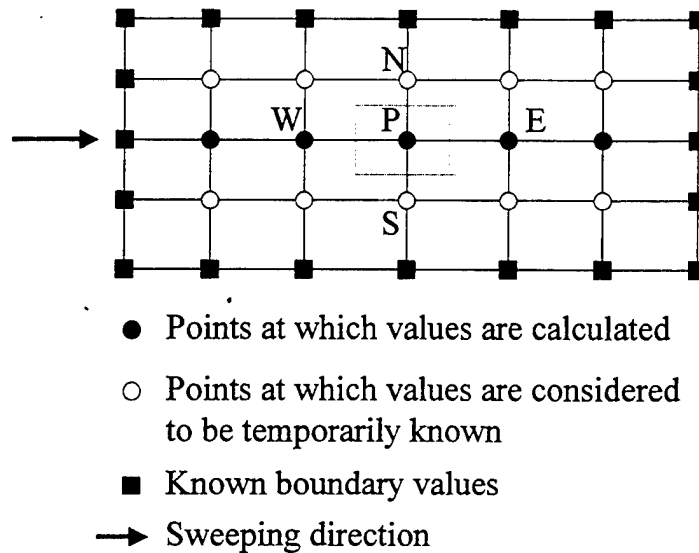


Figure A4 Row-by-row method representation

The matrix `<temp_old>` represents the old temperature values of the grid points. At first, it was initialized and then it took the values of the previous iteration. The matrix `<temp>` represents the temperature values of the grid points, which were found from the present iteration. The sizes of `<temp_old>` and `<temp>` are $(a \times b) \times c$, where a , b and c are the total row number, column number and surface number respectively. The statement `<delta_iteration>` represents the absolute value of the difference between `<temp_old>` and `<temp>`. If `<delta_iteration>` is less 0.1, the resulting temperature value is satisfactory. The statement `<count_iteration>` counts the iteration number. If the program can not converge after 10 iterations, `<delta_iteration>` increases 0.1 and continues to increase 0.1 at every following 5 iterations until the program find a converged temperature value. The radiation heat flux matrix `<radiation1>` and the boiling heat flux matrix `<q_boiling_free>` are only used to calculate the heat flux from the top surface. . The sizes of `<radiation1>` and `<q_boiling_free>` are $(a \times b) \times 1$, where a , b are the total row number and column number respectively.

In the `<aasolve10>` function program, `<north>`, `<south>`, `<top>` and `<bottom>` matrices were passed to the right side of the equation. Their values were calculated by using the temperature values from the previous iterations and the resulting values were added to the values of `<right>`. The diagonal matrix was solved by using Gauss elimination method and the values of T_W , T_P and T_E was found for the present iteration.

In the `<old_temp10>` function program, it was assumed that the plane was moving with the arc source together (Figure A5). Here, due to the non-uniform grid spacing, it was necessary to determine the new position and the temperature value of each grid point by using second degree polynomial interpolation (Figure A6). The values of T_a , T_b and

T_c were found by using the old temperatures from the previous time step. These temperatures was used to determine the new grid temperature T_x with the second degree polynomial interpolation.

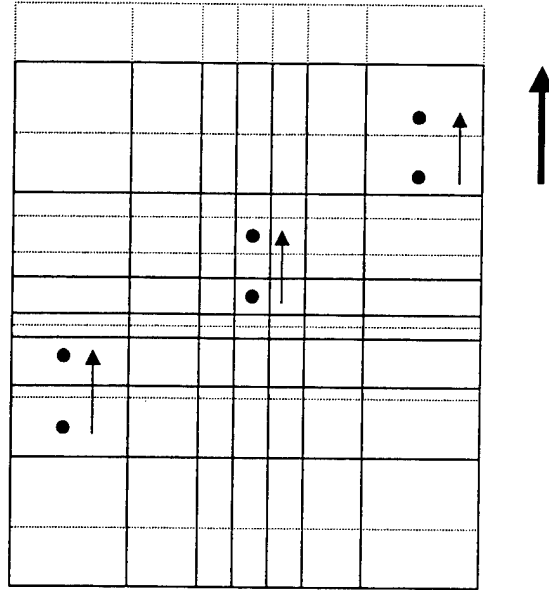


Figure A5 Moving plane

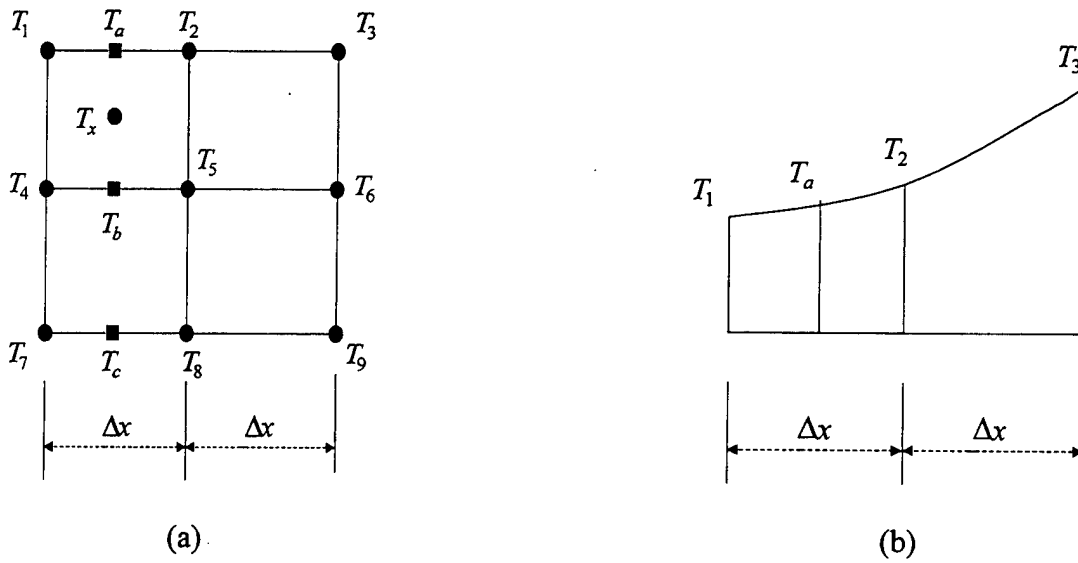


Figure A6 Grid positioning by using second degree polynomial

The function program <save_all10> saves all 3-D temperature data at each time step by overwriting onto. The other function program <save_the10> saves top surface temperatures at each time step into a different file name.

APPENDIX B. PROGRAM CODES

```
%  
%  
% THE MAIN PROGRAM (aath10.m)  
%  
%  
% Need to define the function below to be able to use mcc  
% The symbol "a1" itself has no significance - aath10 is the  
% name of the main program.  
function [a1]=aath10()  
  
clear      % clear all variables - not present in "aath10goon.m"  
close all  % close all open figure windows  
  
% save_me is a counter that is incremented after each time step  
% and is used to decide how often to save the calc temperature  
% data.  
save_me=0;  
  
% save_counter is a counter that is also incremented with each  
% time step and is appended to the file name into which the temp  
% data is stored.  
save_counter=0;    % not present in "aath10goon.m"  
  
% Tinf is the ambient temp in deg C  
Tinf=27;  
  
% T0 is the initial temp in deg C  
T0=27;  
  
% sil is the stefan-boltzmann const (SI units)  
sil=5.67E-8;  
  
% epsilon is the emissivity of the surface  
epsilon=0;  
  
% x-vel component of torch speed (m/s)  
vel_x=0;  
  
% y-vel component of torch speed (m/s)  
vel_y=0.004;  
  
% deltat_t is the time step (in secs)  
delta_t=0.001;  
  
% Distance source moves in the x-direction in each time step(in m)  
deltax=vel_x*delta_t;  
  
% Distance source moves in the y-direction in each time step(in m)  
deltay=vel_y*delta_t;  
  
% q_up is the heat flux distribution on the upper surface (W/m^2)
```

```

% q_up is the heat flux distribution on the upper surface (W/m^2)
% imposed by the torch/arc (implemented in matrix form)
q_up=0;

%% q is a generic variable representing heat flux that may be
%% present at any of the other surfaces and can be included
%% in the nodal equations

% q_ is constant heat flux through the surfaces
q_west=0;
q_east=0;
q_top=0;
q_bottom=0;
q_north=0;
q_south=0;

% c is the cp [J/kg-K]
c=509.3;

% ro is the density [kg/m^3]
ro=7854;

% Convection heat transfer coefficients from the faces [W/m^2-K]
he=0;
hw=0;
hn=0;
hs=0;
ht=0;
hb=0;

% The distances between the grid point points

%% back is the portion of the moving grid behind the point
%% of location of the source [in m]
back=.05;

% b is the coefft in the exponential gridding function (e^(-b*x))
b=.13;

% number is the no. of grid points in the "back" region
number=20;

%% Variables "b" and "number" above can be varied to control the
%% features of the variable grid, such as change in grid spacing,
%% etc.
%% height is a vector that holds the coords of the grid points in %%
the back-region. Note that the spacing between these grid
%% points is the diff. between consecutive height entries and is
%% stored in "int_back"
height(1)=0;

```

```

% Note that int_back(1) > int_back(number), i.e. in reverse order
for kk=1:number
    height(kk+1)=back*(1-exp(-b*kk));
    int_back(kk)=height(kk+1)-height(kk);
end

%% multiply is a scaling factor(close to 1) that scales height and
%% int_back in such a way that it height(number) = back.
multiply=back/(sum(int_back)-int_back(1)/2)
int_back=multiply*int_back;

%% Similar to back, below are the variables front (front region of %%
%% moving grid ahead of source), width (width of plate) and
%% thickness. All distances in m.
front=0.02;
width=0.09;
thickness=0.04;

%% int_front is the grid spacing in the front region. Initially
%% set to 0. Then transfer from int-back one by one until sum of
%% int-front entries exceeds front.
int_front=0;
counter=0;

while (sum(int_front)-int_front(length(int_front))/2)<=front

% int_front below continually changes size as the spacings are
% added.
int_front=[int_front,int_back(length(int_back)-counter)];
counter=counter+1;

end

% Set first member which was initially 0 to null.
% Meaningful spacing starts only from the 2nd member of int_front
int_front(1)=[];

%% The grid distance matrix in the y direction is y_intl starting
%% from the top of front down through the source all the way to
%% the bottom of back.
% fliplr is to ge the desired ordering from front to back.
y_intl=[fliplr(int_front),fliplr(int_back)];

% int_side holds grid spacing values over the width.
% Same logic as for int_front
int_side=0;
counter=0;

while (sum(int_side)-int_side(length(int_side))/2)<=(width/2)

```

```

    int_side=[int_side,int_back(length(int_back)-counter)];
    counter=counter+1;
end

int_side(1)=[];

% The grid distance vector in the x dirextion is x_intl (y_intl
% above.
x_intl=[fliplr(int_side),int_side];

% The grid distance vector in the z dirextion is z_intl (y_intl
% above)
z_intl=0;

counter=0;

while (sum(z_intl)-z_intl(length(z_intl))/2)<=thickness

    z_intl=[z_intl,int_back(length(int_back)-counter)];
    counter=counter+1;

end

z_intl(1)=[];

figure(1)

subplot(3,1,1)
plot(x_intl)
grid on

subplot(3,1,2)
plot(y_intl)
grid on

subplot(3,1,3)
plot(z_intl)
grid on

% row is the number of y nodes (along the length).
row = length(y_intl)-1;

% col is the no. of x-nodes (across the plate width).
col = length(x_intl)-1;

% surface is the no. of z-nodes (across the thickness).
surface = length(z_intl)-1;

```

```

% The value of teta determines the numerical scheme as follows:

% for Fully Implicit Method; teta=1
% for Crank-Nicholson Method; teta=0.5
% for Explicit Method; teta=0

teta=1;

% For a a*b*c three dimensional matrix,
% a:num rows, b:num cols, c:num surfaces.
% Size of new -----> (a*b*c)*3    TP, TE, TW coefficients for the
%                               whole grids.
% Size of top -----> (a*b*c)*1    TT coefficients for the whole
%                               grids.
% size of bottom --> (a*b*c)*1    TB coefficients for the
%                               whole grids.
% size of north ---> (a*b*c)*1    TN coefficients for the whole
%                               grids.
% size of south ---> (a*b*c)*1    TS coefficients for the whole
%                               grids.
% size of right ---> (a*b*c)*1    The values of the righthandside
%                               of the matrix.
% size of temp ----> (a*b)*c      The temperatures of the grids.

% new is the coefft matrix for all the grid points in the 3-D
% volume.

% 1st col holds the coeffts of the West (W) nodes.
% 2nd col holds the coeffts of the (P) nodes.
% 3rd col holds the coeffts of the East (E) nodes.
new=zeros(row*col*surface,3);

% right is the const (or b-vector) on the RHS
right=zeros(row*col*surface,1);

%% south, north, top, bottom hold the coeffts of the corresponding %%
grid points.
south=right;
north=right;
top=right;
bottom=right;

% Apply the initial condition to the plate.
% "temp" is the temperature matrix.

% temp is a matrix that holds temperature values for each surface.
% Note dimensions of temp carefully.
% This initialization is not present "aath10goon.m".
temp=T0*ones(row*surface,col);

```

```

% Below are parameters that specify the heat/arc source.
% r0 is the effective radius [m] of the gaussian heat source.
r0=0.0045;

% ddd is a coefft that determines the gaussian spread.
% The larger the ddd value, the more pointed the source.
ddd=3;

% grate_total is the total energy transfer rate from the source
% [Watts].
grate_total = 2544;

% q0 is the peak heat flux of the gaussian [W/m^2]
q0=grate_total*ddd/pi/r0^2;

% Calculate the q_up (heat flux distribution on the top face)
%% xcenter and ycenter are the grid element numbers of the grid
%% center at which the source/arc is located.
xcenter=length(int_side);
ycenter=length(int_front);

for nn=1:row
    for mm=1:col

% x(mm) is array with the x-coords of the mesh.
        if mm < xcenter
            x(mm)=-sum(x_intl((mm+1):xcenter));
        elseif mm==xcenter
            x(mm)=0;
        else
            x(mm)=sum(x_intl((xcenter+1):mm));
        end

% y(mm) is array with the y-coords of the mesh.
        if nn < ycenter
            y(nn)=sum(y_intl((nn+1):ycenter));
        elseif nn==ycenter
            y(nn)=0;
        else
            y(nn)=-sum(y_intl((ycenter+1):nn));
        end

% initialize q_up using x() and y() coord info.
        q_up(nn,mm)=-q0*exp(-ddd*(x(mm)^2+y(nn)^2)/r0^2);

    end
end
figure

% a mesh command to view the generated grid.
mesh(x,y,q_up)

```

```

for time=delta_t:delta_t:200
% 200 is the final time in secs upto which the program should
% go...

save_me=save_me+1;

disp('the interpolation time is ')
tic % beginning of tic-toc loop

    [temp]=old_temp10(row,col,surface,temp,x,y,deltax,deltay);

toc % end of tic-toc loop.

% figure
% contour(x,y,temp(1:row,1:col))

disp('The matrix formation time is :')

tic % beginning of tic-toc loop.

%% Start to form the matrix by defining every point.
% Beginning of creation of coefft matrix
% loops step through each point on all the surfaces, cols and rows

for kk=1:surface
for jj=1:row
for ii=1:col

% The distances between the reference point and the adjoining
% nodal points.

% Distance between P&E and P&W
xpe=x_intl(ii+1);
xwp=x_intl(ii);

% Distance between P&N and P&S
ypn=y_intl(jj);
ysp=y_intl(jj+1);

% Distance between P&T and P&B
zpt=z_intl(kk);
zbp=z_intl(kk+1);

%% The lateral surface areas of the faces of the control volume
%% around the reference point P.

% East-face area.
Ae=(ypn/2+ysp/2)*(zpt/2+zbp/2);
Aw=Ae;

```

```

% North-face area.
An=(xpe/2+xwp/2)*(zpt/2+zbp/2);
As=An;
% Top-face area.
At=(xpe/2+xwp/2)*(ypn/2+ysp/2);
Ab=At;

% The size of the control volume.

deltaV=(xpe/2+xwp/2)*(ypn/2+ysp/2)*(zpt/2+zbp/2);

% Routine to set thermal conductivities below:

% Initialize TEMP()s to 0 for logical ifs below:
TEMP(1)=0;TEMP(2)=0;TEMP(3)=0;TEMP(4)=0;TEMP(5)=0;TEMP(6)=0;

% TP0 is the temperature of the current node.
TP0=temp((kk-1)*row+jj,ii);

%
%
%   T_west=TEMP(1)      T_east=TEMP(2)      T_north=TEMP(3)
%   T_south=TEMP(4)    T_top=TEMP(5)       T_bottom=TEMP(6)
%
%

if ii==1 % first col, no west neighbor, so...
    TEMP(1)=TP0; % set T_west to TP0
elseif ii==col % last col, no east neighbor, so.....
    TEMP(2)=TP0; % set T_east to TP0
end

if jj==1
    TEMP(3)=TP0;
elseif jj==row
    TEMP(4)=TP0;
end

if kk==1
    TEMP(5)=TP0;
elseif kk==surface
    TEMP(6)=TP0;
end

%% If current node P has a valid neighbor, then use temperature
%% of that node....

if (TEMP(1)==0)
    TEMP(1)=temp((kk-1)*row+jj,ii-1);
end

```

```

if (TEMP(2)==0)
    TEMP(2)=temp((kk-1)*row+jj,ii+1);
end

if (TEMP(3)==0)
    TEMP(3)=temp((kk-1)*row+jj-1,ii);
end

if (TEMP(4)==0)
    TEMP(4)=temp((kk-1)*row+jj+1,ii);
end

if (TEMP(5)==0)
    TEMP(5)=temp((kk-2)*row+jj,ii);
end

if (TEMP(6)==0)
    TEMP(6)=temp(kk*row+jj,ii);
end

% Do loop to calculate k-values of all grid points.
for kjl=1:6

% Now set the thermal conductivity values.

    kelvin=TP0+273.15;
    k1=53; % nominal k1-value in W/m-K set here, but....

% Actual k1-values are calc below.
    if (kelvin>=300 & kelvin<1000)
        k1=53-0.04*(kelvin-300);
    elseif (kelvin>=1000 & kelvin<1800)
        k1=25+6.25E-3*(kelvin-1000);
    elseif kelvin>=1800
        k1=125;
    end

% Find the thermal conductivity at the adjacent point.
    kelvin=TEMP(kjl)+273.15;
    k2=53; % nominal k2-value in W/m-K set here, but....

% Actual k2-values of neighbors calc here.
    if (kelvin>=300 & kelvin<1000)
        k2=53-0.04*(kelvin-300);
    elseif (kelvin>=1000 & kelvin<1800)
        k2=25+6.25E-3*(kelvin-1000);
    elseif kelvin>=1800
        k2=125;
    end
end

```

```

% Harmonic mean thermal conductivity found next
NEW_K(kj1) = 2*k1*k2/(k1+k2);

end

% Now redefine k's of the neighboring faces with values from NEW_K
kw=NEW_K(1);
ke=NEW_K(2);
kn=NEW_K(3);
ks=NEW_K(4);
kt=NEW_K(5);
kb=NEW_K(6);

% Coefficients used in developing the discretised equations.
ae=ke*Ae/xpe;
aw=kw*Aw/xwp;
an=kn*An/ypn;
as=ks*As/ysp;
at=kt*At/zpt;
ab=kb*Ab/zbp;
ap0=ro*c*deltaV/delta_t;
ap=teta*(ae+aw+an+as+at+ab)+ap0;

% Coefficients used for the wall-medium interface part of the
% discretised equations.
coeffe=he*xpe/(2*ke+he*xpe);
coeffw=hw*xwp/(2*kw+hw*xwp);
coeffn=hn*ypn/(2*kn+hn*ypn);
coeffs=hs*ysp/(2*ks+hs*ysp);
coefft=ht*zpt/(2*kt+ht*zpt);
coeffb=hb*zbp/(2*kb+hb*zbp);

% Coefficients used for the heat flux part of the discretised
% equations.

fluxcoeffe=xpe/(2*ke+he*xpe);
fluxcoeffw=xwp/(2*kw+hw*xwp);
fluxcoeffn=ypn/(2*kn+hn*ypn);
fluxcoeffs=ysp/(2*ks+hs*ysp);
fluxcoefft=zpt/(2*kt+ht*zpt);
fluxcoeffb=zbp/(2*kb+hb*zbp);

% Radiation coefft for surface b.c. - see thesis...
radcoeff=sil*epsilon*zpt/(2*kt+ht*zpt);

% row_num is the index of the current point in the matrices being %
% used. row_num varies from 1:surface*col*row
row_num=(kk-1)*row*col+(jj-1)*col+ii;

% For left-back-top corner

if ((kk==1) & (jj==1) & (ii==1))

```

```

% Temperature values for current time step for each relevant node.
TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

% Coeffts of grid points; note south and bottom later are moved to
% the rhs.
% "new" is the A-matrix in the solution.
% Note teta is the parameter which determines the numerical method
% teta=1:fully implicit; teta=0:explicit; teta=0.5:Crank-Nicholson
new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+2*an*coeffn+as+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

% "right" is the constant b-matrix on the rhs...
right(row_num,1)=(1-teta)*(ae*TE0+as*TS0+ab*TB0)+ ...
    (ap0-(1-teta)*(ae+2*aw*coeffw+2*an*coeffn+as+2*at*coefft+ab))*TP0+ ...
    (2*aw*coeffw+2*an*coeffn+2*at*coefft)*Tinf -
    2*at*fluxcoefft*q_up(jj,ii) ...
    -2*aw*fluxcoeffw*q_west-2*an*fluxcoeffn*q_north-2*at*fluxcoefft*q_top;

%% radiation & q_boiling_coeff belong to the rhs but are not included
%% until later since they depend on the current temperature

% (a form of quasi-linearization of nonlinear temp terms)
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

% for right-back-top corner

elseif ((kk==1) & (jj==1) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+2*an*coeffn+as+2*at*coefft+ab);
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+as*TS0+ab*TB0)+ ...
    (ap0-(1-teta)*(2*ae*coeffe+aw+2*an*coeffn+as+2*at*coefft+ab))*TP0+
    ...
    (2*ae*coeffe+2*an*coeffn+2*at*coefft)*Tinf-
    2*at*fluxcoefft*q_up(jj,ii)...
    -2*ae*fluxcoeffe*q_east-2*an*fluxcoeffn*q_north-

    2*at*fluxcoefft*q_top;
radiation(row_num,1)=-2*at*radcoeff;

```

```

q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%   for left-front-top corner

elseif ((kk==1) & (jj==row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+2*as*coeffs+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
north(row_num,1)=teta*an;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+ab*TB0)+ ...
    (ap0-(1-teta)*(ae+2*aw*coeffw+an+2*as*coeffs+2*at*coefft+ab))*TP0+ ...
    (2*aw*coeffw+2*as*coeffs+2*at*coefft)*Tinf-2*at*fluxcoefft*q_up(jj,ii)
...
-2*aw*fluxcoeffw*q_west-2*as*fluxcoeffs*q_south-2*at*fluxcoefft*q_top;
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%   for right-front-top corner

elseif ((kk==1) & (jj==row) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+2*as*coeffs+2*at*coefft+ab);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+ab*TB0)+ ...
    (ap0-(1-teta)*(2*ae*coeffe+aw+an+2*as*coeffs+2*at*coefft+ab))*TP0+
...
    (2*ae*coeffe +2*as*coeffs +2*at*coefft)*Tinf-
2*at*fluxcoefft*q_up(jj,ii)...
-2*ae*fluxcoeffe*q_east-2*as*fluxcoeffs*q_south-
2*at*fluxcoefft*q_top;
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

```

```

% for left-back-bottom corner

elseif ((kk==surface) & (jj==1) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+2*an*coeffn+as+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(ae*TE0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(ae+2*aw*coeffw+2*an*coeffn+as+at+2*ab*coeffb))*TP0+ ...
(2*aw*coeffw+2*an*coeffn+2*ab*coeffb)*Tinf ...
-2*aw*fluxcoeffw*q_west-2*an*fluxcoeffn*q_north-
2*ab*fluxcoeffb*q_bottom;

% for right-back-bottom corner

elseif ((kk==surface) & (jj==1) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+2*an*coeffn+as+at+2*ab*coeffb);
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(aw*TW0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(2*ae*coeffe+aw+2*an*coeffn+as+at+2*ab*coeffb))*TP0+
...
(2*ae*coeffe+2*an*coeffn+2*ab*coeffb)*Tinf ...
-2*ae*fluxcoeffe*q_east-2*an*fluxcoeffn*q_north-
2*ab*fluxcoeffb*q_bottom;

% for left-front-bottom corner

elseif ((kk==surface) & (jj==row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+2*as*coeffs+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;

```

```

north(row_num,1)=teta*an;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+at*TT0)+ ...
(ap0-(1-teta)*(ae+2*aw*coeffw+an+2*as*coeffs+at+2*ab*coeffb))*TP0+ ...
(2*aw*coeffw+2*as*coeffs+2*ab*coeffb)*Tinf ...
-2*aw*fluxcoeffw*q_west-2*as*fluxcoeffs*q_south-
2*ab*fluxcoeffb*q_bottom;

%   for right-front-bottom corner

elseif ((kk==surface) & (jj==row) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+2*as*coeffs+at+2*ab*coeffb);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+at*TT0)+ ...
(ap0-(1-teta)*(2*ae*coeffe+aw+an+2*as*coeffs+at+2*ab*coeffb))*TP0+
...
(2*ae*coeffe +2*as*coeffs +2*ab*coeffb)*Tinf ...
-2*ae*fluxcoeffe*q_east-2*as*fluxcoeffs*q_south-
2*ab*fluxcoeffb*q_bottom;

%   for the top surface

elseif ((kk==1) & (jj>1) & (jj<row) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+an+as+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+as*TS0+ab*TB0)+ ...
(ap0-(1-teta)*(ae+aw+an+as+2*at*coefft+ab))*TP0 + 2*at*coefft*Tinf -
...
2*at*fluxcoefft*q_up(jj,ii)-2*at*fluxcoefft*q_top;

```

```

radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%      for the bottom surface

elseif ((kk==surface) & (jj>1) & (jj<row) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+an+as+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(ae+aw+an+as+at+2*ab*coeffb))*TP0 + 2*ab*coeffb*Tinf -
...
2*ab*fluxcoeffb*q_bottom;

%      for the front surface

elseif ((kk>1) & (kk<surface) & (jj==row) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+an+2*as*coeffs+at+ab);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+at*TT0+ab*TB0)+ ...
(ap0-(1-teta)*(ae+aw+an+2*as*coeffs+at+ab))*TP0 + 2*as*coeffs*Tinf -
...
2*as*fluxcoeffs*q_south;

%      for the back surface

elseif ((kk>1) & (kk<surface) & (jj==1) & (ii>1) & (ii<col))

```

```

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+2*an*coeffn+as+at+ab);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+as*TS0+at*TT0+ab*TB0)+ ...
    (ap0-(1-teta)*(ae+aw+2*an*coeffn+as+at+ab))*TP0 + 2*an*coeffn*Tinf -
...
    2*an*fluxcoeffn*q_north;

%      for the left-lateral surface

elseif ((kk>1) & (kk<surface) & (jj>1) & (jj<row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+as+at+ab);
new(row_num,3)=-teta*ae;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+as*TS0+at*TT0+ab*TB0)+ ...
    (ap0-(1-teta)*(ae+2*aw*coeffw+an+as+at+ab))*TP0 + 2*aw*coeffw*Tinf -
...
    2*aw*fluxcoeffw*q_west;

%      for the right-lateral surface

elseif ((kk>1) & (kk<surface) & (jj>1) & (jj<row) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

```

```

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+as+at+ab);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+as*TS0+at*TT0+ab*TB0)+ ...
(ap0-(1-teta)*(2*ae*coeffe+aw+an+as+at+ab))*TP0 + 2*ae*coeffe*Tinf -
...
2*ae*fluxcoeffe*q_east;

%      for the front-top edge

elseif ((kk==1) & (jj==row) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+an+2*as*coeffs+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+ab*TB0)+ ...
(ap0-(1-teta)*(ae+aw+an+2*as*coeffs+2*at*coefft+ab))*TP0 + ...
(2*as*coeffs+2*at*coefft)*Tinf - 2*as*fluxcoeffs*q_south-
2*at*fluxcoefft*q_top ...
-2*at*fluxcoefft*q_up(jj,ii);
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%      for the front-bottom edge

elseif ((kk==surface) & (jj==row) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+an+2*as*coeffs+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;

north(row_num,1)=teta*an;
top(row_num,1)=teta*at;

```

```

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+at*TT0)+ ...
(ap0-(1-teta)*(ae+aw+an+2*as*coeffs+at+2*ab*coeffb))*TP0 + ...
(2*as*coeffs+2*ab*coeffb)*Tinf - 2*as*fluxcoeffs*q_south-
2*ab*fluxcoeffb*q_bottom;

%      for the back-top edge

elseif ((kk==1) & (jj==1) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+2*an*coeffn+as+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+as*TS0+ab*TB0)+ ...
(ap0-(1-teta)*(ae+aw+2*an*coeffn+as+2*at*coefft+ab))*TP0 + ...
(2*an*coeffn+2*at*coefft)*Tinf - 2*an*fluxcoeffn*q_north-
2*at*fluxcoefft*q_top...
-2*at*fluxcoefft*q_up(jj,ii);
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%      for the back-bottom edge

elseif ((kk==surface) & (jj==1) & (ii>1) & (ii<col))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+aw+2*an*coeffn+as+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(ae+aw+2*an*coeffn+as+at+2*ab*coeffb))*TP0 + ...
(2*an*coeffn+2*ab*coeffb)*Tinf - 2*an*fluxcoeffn*q_north-
2*ab*fluxcoeffb*q_bottom;

```

```

%      for the front-left edge

elseif ((kk>1) & (kk<surface) & (jj==row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+2*as*coeffs+at+ab);
new(row_num,3)=-teta*ae;
north(row_num,1)=teta*an;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+at*TT0+ab*TB0)+ ...
  (ap0-(1-teta)*(ae+2*aw*coeffw+an+2*as*coeffs+at+ab))*TP0 + ...
  (2*aw*coeffw+2*as*coeffs)*Tinf - 2*aw*fluxcoeffw*q_west-
  2*as*fluxcoeffs*q_south;

%      for the front-right edge

elseif ((kk>1) & (kk<surface) & (jj==row) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+2*as*coeffs+at+ab);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+at*TT0+ab*TB0)+ ...
  (ap0-(1-teta)*(2*ae*coeffe+aw+an+2*as*coeffs+at+ab))*TP0 + ...
  (2*ae*coeffe+2*as*coeffs)*Tinf - 2*ae*fluxcoeffe*q_east-
  2*as*fluxcoeffs*q_south;

%      for the back-left edge

elseif ((kk>1) & (kk<surface) & (jj==1) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

```

```

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+2*an*coeffn+as+at+ab);
new(row_num,3)=-teta*ae;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+as*TS0+at*TT0+ab*TB0)+ ...
  (ap0-(1-teta)*(ae+2*aw*coeffw+2*an*coeffn+as+at+ab))*TP0 + ...
  (2*aw*coeffw+2*an*coeffn)*Tinf -2*aw*fluxcoeffw*q_west-
  2*an*fluxcoeffn*q_north;

%      for the back-right edge

elseif ((kk>1) & (kk<surface) & (jj==1) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+2*an*coeffn+as+at+ab);
new(row_num,1)=-teta*aw;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+as*TS0+at*TT0+ab*TB0)+ ...
  (ap0-(1-teta)*(2*ae*coeffe+aw+2*an*coeffn+as+at+ab))*TP0 + ...
  (2*ae*coeffe+2*an*coeffn)*Tinf - 2*ae*fluxcoeffe*q_east-
  2*an*fluxcoeffn*q_north;

%      for the left-lateral-top edge

elseif ((kk==1) & (jj>1) & (jj<row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+as+2*at*coefft+ab);
new(row_num,3)=-teta*ae;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+as*TS0+ab*TB0)+ ...
  (ap0-(1-teta)*(ae+2*aw*coeffw+an+as+2*at*coefft+ab))*TP0 + ...
  (2*aw*coeffw+2*at*coefft)*Tinf - 2*aw*fluxcoeffw*q_west-
  2*at*fluxcoefft*q_top...
  -2*at*fluxcoefft*q_up(jj,ii);

```

```

radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%      for the left-lateral-bottom edge

elseif ((kk==surface) & (jj>1) & (jj<row) & (ii==1))

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(ae+2*aw*coeffw+an+as+at+2*ab*coeffb);
new(row_num,3)=-teta*ae;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(ae*TE0+an*TN0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(ae+2*aw*coeffw+an+as+at+2*ab*coeffb))*TP0 + ...
(2*aw*coeffw+2*ab*coeffb)*Tinf - 2*aw*fluxcoeffw*q_west-
2*ab*fluxcoeffb*q_bottom;

%      for the right-lateral-top edge

elseif ((kk==1) & (jj>1) & (jj<row) & (ii==col))

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+as+2*at*coefft+ab);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+as*TS0+ab*TB0)+ ...
(ap0-(1-teta)*(2*ae*coeffe+aw+an+as+2*at*coefft+ab))*TP0 + ...
(2*ae*coeffe+2*at*coefft)*Tinf - 2*ae*fluxcoeffe*q_east-
2*at*fluxcoefft*q_top...
-2*at*fluxcoefft*q_up(jj,ii);
radiation(row_num,1)=-2*at*radcoeff;
q_boiling_coeff(row_num,1)=-2*at*fluxcoefft;

%      for the right-lateral-bottom edge

elseif ((kk==surface) & (jj>1) & (jj<row) & (ii==col))

```

```

TP0=temp((kk-1)*row+jj,ii);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);

new(row_num,2)=ap0+teta*(2*ae*coeffe+aw+an+as+at+2*ab*coeffb);
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;

right(row_num,1)=(1-teta)*(aw*TW0+an*TN0+as*TS0+at*TT0)+ ...
(ap0-(1-teta)*(2*ae*coeffe+aw+an+as+at+2*ab*coeffb))*TP0 + ...
(2*ae*coeffe+2*ab*coeffb)*Tinf - 2*ae*fluxcoeffe*q_east-
2*ab*fluxcoeffb*q_bottom;

%      for the interior nodes
%% as expected, note that the interior node has all of its neighbors
%% contributing to the coefft matrices
else

TP0=temp((kk-1)*row+jj,ii);
TE0=temp((kk-1)*row+jj,ii+1);
TW0=temp((kk-1)*row+jj,ii-1);
TN0=temp((kk-1)*row+jj-1,ii);
TS0=temp((kk-1)*row+jj+1,ii);
TT0=temp((kk-2)*row+jj,ii);
TB0=temp(kk*row+jj,ii);

new(row_num,2)=ap;
new(row_num,3)=-teta*ae;
new(row_num,1)=-teta*aw;
north(row_num,1)=teta*an;
south(row_num,1)=teta*as;
top(row_num,1)=teta*at;
bottom(row_num,1)=teta*ab;

right(row_num,1)=(1-teta)*(ae*TE0+aw*TW0+an*TN0+as*TS0+at*TT0+ab*TB0)+
...
(ap0-(1-teta)*(ae+aw+an+as+at+ab))*TP0;

end      % the "end" of if loop to decide where node is located...

end      % the "end" of loop for ii= ...(cols)

end      % the "end" of loop for jj= ...(rows)

end      % the "end" of loop for kk= ...(surfaces)

toc      % end of tic-toc loop

```

```

%% solve the matrix and find the new temperatures by using the
"aasolve10.m"
%% function - row-by-row sweeping iterations from surface to surface

    [temp]=aasolve10(row,col,surface,new,right,north,south,top,bottom,
    ...
        temp,radiation,q_boiling_coeff,Tinf,x_intl,y_intl,x,y);

% print the maximum temp in the entire 3-d grid
    max(max(temp))

%% save the row number, column_number, surface number, time and
temperature
%% in every iteration

save_me % print value of save_me counter

% time_step_interval is the freq at which to save temp data
time_step_interval=1;

if (save_me/time_step_interval) == fix(save_me/time_step_interval)

%% save_counter is used in "save_the10.m" function to apend to file-
name
%% in which temp data is being saved
    save_counter=save_counter+1;

% save_the10 saves top-surface temp at each time-step into a diff file
name

save_the10(row,col,surface,time,temp(1:row,1:col),save_counter,b,x,y);

%% save_all10 saves all 3-d temp data at each time-step by overwriting
onto
%% the same file name - imp for restarting the iterations...

save_all10(number,time,temp,b,delta_t,back,front,width,thickness,save_c
ounter,save_me);

end

num_milli_secs=500; % freq at which to save all 3-d temp data
for aa=1:20

    if (round(save_me)==round(num_milli_secs*aa))
        save_the10(row,col,surface,time,temp,save_counter,b,x,y);
    end

end

end

```

```

%      THE FUNCTION PROGRAM (aasolve10.m)

% this program solves the matrix produced by the main program
% by using iterative sweeping method.
%

function [temp] = aasolve10(row,col,surface,new,right,north,south,top,
...

bottom,temp,radiation,q_boiling_coeff,Tinf,x_intl,y_intl,x,y);

%   for a a*b*c  three dimensional matrix,
%
%   size of new ----> (a*b*c)*3   TP, TE, TW coefficients for the
%                                   whole grids
%   size of top ----> (a*b*c)*1   TT coefficients for the whole grids
%   size of bottom --> (a*b*c)*1  TB coefficients for the whole grids
%   size of north ---> (a*b*c)*1  TN coefficients for the whole grids
%   size of south ---> (a*b*c)*1  TS coefficients for the whole grids
%   size of right ---> (a*b*c)*1  The values of the righthandside of
%                                   the matrix
%   size of temp ----> (a*b)*c    The temperatures of the grids
%   size of radiation --> (a*b)*1  The radiation coefficients
%   size of q_boiling_coeff --> (a*b)*1  The boiling and free
%   convection heat fluxes from the top surface

% initialize the temp_old, temp_old is the old temperatures of the
% grid points. It is used to compare the temperatures before and after
% the iterations. If the difference is less than 0.1, the iteration
% is enough.

temp_old=temp+100*ones(size(temp));

it_num=0;
temp_saturation=100;

delta_iteration=0.1;
count_iteration=0;

while max(max(abs(temp_old-temp))) > delta_iteration
count_iteration = count_iteration+1;

if count_iteration>delta_iteration*50+5
    delta_iteration=delta_iteration+0.1
end

max(max(abs(temp_old-temp)))

it_num=it_num+1          % iteration number

```

```
disp(max(max(temp)))
temp_old=temp;
```

```
% The variables of the boiling regimes
```

```
g=9.81;
csf=0.0132;
ne=1;
hfg=2257E3;
ro_l=957.85;
ro_v=0.6;
surface_tension=58.9E-3;
Prandtl=1.75;
miw_l=2.775E-4;
miw_v=12.02E-6;
cpl=4211;
cpv=2029;
epsi_s=0.82;
k_liq=0.682;
k_vap=0.0249;
alpha_l=0.169E-6;
T_sat=100;
T_liquid=27;
sigma_s=5.67E-8;
```

```
pisi_pisi=zeros(row,col);
```

```
% start the iterations
```

```
for cz=1:surface % for the surfaces
```

```
for cy=1:row % for the rows
```

```
index1=(cz-1)*row*col+(cy-1)*col+1;
index2=(cz-1)*row*col+cy*col;
```

```
new1=new(index1:index2,:);
top1=top(index1:index2,1);
bottom1=bottom(index1:index2,1);
north1=north(index1:index2,1);
south1=south(index1:index2,1);
right1=right(index1:index2,1);
radiation1=zeros(length(index1:index2),1);
q_boiling_free=zeros(length(index1:index2),1);
```

```
for cx=1:col % for the columns
```

```

% multiply the coefficients by the temperatures

if cz==1

    qqqq=0; % heat flux to the outside because of boiling or free
    convection

% The radiation heat flux from the top surface

radiation1(1:col,1)=radiation((cy-
1)*col+1):(cy*col),1).*(temp(cy,cx)^4);
radiation1(cx,1)=radiation((cy-1)*col+cx,1).*(temp(cy,cx)^4);

% The boiling or free convection heat flux from the top surface

delta_temp = temp(cy,cx) - temp_saturation;

if (delta_temp<=5)

% Rayleigh number

    T_film=(temp(cy,cx)+Tinf)/2+273.15;
    g=9.81;
    beta=-0.000000017937*(T_film^2)+0.0000188*T_film-0.0037515;
    alpha=-1.875E-12*(T_film^2)+1.5525E-9*T_film-1.4995E-7;
    nu=1.125E-10*(T_film^2)-8.285E-8*T_film+1.5584E-5;
    kconduc=-0.000008125*(T_film^2)+0.0063775*T_film-0.56895;

% characteristic length L
% The free convection coefficient will be found for a 1*1 m2 area

    L=1/4;

    Ra=abs(g*beta*(temp(cy,cx)-Tinf)*(L^3)/(nu*alpha));

    if Ra<16

        Nusselt=1;

    else

        if (Ra<=1E7)
            Nusselt=0.54*(realpow(Ra,0.25));
        else
            Nusselt=0.15*(realpow(Ra,(1/3)));
        end

    end

    h_boiling=Nusselt*kconduc/L;

```

```

q_boiling_free(cx,1)=q_boiling_coeff((cy-1)*col+cx,1)*...
    h_boiling*(temp(cy,cx)-Tinf);

qqqq=h_boiling*(temp(cy,cx)-Tinf);

end

if (delta_temp>5 & delta_temp<=30)

q_prime_s=miw_l*hfg*sqrt(g*(ro_l-ro_v)/surface_tension)*...
    (cpl*delta_temp/csf/hfg/Prandtl^ne)^3;

tau=pi/3*sqrt(2*pi)*sqrt(surface_tension/g/(ro_l-ro_v))*...
    (realpow(abs(ro_v^2/surface_tension/g/(ro_l-
ro_v)),0.25));

q_boiling_free(cx,1)=q_boiling_coeff((cy-
1)*col+cx,1)*q_prime_s*...
    (1+(2*k_liq*(T_sat-T_liquid)/...
sqrt(pi*alpha_l*tau))*24/(pi*hfg*ro_v)*...
realpow(abs(ro_v^2/surface_tension/g/(ro_l-
ro_v)),.25));

qqqq=q_prime_s*...
    (1+(2*k_liq*(T_sat-T_liquid)/...
sqrt(pi*alpha_l*tau))*24/(pi*hfg*ro_v)*...
realpow(abs(ro_v^2/surface_tension/g/(ro_l-
ro_v)),.25));

end

if (delta_temp>30 & delta_temp<=120)

q_max=0.149*hfg*ro_v*realpow(abs(surface_tension*g*(ro_l-
ro_v)/ro_v^2),0.25);

q_min=0.09*ro_v*hfg*realpow(abs(surface_tension*g*(ro_l-ro_v)/...
(ro_l+ro_v)^2),0.25);

1
log_q=log10(q_max/q_min)/log10(30/120)*log10(delta_temp/120)+log10(q_mi
n);
q_boiling_free(cx,1)=q_boiling_coeff((cy-1)*col+cx,1)*10^(log_q);
qqqq=10^(log_q);
end

if (delta_temp>120)
lambda=2*pi*realpow(abs(surface_tension/g/(ro_l-ro_v)),0.5);

```

```

h_conduct=0.59*realpow(abs(g*(ro_lro_v)*ro_v*k_vap^3*...
(hfg+0.68*cpv*delta_temp)/(lambda*miw_v*delta_temp)),.25);

h_radiate=sigma_s*epsi_s*((temp(cy,cx))^4-(T_sat)^4)/...
(temp(cy,cx)-T_sat));

h_boiling=h_conduct+0.75*h_radiate;

q_boiling_free(cx,1)=q_boiling_coeff((cy-1)*col+cx,1)* ...
h_boiling*(temp(cy,cx)-Tinf);

qqqq=h_boiling*(temp(cy,cx)-Tinf);
end

pisi_pisi(cy,cx)=qqqq;
end

if cz>1
    topl(cx,1)=topl(cx,1)*temp((cz-2)*row+cy,cx);
end

if cz<surface
    bottom1(cx,1)=bottom1(cx,1)*temp(cz*row+cy,cx);
end

if cy>1
    north1(cx,1)=north1(cx,1)*temp((cz-1)*row+cy-1,cx);
end

if cy<row
    south1(cx,1)=south1(cx,1)*temp((cz-1)*row+cy+1,cx);
end

end

% index1...index2 shows the rows in the new matrix that the
% calculations are done

if cz==1

right1(1:col,1)=right1(1:col,1)+radiation1(1:col,1)+q_boiling_free(1:co
1,1);

end

right1(1:col,1)=right1(1:col,1)+north1(1:col,1) ...
+south1(1:col,1)+topl(1:col,1)+bottom1(1:col,1);

```

```

% solve the diagonal matrix by using Gauss elimination method

for i=1:col-1;
    newl(i,3)=newl(i,3)/newl(i,2);
    rightl(i,1)=rightl(i,1)/newl(i,2);
    newl(i,2)=1;

    newl(i+1,2)=newl(i+1,2)-newl(i+1,1)*newl(i,3);
    rightl(i+1,1)=rightl(i+1,1)-rightl(i,1)*newl(i+1,1);
    newl(i+1,1)=0;

end

    rightl(col,1)=rightl(col,1)/newl(col,2);
    newl(col,2)=1;

for i=col:-1:2
    rightl(i-1,1)=rightl(i-1,1)-rightl(i,1)*newl(i-1,3);
    newl(i-1,3)=0;
end

% The calculation is over, update the temperature matrix for the new
values

temp((cz-1)*row+cy,:)=rightl(1:col,:);

end

end

end

% figure
% mesh(x,y,pisi_pisi);
% pisi_pisi

% title('heat')

```

```

%      THE FUNCTION PROGRAM (old_temp10.m)
%
%% The function program (old_temp10.m) finds out the new positions of
%% the moving grid points at the new time step and the old temperature
%% values of the new grid points by using second degree polynomial
%% interpolation.

function [temp]=old_temp10(row,col,surface,temp,x,y,deltax,deltay);

%% This (for) loop finds out the x-coordinates of the new area where
%% the grid points move after the time delta_t.

for for1=1:col

    new_pos_x(1,for1)=x(1,for1)+deltax;

    count = 1;

    while ((x(1,count) < new_pos_x(1,for1)) & (count<col+10))

        if count==col
            count=col+20;      % There was "break" command here before.
        end

        count=count+1;
    end

    if count>=col+20
        count=col;
    end

    if (deltax>0)

        if count>2                % deltax>0 means if the source goes to
the right
            count = count-2;
        else
            if count > 1
                count =count-1;
            end
        end
    end

    if (deltax<=0)                % deltax>0 means if the source goes to
the right

        if count==col
            count=count-2;

```

```

else
    if count>1
        count =count-1;
    end
end

end

pos_x(for1) = count;

end

%% This (for) loop finds out the y-coordinates of the new area where
%% the grid points move after the time delta_t.

for1=0;

for for1=1:1:row

    new_pos_y(for1)=y(1,for1)+deltay;

    count = 1;

    while ((y(1,count) > new_pos_y(1,for1)) & (count<row+10))

        if count==row
            count=row+20;
        end

        count=count+1;
    end

    if count>=row+20
        count=row;
    end

    if (deltay>0)

        if count==row
            count=count-2;

        else
            if count>1
                % deltax>0 means if the source goes to
                % the right

                count = count-1;
            end
        end
    end

    if (deltay<=0)
        % deltax>0 means if the source goes to the
        % right
    end
end

```

```

        if count>2
            count=count-2;
        else
            if count==2
                count =count-1;
            end
        end
    end

    end

    pos_y(for1) = count;

end

%% Now, we know the coordinates of the area where the grid points
%% move. We have to use second degree polynomial interpolation to find
%% the temperature values of the new grid points. Here, we use the
%% temperature values from the previous time step.

for for1=1:surface

    for for2=1:row

        for for3=1:col

            % temperatures
            y0=temp((for1-1)*row+pos_y(for2),for3);
            y1=temp((for1-1)*row+pos_y(for2)+1,for3);
            y2=temp((for1-1)*row+pos_y(for2)+2,for3);

            % positions
            x0=y(pos_y(for2));
            x1=y(pos_y(for2)+1);
            x2=y(pos_y(for2)+2);

            % our second degree polynomial is  $y=a*x^2+b*x+c$ 
            % a, b, c are the coefficients. Now find these coefficients...
            b=(y0*(x2^2-x1^2)+y1*(x0^2-x2^2)+y2*(x1^2-x0^2))/...
                (x0*(x2^2-x1^2)+x1*(x0^2-x2^2)+x2*(x1^2-x0^2));

            if abs(x0)~=abs(x2)
                a=(y0-y2+b*(x2-x0))/(x0^2-x2^2);
            else
                a=(y0-y1+b*(x1-x0))/(x0^2-x1^2);
            end

            c=y0-a*x0^2-b*x0;

            temp_old((for1-
1)*row+for2,for3)=a*new_pos_y(for2)^2+b*new_pos_y(for2)+c;

```

```

% [pol_coeff]=polyfit(y(pos_y(for2):(pos_y(for2)+2)), ...
% temp(((for1-1)*row+pos_y(for2)):(for1-
1)*row+pos_y(for2)+2)),for3)',2);
% temp_old((for1-
1)*row+for2,for3)=polyval(pol_coeff,new_pos_y(for2));

    end
end

end

for for1=1:surface

for for2=1:row

    for for3=1:col

        y0=temp_old((for1-1)*row+for2,pos_x(for3));
        y1=temp_old((for1-1)*row+for2,pos_x(for3)+1);
        y2=temp_old((for1-1)*row+for2,pos_x(for3)+2);

        x0=x(pos_x(for3));
        x1=x(pos_x(for3)+1);
        x2=x(pos_x(for3)+2);

        b=(y0*(x2^2-x1^2)+y1*(x0^2-x2^2)+y2*(x1^2-x0^2))/ ...
        (x0*(x2^2-x1^2)+x1*(x0^2-x2^2)+x2*(x1^2-x0^2));

        if abs(x0)~=abs(x2)
            a=(y0-y2+b*(x2-x0))/(x0^2-x2^2);
        else
            a=(y0-y1+b*(x1-x0))/(x0^2-x1^2);
        end

        c=y0-a*x0^2-b*x0;

        temp((for1-
1)*row+for2,for3)=a*new_pos_x(for3)^2+b*new_pos_x(for3)+c;

% [pol_coeff]=polyfit(x(pos_x(for3):(pos_x(for3)+2)), ...
% temp_old((for1-
1)*row+for2,((pos_x(for3)):(pos_x(for3)+2))),2);
% temp((for1-1)*row+for2,for3)=polyval(pol_coeff,new_pos_x(for3));

        end
    end

end

end

```

```

%           THE FUNCTION PROGRAM (save_all10.m)
%
%% the function program (save_all10.m) saves all 3-d temp data at each
%% time-step by overwriting onto

function
[] = save_all10(number, time, temp, b, delta_t, back, front, width, thickness, save_counter, save_me);

save save_all10

```

```

%           THE FUNCTION PROGRAM (save_the10.m)
%
% the function program (save_the10) saves top-surface temp at each
% time-step into a diff file name
%% (aath10.m) and (aath10goon.m) can not be compiled without putting
%% "%" symbol in front of save....,after compiling erase "%" symbol and
%% save the program again.

function []=save_the10(row,col,surface,time,temp,save_counter,b,x,y);

save (['temperature10_',num2str(save_counter)]);

```


LIST OF REFERENCES

- [1] Brown, R.T. and Masubuchi, K., "Fundamental Research on Underwater Welding," *Welding Journal*, Vol. 54, No. 6, pp. 178s-188s, June 1975.
- [2] Tsai, C.L. and Masubuchi, K., "Mechanisms of Rapid Cooling and Their Design Considerations in Underwater Welding," *11th Annual Offshore Technology Conference*, OTC 3469, Houston, TX, 1979.
- [3] "Underwater Cutting and Welding," *U.S. Navy Technical Manual*, U.S.N. Supervisor of Diving, Naval Ship Systems Command.
- [4] Brown, A.J., Staub, J.A. and Masubuchi, K., "Fundamental Study of Underwater Welding," *4th Annual Offshore Technology Conference*, OTC 1621, Houston, TX, 1972.
- [5] Rosenthal, D., "The Theory of Moving Sources of Heat and Its Application to Metal Treatments," *Transactions of The A.S.M.E.* Vol. 68, pp. 849-866, November 1946.
- [6] Masubuchi, K., *Analysis of Welded Structures*, 1st Ed. London, Pergamon Press, 1980.
- [7] Kou, S., *Welding Metallurgy*, John Wiley and Sons, 1987.
- [8] Oreper, G.M. and Szekely, J., "Heat and Fluid-Flow Phenomena in Weld Pools," *Journal of Fluid Mechanics*, Vol.147, pp. 53-79, 1984.
- [9] Kou, S. and Wang, Y.H., "Computer Simulation of Convection in Moving Arc Weld Pools," *Metallurgical Transactions A*, Vol. 17A, pp. 2271-2277, December 1986.

- [10] Kou, S. and Wang, Y.H., "Three-Dimensional Convection in Laser Melted Pools," *Metallurgical Transactions A*, Vol. 17A, pp. 2265-2270, December 1986.
- [11] Correa, S.M. and Sundell, R.E., "A Computational and Experiment Study of The Fluid Flow in Weld Pools," S. Kou and R. Mehrabian. Eds., *Modeling and Control of Casting and Welding Processes*, Metallurgical Society, pp. 211-227, Warrendale, PA, 1986.
- [12] Saedi, H.R. and Unkel, W., " Thermal-Fluid Model for Weld Pool Geometry Dynamics," *Journal of Dynamic Systems, Measurement and Control*, Vol. 111, pp. 268-276, June 1989.
- [13] Zacharia, T., Eraslan, A.H., Aidun, D.K. and David, S.A., " Three-Dimensional Transient Model for Arc Welding Process," *Metallurgical Transactions*, Vol. 20B, pp. 645-659, October 1989.
- [14] Kim, S.D. and Na, S.J., " A Study on Heat and Mass Flow in Stationary Gas Tungsten Arc Welding Using The Numerical Mapping Method," *Journal of Engineering Manufacture*, Part B, pp. 233-242, 1989.
- [15] Ramanan, N. and Korpela, S.A., " Fluid Dynamics of a Stationary Weld Pool," *Metallurgical Transactions*, Vol. 21A, pp. 45-57, January 1990.
- [16] Ule, R.L., Y. Joshi and E.B. Sedy, "A New Technique for Three-Dimensional Transient Heat Transfer Computations of Autogenous Arc Welding," *Metallurgical Transactions B*, Vol. 21B, pp. 1033-1047, December 1990.
- [17] Kanouff, M. and Greif, R., " The Unsteady Development of a GTA Weld Pool," *International Journal of Heat and Mass Transfer*, Vol. 35, pp. 967-969

[18] Joshi, Y., Dutta, P., Schupp, P.E., Espinosa, D., "Nonaxisymmetric Convection in Stationary Gas Tungsten Arc Weld Pools," *Journal of Heat Transfer*, Vol. 119, pp. 164-172, February 1997.

[19] Espinosa, D.C., Master's Thesis, Naval Postgraduate School, Monterey, CA, 1991.

[20] Versteeg, H.K. and Malalasekera, W., *An Introduction to Computational Fluid Dynamics The Finite Volume Method*, Longman Group Ltd., 1995.

[21] Patankar, S.V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corporation, 1980.

[22] Patankar, S.V., *Computation of Conduction and Duct Flow Heat Transfer*, Innovative Research, Inc., Maple Grove, MN, 1991.

[23] Kreith, F. and Bohn, M.S., *Principles of Heat Transfer*, 5th Ed., West Publishing Company, St. Paul, MN, 1993.

[24] Rohsenow, W.M. and Choi, H., *Heat, Mass and Momentum Transfer*, Prentice-Hall, Inc., New Jersey, 1961.

[25] Incropera, F.P. and DeWitt, D.P., *Introduction to Heat Transfer*, 3rd Ed., John Wiley & Sons, 1996.

[26] Tsai, C.L. and Masubuchi, K., "Mechanisms of Rapid Cooling in Underwater Welding," *Applied Ocean Research*, Vol. 1, No. 2, 1979.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center.....2 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	
2. Dudley Knox Library.....2 Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	
3. Chairman, Code ME.....1 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5000	
4. Professor Ashok Gopinath, Code ME/GK.....2 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93943-5000	
5. Naval/Mechanical Engineering Curricular Office, Code 34.....1 Naval Postgraduate School Monterey, CA 93943-5000	
6. Deniz Kuvvetleri Komutanligi.....2 Personel Daire Baskanligi Bakanliklar Ankara, TURKEY 06100	
7. Deniz Harp Okulu Komutanligi.....1 Kutuphane Tuzla, Istanbul, TURKEY 81704	
8. Yasar Vehbi Isiklar.....2 Feyzullah Mahallesi Esenyurt Sokak Tufekciler Apt. No.16/6 Maltepe, Istanbul, TURKEY	

9. Istanbul Teknik Universitesi Makina Muh.Bl.....1
Ratip Berber Kutuphanesi
Maslak, Istanbul, TURKEY
10. Ibrahim Girgin.....1
Deniz Harp Okulu Komutanligi
Teknik Bilimler Bolum Baskanligi
Makina Ogretim Uyesi
Tuzla, Istanbul, TURKEY 81704